The Autonomous Sciencecraft Embedded Systems Architecture

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

Jet Propulsion Laboratory
California Institute of
Technology

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

Goddard Space Flight Center

Darrell Boyer

Interface & Control Systems
Sandra Hayden[5], Adam Sweet[5],
Scott Christa[6]

NASA Ames Research Center

Abstract *- An Autonomous Science Agent has been flying onboard the Earth Observing One Spacecraft since 2003. This software enables the spacecraft to autonomously detect and responds to science events occurring on the Earth such as volcanoes, flooding, and snow melt.*

This agent includes artificial intelligence software systems that perform science data analysis, deliberative planning, and run-time robust execution. This software is in routine use to fly the EO-1 mission. In this paper we discuss the architecture used to integrate these systems and lessons learned from its multi-year flight on EO-1.

Keywords: Autonomy, Agents, Three Layer Architectures

1 Introduction

The Autonomous Sciencecraft Experiment (ASE) [15] is currently flying autonomous agent software on the Earth Observing One (EO-1) spacecraft [6]. This software uses several integrated autonomy technologies to enable autonomous science. The ASE agent software cis comprised of multiple software systems:

- 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) [7] package to enable event-driven processing and low-level autonomy
- The Continuous Activity Scheduling Planning Execution and Replanning (CASPER) [2] software that replans activities, including downlink, based on science observations in the previous orbit cycles
- The Livingstone 2 Model-based Diagnosis System [13] which tracks spacecraft state and detects anomalies in operations.

The onboard science algorithms analyze the images to extract static features and detect changes relative to previous observations. The software analyzes EO-1 Hyperion data to automatically identify regions of interest including land, ice, snow, water, and thermally hot areas. Repeat imagery using these algorithms can detect regions of change (such as flooding and ice melt) as well as regions of activity (such as lava flows). Using these algorithms onboard enables retargeting and search, e.g., retargeting the instrument on a subsequent orbit cycle to identify and capture the full extent of a flood. On future interplanetary space missions, onboard science analysis will enable capture of short-lived science phenomena. These can be captured at the finest time-scales without overwhelming onboard memory or downlink capacities by varying the data collection rate on the fly. Examples include: eruption of volcanoes on Io, formation of jets on comets, and phase transitions in ring systems. Generation of derived science products (e.g., boundary descriptions, catalogs) and change-based triggering will also reduce data volumes to a manageable level for extended duration missions that study long-term phenomena such as atmospheric changes at Jupiter and flexing and cracking of the ice crust and resurfacing on Europa.

The onboard planner (CASPER) generates mission operations plans from goals provided by the onboard science analysis module. The model-based planning algorithms enables rapid response to a wide range of operations scenarios based on a deep model of spacecraft constraints, including faster recovery from spacecraft anomalies. The onboard planner accepts as inputs the science and engineering goals and ensures high-level goal-oriented behavior.

The robust execution system (SCL) accepts the CASPER-derived plan as an input and expands the plan into low-level commands. SCL monitors the execution of the plan and has the flexibility and knowledge to perform event-driven commanding to enable local improvements in execution as well as local responses to anomalies.

Contact Author: Steve Chien – chien@aig.jpl.nasa.gov
1 – Mitretek, 2 – Microtel LLC, 3 – The Hammers Company, 4 - Honeywell

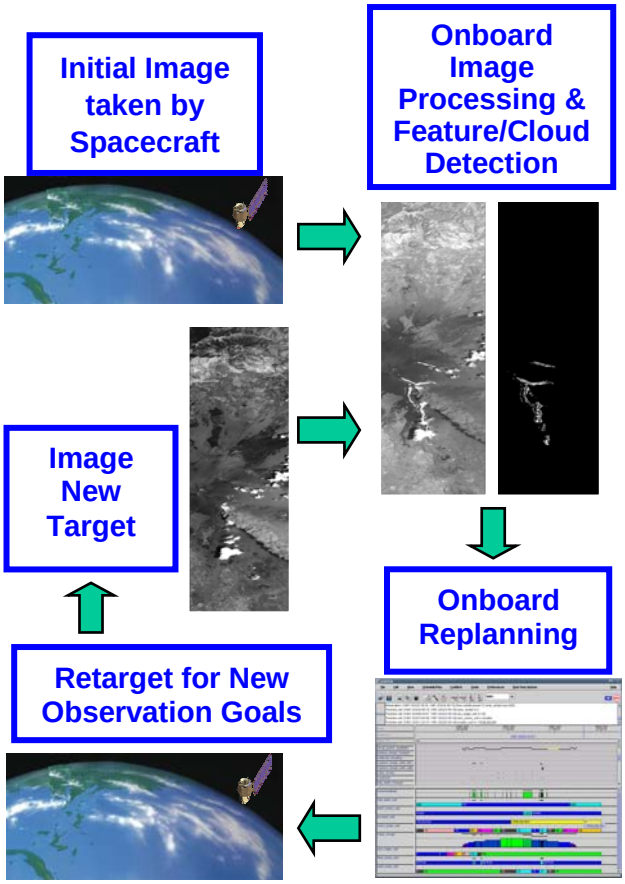

Figure 1. Autonomous Science Scenario

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

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

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

1. Limited, intermittent communications to the agent. A typical spacecraft in low earth orbit (such as EO-1) has 8 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. An average spacecraft CPUs offer 25 MIPS and 128 MB RAM – far less than a typical personal computer. Our CPU allocation for ASE on EO-1 is 4 MIPS and 128MB RAM.
5. High stakes. A typical space mission costs hundreds of millions of dollars, any failure has significant economic impact. The total EO-1 Mission cost is over $100 million dollars. Over financial cost, many launch and/or mission opportunities are limited by planetary geometries. In these cases, if a space mission is lost it may be years before another similar mission can be launched. Additionally, a space mission can take years to plan, construct the spacecraft, and reach their targets. This delay can be catastrophic.

In the remainder of this paper we describe the ASE software architecture, components, and lessons learned regarding its architecture.

2 The EO-1 Mission

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

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

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

3 The EO-1 Science Agent

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

This layered architecture was chosen for two principal reasons:

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

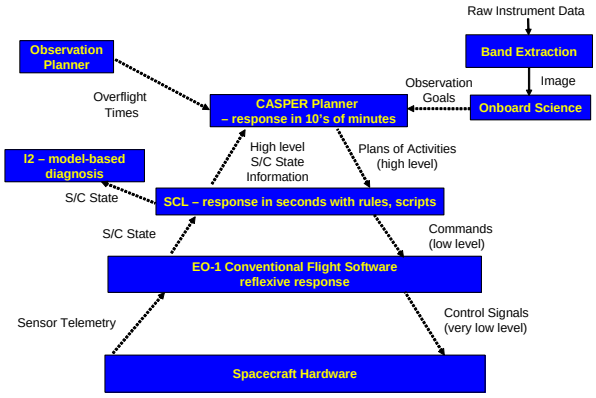

Figure 2. Autonomy Software Architecture

Each of the software modules operates at a separate priority level within the VxWorks real-time operating system onboard EO-1. The batch processes (Science) have the lowest priority, with CASPER, L2, and SCL with increasing priority.

This agent architecture is designed to scale to multiple agents. In one such scaling each agent would be a separate spacecraft, with the entire ASE software running and each agent coordinating by negotiating goals at the planner level [18]. In other applications, such as for tightly coupled tasks such as formation flying, the agents may need to communicate at the execution level [19].

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

3.1 Onboard Science Analysis

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

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

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

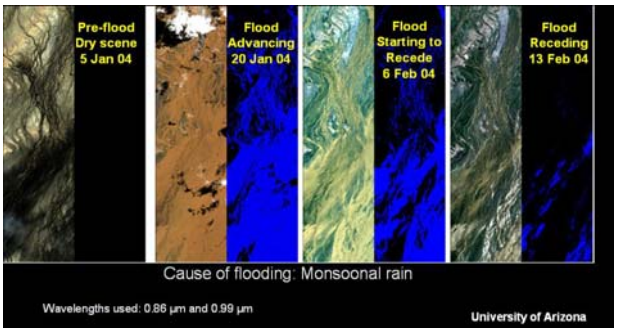

Figure 3. Flood detection timeseries imagery of Australia's Diamantina river with visual spectra at left and flood detection map at right.

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

3.2 Onboard Mission Planning

The CASPER [2] planner enables ASE to autonomously replan its future activities based on science event detections. CASPER is a deliberative, model-based AI planner which uses a local search approach [12] to develop operations plans.

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

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

3.3 Onboard Robust Execution

ASE uses the Spacecraft Command Language (SCL) [7] to provide robust execution. SCL is a software package that integrates procedural programming with a real-time, forward-chaining, rule-based system. A publish/subscribe software bus allows the distribution of notification and request messages to integrate SCL with other onboard software. This design enables both loose or tight coupling between SCL and other flight software as appropriate.

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

3.4 Model-based Diagnosis

Beginning in the Fall 2004 we have begun flying the Livingstone 2 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 intent is that a trained subsystem engineer could build these models even at the design stage. The diagnostic capability of an on-board agent can then use the models to monitor the health of the spacecraft and detect faults. The most significant advances of L2 over previous work which were demonstrated are:

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

3.5 Status

The ASE software has flown in a series of increasing tests beginning in March 2003. Full autonomous science operations were first demonstrated in January 2004. As of July 2005, ASE software had been used to successfully acquire over 2400 science images and had operated as long as three weeks continuously. Our operations have been so successful the EO-1 flight operations team now uses the ASE software for normal operations.

4. Systems and Architectural Lessons Learned on ASE

One of the most important lessons learned in the flight of ASE was that the overall architecture and systems used were extremely synergistic and key to the success of the experiment. The two components that represent mission operations and spacecraft constraint knowledge (CASPER and SCL) have very different representations. CASPER

uses an activity centered model that represents spacecraft states, resources, and timing requirements. SCL easily encodes procedures in the form of Do A then Do B then wait for condition C, then do E. These varied representations were invaluable in representing a wide range of operations situations and responses (this result is consistent with other experiences in automated planning [16 17]).

Additionally, a key aspect of ASE is spacecraft safety [8]. The ASE layered architecture enabled redundant implementation of safety constraints. Design of these constraints included a spacecraft safety review. In this process, experts from each of the spacecraft subsystem areas (e.g. guidance, navigation and control, solid state recorder, Hyperion instrument, power, …) studied the description of the ASE software and commands that the ASE SW would execute and 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. Such analysis for two risks is shown below.

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.

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

5 Related Work, and Conclusions

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 [5], 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 [1]. The 3CS mission was launched in December 2004 but the spacecraft were lost due to a deployment failure. The 3CS autonomy software includes onboard science data validation, replanning, robust execution, and multiple model-based anomaly detection. The 3CS mission is considerably less complex than EO-1 but still represents an important step in the integration and flight of onboard autonomy software.

More recent work from NASA Ames Research Center is focused on building the IDEA planning and execution architecture [9]. 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 [14]. However this mission was cancelled and the software was adapted for flight on EO-1. The principal changes from the Techsat-21 to EO-1 are that the science payload was changed from a synthetic aperture radar (SAR) to a hyperspectral imaging device (Hyperion). This change requires significant alteration to the science targets

and analysis algorithms. The basic software architecture and components (e.g. CASPER and SCL) have remained the same. This paper also reports on some of our experiences in getting the software to flight and operations. ASE has also been integrated with other satellite data (via the ground and internet) as well as with in-situ ground sensors [4].

ASE on EO-1 demonstrates an integrated autonomous mission using onboard science analysis, replanning, and robust execution. The ASE performs intelligent science data selection that will lead to a reduction in data downlink. In addition, the ASE will increase science return through autonomous retargeting. Demonstration of these capabilities onboard EO-1 will enable radically different missions with significant onboard decision-making leading to novel science opportunities. The paradigm shift toward highly autonomous spacecraft will enable future NASA missions to achieve significantly greater science returns with reduced risk and reduced operations cost.

References

1. 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.
2. S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Breckenridge, CO, April 2000. (also casper.jpl.nasa.gov)
3. A.G. Davies, R. Greeley, K. Williams, V. Baker, J. Dohm, M. Burl, E. Mjolsness, R. Castano, T. Stough, J. Roden, S. Chien, R. Sherwood, "ASC Science Report," August 2001. (see ase.jpl.nasa.gov)
4. S. Chien, B. Cichy, A. Davies, D. Tran, G. Rabideau, R. Castano, R. Sherwood, D. Mandel, S. Frye, S. Shulman, J. Jones, S. Grosvenor, An Autonomous Earth-Observing Sensorweb, IEEE Intelligent Systems. May/June 2005.
5. E. Gat et al., Three-Layer Architectures. in D. Kortenkamp et al. eds. AI and Mobile Robots. AAAI Press, 1998.
6. Goddard Space Flight Center, EO-1 Mission page: http://EO-1.gsfc.nasa.gov
7. Interface and Control Systems, SCL Home Page, www.interfacecontrol.com
8. B. Cichy, S. Chien, S. Schaffer, D. Tran, G. Rabideau, R. Sherwood, “Validating the Autonomous EO-1 Science Agent,” International Workshop on Planning and Scheduling for Space (IWPSS 2004). Darmstadt, Germany. June 2004
9. N. Muscettola, G. Dorais, C. Fry, R. Levinson, and C. Plaunt, “IDEA: Planning at the Core of Autonomous Reactive Agents,” Proceedings of the Workshops at the AIPS-2002 Conference, Tolouse, France, April 2002.
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. G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," Intl Symp Artificial Int Robotics & Automation in Space, Noordwijk, The Netherlands, June 1999.
13. J. Kurien and P. Nayak. “Back to the future for consistency-based trajectory tracking.” Proc 7th Natl Conf Artificial Intelligence (AAAI'2000), 2000.
14. S. Chien, et al., “The Techsat-21 Autonomous Space Science Agent,” International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2002). Bologna, Italy. July 2002
15. S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, D. Mandl, S. Frye, B. Trout, S. Shulman, D. Boyer, “Using Autonomy Flight Software to Improve Science Return on Earth Observing One,” Journal of Aerospace Computing, Information, and Communication. April 2005
16. . T. Estlin, S. Chien, and X. Wang, ``Hierarchical Task Network and Operator-Based Planning: Two Complementary Approaches to Real-World Planning," *Journal of Experimental and Theoretical Artificial Intelligence*, 13:379-395, 2001.
17. David E. Wilkins and Marie desJardins “A Call for Knowledge-Based Planning” AI Magazine 22(1): Spring 2001, 99-115.
18. B. Clement, A. Barrett, “Continual Coordination through Shared Activities” 2nd International Conference on Autonomous and Multi-Agent Systems (AAMAS 2003). Melbourne, Australia. July 2003.
19. A. Barrett. "Distributing a Model-Based Executive for Robot Teams." In Proceedings of 2005 IEEE International Conference onSystems, Man, and Cybernetics. The Big Island, Hawaii. October 2005.

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.