Safe Agents in Space: Preventing and Responding to Anomalies in the Autonomous Sciencecraft Experiment

Daniel Tran, Steve Chien, Gregg Rabideau, Benjamin Cichy
Jet Propulsion Laboratory, California Institute of Technology
Firstname.Lastname@jpl.nasa.gov

Abstract. This paper describes the design of the Autonomous Sciencecraft Experiment, a software agent that has been running on-board the EO-1 spacecraft since 2003. The agent recognizes science events, retargets the spacecraft to respond to the science events, and reduces data downlink to only the highest value science data. The autonomous science agent was designed using a layered architectural approach with specific redundant safeguards to reduce the risk of an agent malfunction to the EO-1 spacecraft. The agent was designed to be "safe" by first preventing anomalies, then by automatically detecting and responding to them when possible. This paper describes elements of the design that increase the safety of the agent, several of the anomalies that occurred during the experiment, and how the agent responded to these anomalies.

1 Introduction

Autonomy technologies have incredible potential to revolutionize space exploration. In the current mode of operations, space missions involve meticulous ground planning significantly in advance of actual operations. In this paradigm, rapid responses to dynamic science events can require substantial operations effort. Artificial intelligence technologies enable onboard software to detect science events, re-plan upcoming mission operations, and enable successful execution of re-planned responses. Additionally, with onboard response, the spacecraft can acquire data, analyze it onboard to estimate its science value, and only downlink the highest priority data. For example, a spacecraft could monitor active volcano sites and only downlink images when the volcano is erupting. Or a spacecraft could monitor ice shelves and downlink images when calving activities are high. Or a spacecraft could monitor river lowlands, and downlink images when flooding occurs. This onboard data selection can vastly improve the science return of the mission by improving the efficiency of the limited downlink. Thus, there is significant motivation for onboard autonomy.

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

1. Limited, intermittent communications to the agent. A spacecraft in low earth orbit typically has 8 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 fre-

quently. 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.
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.
5. High stakes. A typical space mission costs hundreds of millions of dollars and any failure has significant economic impact. 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.

This paper discusses our efforts to build and operate a safe autonomous space science agent. The principal contributions of this paper are as follows:

1. We describe our layered agent architecture and how that enables additional agent safety.
2. We describe our knowledge engineering and model review process designed to enforce agent safety.
3. We describe the process the agent use to detect anomalies and how it responds to these situations.
4. We describe several of the anomalies that occurred during in-flight testing, the response of the agent, and what steps were taken to prevent its occurrence in the future.

This work has been done for the Autonomous Sciencecraft Experiment (ASE) [2], an autonomy software package currently in use on NASA's New Millennium Earth Observer One (EO-1) [5] spacecraft.

In this paper we address a number of issues from the workshop call.

Definition of agent safety and how to build a safe agent – we define agent safety as ensuring the health and continued operation of the spacecraft. We design our agent to have redundant means to enforce all known spacecraft operations constraints. We also utilize declarative knowledge representations, whose models are extensively reviewed and tested. We use code generation technologies to automatically generate redundant checks to improve software reliability. Additionally, our experiment is also designed to fly in a series of increasing autonomous phases, to enable characterization of performance of the agent and to build confidence.

Robust to environment (unexpected) – our agent must be robust to unexpected environmental changes. Our agent uses a classic layered architecture approach to dealing with execution uncertainties.

How to constrain agents – because of strong concerns for safety, our agent architecture is designed to enable redundancy, adjustable autonomy, and fail-safe disabling of agent capabilities. The layering of the agent enables lower levels of the agent to inhibit higher-level agent behavior. For example, the task executive systems (SCL) does not allow dangerous commands from the planner to be sent on to the flight software. The flight software bridge (FSB) can be instructed to disable any commands from the autonomy software or to shutdown components of or the entire autonomy software. The EO-1 flight software also includes a fault protection function designed to inhibit potentially hazardous commands from any source (including the autonomy software, stored command loads from the ground, or real-time commands).

The remainder of this paper is organized as follows. First we describe the ASE software architecture, with an emphasis on how it enhances safe agent construction. Next we discuss the efforts made to prevent, detect, and respond to in-flight anomalies. Finally we present several of the anomalies that have occurred to date. We describe how the software responded to these anomalous situations, and the steps taken to prevent it from occurring in the future.

2 ASE

The autonomy software on EO-1 is organized as a traditional three-layer architecture [4] (See Figure 1.). At the top layer, the Continuous Activity Scheduling Planning Execution and Replanning (CASPER) system [1, 7] is responsible for mission planning functions. Operating on the tens-of-minutes timescale, CASPER responds to events that have widespread (across orbits) effects, scheduling science activities that respect spacecraft operations and resource constraints. Activities in a CASPER schedule become inputs to the Spacecraft Command Language (SCL) execution system [6].

SCL initiates a set of scripts that issue the complete sequence of commands to the flight software. Prior to issuing each command, constraints are checked again to confirm the validity of the command as well as ensure the safety of the spacecraft. After the command is sent, SCL checks for a successful initiation and completion of the command. When a full sequence for a data collection is complete, one or more image processing algorithms are performed which may result in new requests to the planner.

2.1 Mission Planning

Responsible for long-term mission planning, the ASE planner (CASPER) accepts as inputs the science and engineering goals and ensures high-level goal-oriented behavior. These goals may be provided by either the ground operators or triggered by the onboard science algorithms. The model-based planning algorithms enables rapid response to a wide range of operations scenarios based on a deep model of spacecraft constraints, including faster recovery from spacecraft anomalies. CASPER uses repair-based techniques [1] that allow the planner to make rapid changes to the current plan to accommodate the continuously changing spacecraft state and science requests. During repair, CASPER collects a set of conflicts that represent violations of spacecraft constraints. Generic algorithms are used to select and analyze a conflict to produce a set of potential plan modifications that may resolve the conflict. Heuristics are used to select a potential modification, and the plan is updated and reevaluated for new conflicts. This process continues until no conflicts remain.

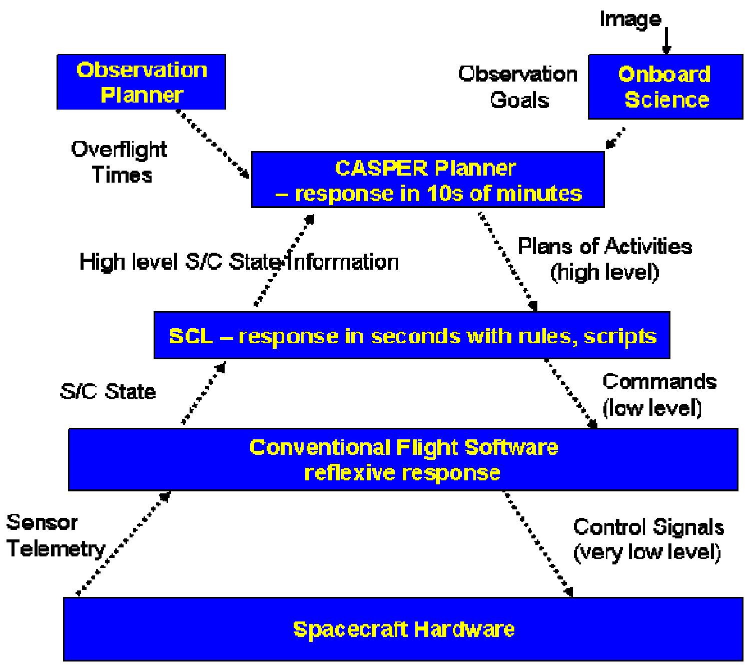
Image
Observation Planner
Observation Goals
Onboard Science
Overflight Times
CASPER Planner – response in 10s of minutes
High level S/C State Information
Plans of Activities (high level)
SCL – response in seconds with rules, scripts
S/C State
Commands (low level)
Conventional Flight Software reflexive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

Fig. 1. Autonomy Software Architecture

2.2 Robust Execution

At the middle layer, SCL is responsible for generating and executing a detailed sequence of commands that correspond to expansions of CASPER activities. SCL also implements spacecraft constraints and flight rules. Operating on the several-second timescale, SCL responds to events that have local effects, but require immediate attention and a quick resolution. SCL performs activities using scripts and rules. The scripts link together lower level commands and routines and the rules enforce additional flight constraints.

SCL issues commands to the EO-1 flight software system (FSS), the basic flight software that operates the EO-1 spacecraft. The interface from SCL to the EO-1 FSS is at the same level as ground generated command sequences. This interface is implemented by the Autonomy Flight Software Bridge (FSB), which takes a specified set of autonomy software messages and issues the corresponding FSS commands. The FSB also implements a set of FSS commands that it responds to that perform functions such as startup of the autonomy software, shutdown of the autonomy software, and other autonomy software configuration actions.

The FSS accepts low-level spacecraft commands which can be either stored command loads uploaded from the ground (e.g. ground planned sequences) or real-time commands (such as commands from the ground during an uplink pass). The autonomy software commands appear to the FSS as real-time commands. As part of its core, the FSS has a full fault and spacecraft protection functionality which is designed to:

1. Reject commands (from any source) that would endanger the spacecraft.
2. When in situations that threatens spacecraft health, execute pre-determined sequences to “safe” the spacecraft and stabilize it for ground assessment and reconfiguration.

For example, if a sequence issues commands that point the spacecraft imaging instruments at the sun, the fault protection software will abort the maneuver activity. Similarly, if a sequence issues commands that would expend power to unsafe levels, the fault protection software will shut down non-essential subsystems (such as science instruments) and orient the spacecraft to maximize solar power generation. While the intention of the fault protection is to cover all potentially hazardous scenarios, it is understood that the fault protection software is not foolproof. Thus, there is a strong desire to not command the spacecraft into any hazardous situation even if it is believed that the fault protection will protect the spacecraft.

2.3 Science Analysis

The image processing software is scheduled by CASPER and executed by SCL where the results from the science analysis software generate new observation requests presented to the CASPER system for integration in the mission plan.

This layered architecture for the autonomy SW is designed such that each lower layer is verifying the output of the higher layers. Requests from the science analysis, or from operators on the ground, are checked by the planner prior to being sent to SCL. The planner activities are checked by SCL prior to being sent on to the FSS. Finally, the FSS fault protection checks the SCL outputs as well.

3 Anomalies

As with any large software system and complex science scenarios, anomalous situations are expected to occur during operations. This section will describe how the ASE model was developed to enforce agent safety. We also discuss how the agent was developed to detect for anomalies and several of the responses encoded within the model. Finally, we describe several of the anomalies that have occurred during in-flight testing, the cause of the anomalies, how the agent responded to these situations, and modifications taken to prevent it from occurring in the future.

3.1 Prevention

With the control aspects of the autonomy software embodied in the CASPER & SCL models, our methodology for developing and validating the CASPER and SCL models is critical to our safe agent construction process. These models include constraints of the physical subsystems including: their modes of operation, the commands used to control them, the requirements of each mode and command, and the effects of commands. At higher levels of abstraction, CASPER models spacecraft activities such as science data collects and downlinks, which may correspond to a large number of commands. These activities can be decomposed into more detailed activities until a suitable level is reached for planning. CASPER also models spacecraft state and its progression over time. This includes discrete states such as instrument modes as well as resources such as memory available for data storage. CASPER uses its model to continuously generate and repair schedules, tracking the current spacecraft state and resources, the expected evolution of state and resources, and the effects on planned activities.

Table 1: Sample 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.

SCL continues to model spacecraft activities at finer levels of detail. These activities are modeled as scripts, which when executed, may execute additional scripts, ultimately resulting in commands to the EO-1 FSS. Spacecraft state is modeled as a database of records in SCL, where each record stores the current value of a sensor, resource, or sub-system mode. The SCL model also includes flight rules that monitor spacecraft state, and execute appropriate scripts in response to changes in state. SCL uses its model to generate and execute sequences that are valid and safe in the current context. While SCL has a detailed model of current spacecraft state and resources, it does not generally model future planned spacecraft state and resources.

Development and verification of the EO-1 CASPER and SCL models was a multiple step process.

1. First a target set of activities was identified. This was driven by a review of existing documents and reports. This allowed the modeler to get a high-level overview of the EO-1 spacecraft, including its physical components and mission objectives. Because EO-1 is currently in operation, mission reports were available from past science requests. These reports were helpful in identifying the activities performed when collecting and downlinking science data. For example, calibrations are performed before and after each image, and science requests typically include data collection from both the Hyperion (hyperspectral) and Advanced Land Imager (ALI) instruments.
2. Once the activities were defined, a formal EO-1 operations document [3] was reviewed to identify the constraints on the activities. For example, due to thermal constraints, the Hyperion cannot be left on longer than 19 minutes, and the ALI no longer than 60 minutes. The EO-1 operations team also provided spreadsheets that specified timing constraints between activities. Downlink activities, for example, are often specified with start times relative to two events: acquisition of signal (AOS) and loss of signal (LOS). Fault protection documents listing fault monitors (TSMs) were also consulted, using the reasoning that acceptable operations should not trigger TSMs.
3. With the model defined, CASPER was able to generate preliminary command sequences from past science requests that were representative of flight requests. These sequences were compared with the actual sequences for the same request. Significant differences between the two sequences identified potential problems with the model. For example, if two commands were sequenced in a different order, this may reveal an overlooked constraint on one or both of the commands. We were also provided with the actual downlinked telemetry that resulted from the execution of the science observation request. This telemetry is not only visually compared to the telemetry generated by ASE, but it can also be "played back" to a ground version of the ASE software to simulate the effects of executing sequences. The command sequences were aligned with the telemetry to identify the changes in spacecraft state and the exact timing of these changes. Again, any differences between the actual telemetry and the ASE telemetry revealed potential errors in the model. A consistent model was defined after several iterations of generating commands and telemetry, comparing with actual commands and telemetry, and fixing errors. These comparisons against ground generated se-

quences were reviewed by personnel from several different areas of the operations staff to ensure acceptability (e.g. overall operations, guidance, navigation and control, science operations, instrument operations).

4. Model reviews were conducted where the models are tabletop reviewed by a team of personnel with a range of operations and spacecraft background. This is to ensure that no incorrect parameters or assumptions are represented in the model.

Finally, a spacecraft safety review process was performed. By studying the description of the ASE software and the commands that ASE would execute, experts from each of the spacecraft subsystem areas (e.g., guidance, navigation and control, solid state recorder, Hyperion instrument, power) derived a list of potential hazards to spacecraft health. For each of these hazards, a set of possible safeguards was conjectured: implemented by operations procedure, implemented in CASPER, implemented in SCL, and implemented in the FSS. Every safeguard able to be implemented with reasonable effort was implemented and scheduled for testing. An analysis for two of the risks is shown below.

3.2 Detection

The EO-1 FSS has a set of Telemetry and Statistics Monitoring (TSM) tasks that monitor the state of the spacecraft. TSMs typically detect anomalies by comparing a state value with expected thresholds for that value.

The FSS also includes processes for transmitting engineering data from the spacecraft subsystems for recording and future playback. ASE tapped into this data stream so that it could automatically monitor spacecraft state and resources. The data is received at 4Hz and the relevant information is extracted and stored into the SCL database. SCL uses the latest information when making decisions about command execution. Spacecraft state and resources are checked:

- Prior to executing the command to verify command prerequisites are met.
- After executing the command to verify the receipt of the command.
- After an elapsed time period when the effects of the command are expected.

The SCL model also contains a set of rules that continuously monitor the state of the spacecraft and relays any change to the database to CASPER. CASPER compares the new data with its predicted values and makes any necessary updates to the predictions. Effects of these updates to the current schedule are monitored and conflicts detected. If a set of conflicts are detected, CASPER will begin modifying the plan to find conflict-free schedule.

During ground contacts, mission operators can monitor the EO-1 spacecraft telemetry in real-time, as well as playback recorded telemetry and messages. Limits are set on the various data points to alarm operators when values fall out of their expected ranges. To manually monitor ASE, we developed a set of telemetry points for each of the ASE modules. This is typically high-priority heath and status data that is continuously saved to the onboard recorder and downlinked during ground contacts. The real-time engineering data for EO-1 is monitored with the ASIST ground software tool developed at GSFC.

The FSB, which acts as a gateway, has several telemetry points to verify that we have enabled or disabled the flow of spacecraft commands and telemetry. It also has command counters for those issued to the ASE software. SCL provides telemetry on its state including counters for the number of scripts executed. CASPER provides statistics on the planning algorithm including the types of conflicts that it addresses and what changes it makes to the plan when repairing the conflicts. It also generates telemetry that identifies any differences it finds between the actual spacecraft state and the state it expects during the execution of the plan.

The telemetry points for each module is useful in providing a high level view of how the software is behaving, but debugging anomalies from this would be difficult. Therefore, each software module also saves more detailed data to log files stored on a RAM disk. As they are needed, these log files are downlinked either to debug new issues or to further validate the success of the test.

3.3 Response

Anomaly detection may trigger a response from any one of the software modules, or from the operations team. At the highest level, CASPER can respond to some anomalies by replanning future activities. For example, CASPER may delete low priority requests to alleviate unexpected over-utilization of resources. At the middle layer, SCL can respond to small anomalies with robust execution. In most cases, it can delay the execution of a command if the spacecraft has not reached the state required by the command. Because of interactions within a sequence, however, a command cannot be delayed indefinitely and may eventually fail. If a critical command fails, SCL may respond with additional commands in attempt to recover from the failure. This is typically a retry of the commands or set of commands that has failed. If the command remains failed, the effects of the command do not propagate to the SCL database, which may trigger CASPER to replan. Finally, at the lowest level, the EO-1 FSS fault protection is used to detect potentially hazardous spacecraft states and trigger commands to transition the spacecraft out of those states. For example, the Hyperion temperature increases while the instrument is in-use. The FSS fault protection monitors this temperature and initiates shut-down commands when the maximum temperature is exceeded.

4 Case Study

In this section, we describe several of the in-flight anomalies that have occurred, including the responses taken by the agent, and the changes performed to prevent future anomalies from occurring.

4.1 Anomaly Types

The anomalies that have occurred onboard can be classified into the following types: modeling, software, operator, hardware.

Modeling – This is the most common type of error, caused by an incorrect model of the spacecraft within CASPER and SCL. Many of these errors were not detected during testing and validation because the EO-1 mission did not have a high-fidelity testbed, requiring the development of simulators that made several incorrect assumptions of the spacecraft behavior.

Software – These are your standard software implementation errors that occur with any large project. The design of the agent needed to be robust to errors that occur between the CASPER, SCL, FSW, and science modules.

Operator – Commands are regularly issued from mission operators during ground contacts. These commands may modify the state of the spacecraft, so the agent will need to be robust to these situations.

Hardware – The Livingston 2 software component (described in section 5) was designed to detect and diagnose this type of error. However, because hardware errors are rare on spacecraft systems, we chose not to focus on detecting these.

The remainder of this section will describe four anomalies in detail, mainly of the modeling, software, and operator type. We will describe how they were caused, how the agent responded to the anomaly, and what work was done to prevent it from occurring in the future.

4.2 Onboard Anomalies

April 1, 2004

During this early stage of the project, we were testing a single response scenario where the onboard science module would analyze an image, and issue requests to the onboard planner for more images of that target. The scenario went as follows:

- Image Prudhoe Bay, Alaska
- Playback data from the solid state recorder into RAM for image processing (bandstripping)
- Run image classifier to identify snow, water, ice, and land. This was scheduled to run immediately after the bandstripping process.
- Run image classifier to identify for cloud coverage. This was scheduled to run 60 minutes after the bandstripping process.
- If the classification of the scene was above a threshold defined by mission scientists, request additional images of Prudhoe Bay.

Several of the constraints modeled within CASPER and SCL were:

- The bandstripping process cannot begin unless the target Prudhoe Bay was successfully imaged

- The image classifiers cannot begin unless the bandstripping process was successful

During the first ground contact following this scenario, mission operators noticed several warnings from SCL telemetry and that the onboard science module did not perform any image processing. After collecting log files from SCL and CASPER, and replaying back telemetry collected during the test, it was determined that SCL had failed the bandstripping script because of a time out during the verification of the command completion. In actuality, this verification failure was not a result of bandstripping failing, but of a bug within the flight software time code. It is still interesting, however, to examine the response of the agent.

The failure of the bandstripping script resulted in a lack of change to a SCL database record. This record is continuously monitored by CASPER and a conflict with the scheduled image classifier algorithm was recognized. However, because the first image classifier algorithm was scheduled immediately after bandstripping, CASPER had already committed to executing the classifier activity. When making this type of commitment, CASPER locks the activity (preventing any re-scheduling) and sends the execution request to SCL. The command was received by SCL, but failed the pre-requisite check, blocking the command from being sent to the science module.

The second image classifier was scheduled 60 minutes after the end of bandstripping, and thus CASPER was able to modify the plan to resolve the conflict by removing it from the schedule.

This anomaly demonstrated how the layered architecture ensured the safety of the agent. CASPER was not responsive enough to prevent the first image classifier from being issued to SCL, but the SCL pre-requisite check failed and thus the command was not issued. However in the second case, CASPER was able to respond to this failure by removing the subsequent image processing activity from the schedule.

A modification of the agent architecture to prevent these false-positive anomalies from occurring would be to have redundant checks in the completion of the commands. In this example, a single SCL database item indicated that bandstripping had failed, when in fact, it had succeeded. The model could have been updated to check multiple database records for the status of the bandstripping, instead of relying on solely on a single data point to verify completion of the command.

July 15, 2004

This anomaly demonstrates how SCL was able to respond to a verification failure of command sequence. During this test, the anomaly occurred during normal operations for an X-Band ground contact. The scenario was:

- Using the X-Band transmitter, downlink all images from the solid state recorder
- Clear all images from the solid state recorder

Several of the constraints modeled were:

- The correct voltage/current level of transceiver must be met prior to operating X-Band activities.
- The downlink must complete successfully prior to clearing all the images from the solid state recorder.

During the ground contact, mission operators noticed several warnings from SCL and also that EO-1 had not begun the X-Band downlink of images collected. The operators manually initiated the X-Band contact and completed dumping the data. After analyzing log files, it was determined that a prerequisite failure in the SCL model for the current/voltage of the transceiver prevented the contact from being initiated. As a result of the X-Band failure, SCL also rejected the command to clear all the images from the solid state recorder.

This was actually an error within the SCL model. An early version of the model included a constraint that the transceiver cannot be powered on unless the current/voltage was at the correct level. However, the threshold values for the current/voltage in reality are not valid until the transceiver is powered on.

Unfortunately, this modeling error slipped through our testing and validation process because of the lack of a high fidelity testbed. The EO-1 testbed did not have a transceiver for testing and therefore, the current/voltage values were static (at the "on" levels) in the simulation. Without valid values on the current/voltage prior to powering on the X-Band transceiver, our resolution to this problem was to simply remove the current/voltage constraint from the SCL model.

January 31, 2005

This anomaly describes CASPER's response to an unexpected change in the state of the spacecraft. During one of the scheduled ground contacts, the agent did not initiate the command sequence as requested from mission planners. An anomaly had occurred that removed the contact sequence from the mission plan. After analysis of collected telemetry, the cause of the anomaly was due to human intervention with the spacecraft several hours prior. An unscheduled contact had been initiated by mission planners, which was performed externally from the onboard planner. The unscheduled contact required mission operators to perform a blind acquisition of EO-1 and manually power on the transceiver, changing the state of the onboard transceiver to "on". At the end of this contact, the operators manually powered down the transceiver.

The change to the transceiver state resulted in an update to the SCL database, which propagated to the CASPER schedule and created a conflict with the next ground contact activity. The conflict was with a constraint in the CASPER model that only allowed the transceiver state to transition from "on" to "off" or from "off" to "on". When the update to the transceiver state was received, it set the current state to the transceiver to "on". This created a conflict with the next scheduled contact that had planned to turn the transceiver on when the state was already "on". To resolve this conflict, CASPER removed the next contact from the plan. Once the mission operator powered down the transceiver at the end of the unscheduled contact, subsequent contacts were conflict free, but the deleted contact was not rescheduled due to the risk of inserting the goal too close to its scheduled time.

To prevent this anomaly for future operations, we simply removed the transition constraints from the CASPER model of the transceiver. While not ideal, it was determined that this presented no risk to the spacecraft, and allowed the ASE software to support real-time contact requests from mission planners without affecting the remainder of the schedule.

In this anomaly, although the update to the state of the transceiver was short-lived as it was eventually powered off by mission operators, its affect on the planner propagated to the next scheduled contact, resulting in the contact being removed from the schedule. One possible solution to prevent this from occurring in the future is to delay resolving conflict until necessary. Some initial work has been started on CASPER to support time-sensitive repair heuristics, but is still experimental and was not deployed on EO-1.

February 26, 2005

This anomaly occurred during a normal data collect of an area in Peru. During the first contact following this image, SCL telemetry indicated 1 warning from execution. After analysis of log files, this was determined to be caused by a command verification failure when issuing the command to close the ALI cover. The response of SCL was to reissue the command to close the covers, ensuring that the covers would be closed at the end of the collect.

Further investigation into the problem showed that the cover did indeed close after the first command. However, due to a spike in CPU consumption during that time, SCL was not able to process the change in the database record indicating that the cover was closed. While SCL has the highest priority among the ASE tasks, it is not higher than any of the FSS tasks. We are still investigating the cause for the increase in CPU load.

In this situation, the actions of the agent were correct in preserving the safety of the spacecraft. However, a change to the model can be used to ensure this anomaly does not occur in the future. Again, similar to the first anomaly described in this paper, redundant checks to multiple SCL database items can be used to determine the true state of the covers. From example, the EO-1 Hyperion instrument covers have two data-points representing the state of the cover. One data- point indicates if the cover is either open or closed, while the other is a continuous value, representing how far the cover has been opened. A check that reasons using both of these data-points would be less prone to false-positives.

5 Livingston 2

More recently (Fall 2004), in collaboration with Ames Research Center, we have begun flying the Livingstone 2 (L2) [12] diagnosis system. Both L2 and CASPER use models of the spacecraft separate from the reasoning engine: the models are tailored for a particular application without the need to change the software, allowing reuse of the advanced reasoning software across applications. The diagnostic capability of an on-board agent can then use the models to monitor the health of the spacecraft and detect faults. Early development of the L2 model currently does not support responding to anomalous situations, only detection of them.

However, during the times of the described anomalies, L2 was not operational. Also its current model only supports monitoring the operations of the spacecraft and not the CASPER or SCL software. Therefore, anomalous situations within CASPER or SCL would not be detected by L2.

6 Related Work

In 1999, the Remote Agent experiment (RAX) [10] executed for a several days onboard the NASA Deep Space One mission. RAX is an example of a classic three-tiered architecture [4], 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 [11] is a European Space Agency (ESA) mission demonstrates onboard autonomy and launched in 2001. However, ASE has more of a focus on model-based autonomy than PROBA.

The Three Corner Sat (3CS) University Nanosat mission used CASPER onboard planning software integrated with the SCL ground and flight execution software [13]. The 3CS mission was launched in December 2004 but the spacecraft were lost due to a deployment failure. The 3CS autonomy software includes onboard science data validation, replanning, robust execution, and multiple model-based anomaly detection. The 3CS mission is considerably less complex than EO-1 but still represents an important step in the integration and flight of onboard autonomy software.

7 Conclusions

This paper has described the design of a safe agent for the Autonomous Sciencecraft Experiment along with several of the anomalies and the software's responses that have occurred during in-flight testing. First, we described the salient challenges in developing a robust, safe, spacecraft control agent. Second, we described how we used a layered architecture to enhance redundant checks for agent safety. Third, we described our model development, validation, and review. Fourth, we described how the agent responds and detects anomalous situations. Finally, we described several case studies of anomalies that have occurred in-flight and the response taken by the agent to maintain the safety of the spacecraft.

8 Acknowledgements

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. S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, Breckenridge, CO, April 2000. (see also http://casper.jpl.nasa.gov)
2. S. Chien et al., "The Earth Observing One Autonomous Science Agent", *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference*, New York City, NY, July 2004.
3. S. Chien et al, EO 1 Autonomous Sciencecraft Experiment Safety Analysis Document, 2003.
4. E. Gat, Three layer architectures, in Mobile Robots and Artificial Intelligence, (Kortenkamp, Bonasso, and Murphy eds.), Menlo Park, CA: AAAI Press, pp. 195-210.
5. Goddard Space Flight Center, EO-1 Mission page: http://eo1.gsfc.nasa.gov
6. Interface and Control Systems, SCL Home Page, http://www.interfacecontrol.com
7. G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," *International Symposium on Artificial Intelligence Robotics and Automation in Space*, Noordwijk, The Netherlands, June 1999.
8. G. Rabideau, S. Chien, R. Sherwood, D. Tran, B. Cichy, D. Mandl, S. Frye, S. Shulman, R. Bote, J. Szwaczkowski, D. Boyer, J. Van Gaasbeck, "Mission Operations with Autonomy: A preliminary report for Earth Observing-1", *International Workshop on Planning and Scheduling for Space*". Darmstadt, Germany, June 2004
9. D. Tran, S. Chien, G. Rabideau, B. Cichy, "Flight Software Issues in Onboard Automated Planning: Lessons Learned on EO-1", *International Workshop on Planning and Scheduling for Space,* Darmstadt, Germany. June 2004
10. NASA Ames, Remote Agent Experiment Home Page, http://ic.arc.nasa.gov/projects/remote-agent/. See also Remote Agent: To Boldly Go Where No AI System Has Gone Before. Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian Williams. *Artificial Intelligence* 103(1-2):5-48, August 1998
11. The PROBA Onboard Autonomy Platform, http://www.estec.esa.nl/proba
12. J. Kurien and P. Nayak. "Back to the future for consistency-based trajectory tracking." *In Proceedings of the 7th National Conference on Artificial Intelligence* (AAAI'2000), 2000.
13. S. Chien, B. Engelhardt, R. Knight, G. Rabideau, R. Sherwood, E. Hansen, A. Ortiviz, C. Wilklow, S. Wichman , "Onboard Autonomy on the Three Corner Sat Mission," *Proc i-SAIRAS* 2001, Montreal, Canada, June 2001.