AI Magazine Volume 28 Number 4 (2007) (© AAAI)

Autonomy in Space
Current Capabilities and Future Challenges

Ari Jónsson, Robert A. Morris, and Liam Pedersen

■ This article provides an overview of the nature and role of autonomy for space exploration, with a bias in focus towards describing the relevance of AI technologies. It explores the range of autonomous behavior that is relevant and useful in space exploration and illustrates the range of possible behaviors by presenting four case studies in space-exploration systems, each differing from the others in the degree of autonomy exemplified. Three core requirements are defined for autonomous space systems, and the architectures for integrating capabilities into an autonomous system are described. The article concludes with a discussion of the challenges that are faced currently in developing and deploying autonomy technologies for space.

Space exploration is a testament to the human desire to better understand our world and the universe that surrounds us. As NASA and other space agencies around the world formulate and deploy missions to return to the moon and explore Mars and beyond, the realization is emerging that smarter mobile systems that are themselves instruments of knowledge and understanding must be developed that can respond to the uncertain environment in which they are operating with limited human intervention. In addition, there is a continuing urge to reduce the overall cost of operating in space, and it is reasonable to believe that substantial savings can be realized by automating space vehicle operations and maintenance.

The need for automated operations in space applications is well known. Stringent communications constraints (limited communication windows, long communication latencies, and limited bandwidth), limited access and availability of operators, limited crew availability, system complexity, and many other factors preclude direct human oversight of many functions. In fact, almost all spacecraft require a level of autonomy, if only as a backup when communications with humans fail for some reason.

Until recently, the notion of autonomy has always been restricted to predefined explicit behaviors and programs. The spacecraft or system has no "understanding" of the situation or what the desired outcome is—it simply executes the script triggered by events. This restricted form of autonomy is adequate for many satellites operating in predictable environments such that activity sequences can be determined well in advance. However, it breaks down under increasing uncertainty and nondeterminism, such as when navigating on a planetary surface or investigating unpredictable and transient events.

Higher levels of autonomy and automation using artificial intelligence technologies enable a wider variety of more capable missions and enable humans to focus on those tasks for which they are better suited. Indeed, in many situations autonomy is far more than just a convenience; it is critical to the success of the mission. Deep space exploration in particular requires more autonomy, as communications with ground mission operators are sufficiently infrequent that the spacecraft must react to opportunities and hazards without immediate human oversight.

This article provides an overview of the nature and role of autonomy for space exploration, with a bias in focus towards describing the relevance of AI technologies. First, we explore the range of autonomous behavior that is relevant and useful in space exploration. Second, we illustrate the range of possible behaviors by presenting four case studies in space-exploration systems, each differing from the others in the degree of autonomy exem-

 ISSN 0738-4602

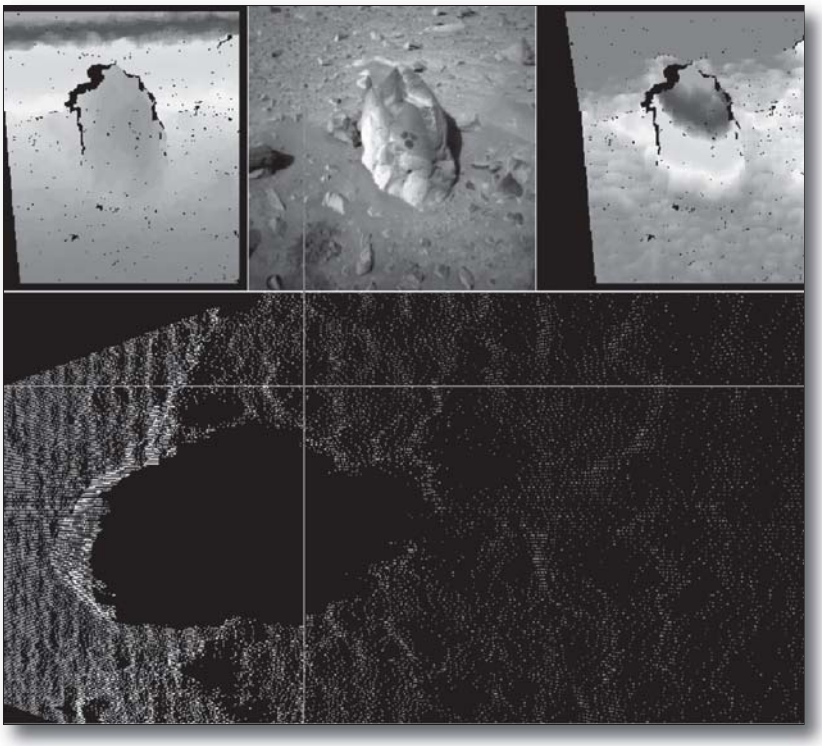

Figure 1. MER's GESTALT Obstacle Avoidance System Relied on Three-Dimensional Surface Models Computed from Stereo Images.

Detectable hazards include rocks, high slopes, and trenches (Maimone, Leger, and Biesiadecki 2007).

plified. Third, three core requirements are defined for autonomous space systems. Fourth, we describe architectures for integrating capabilities into an autonomous system. Finally, we discuss the challenges that are faced currently in developing and deploying autonomy technologies for space.

Autonomy Applications

The range of potential applications of autonomy for space operations is vast. We can distinguish broadly between three kinds of operations in space: predictable, unpredictable, and real-time response. Many flight operations such as navigation and maneuvering in space, orbiting a celestial body, observations, communication, and safekeeping activities are highly predictable and can be planned well in advance. A need for automated planning and scheduling is driven by the complexity of operations subject to tight resource constraints. In contrast, surface operations such as long- and short-range traverse, sensing, approaching an object of interest to place tools in contact with it, drilling, coring, sampling, assembly of structures, and many others are characterized by a high degree of uncertainty resulting from the interaction with the environment. For example, wheel slippage can cause a rover to drift off course or consume more energy during a traverse. Operations in these environments without autonomy to monitor progress and adjust behavior accordingly would be greatly restricted, particularly as communication delays to Earth are increased. Finally, operations such as entry, descent, and landing (EDL), automated spacecraft docking, or navigating a blimp on Titan require a real-time response from the vehicle that can preclude any interaction with mission control.

We can also distinguish between autonomy for unmanned missions from autonomy in support of human space missions. Research has been conducted in each of these application areas to determine the potential role and benefits of autonomy.

It is impossible in the space allotted here to offer a complete survey of work related to the development of autonomous space systems. Instead we focus broadly on the kinds of AI-based software technology that contribute to the development of capabilities for autonomy. We choose to categorize these under intelligent sensing, planning and execution, fault protection and health management, and distributed decision making.

Intelligent Sensing

Constrained downlink resources and long latencies impose severe limits on spacecraft operations, reducing performance and directly limiting science return. Many instruments, particularly imagers, are capable of acquiring much more data that can be downlinked or even stored on board the spacecraft for later download. For example, the THEMIS instrument on Mars Odyssey is turned on 100 percent of the time but only collects data in the range of 5–10 percent of the time due to downlink limitations (Wagstaff et al. 2005).

Interpreting rich sensor data and responding to it in a timely manner are essential for a Mars rover navigating around obstacles (figure 1) or a distant spacecraft observing transient phenomena that require immediate follow-up measurements, such as volcanic eruptions on Io, meteorological activity on Mars (figure 2), or the formation of cometary jets.

Intelligent sensing can be broadly defined as the ability to infer system state or a model (or part thereof) of the environment from environmental sensor data. Computer vision has proven to be particularly relevant for intelligent sensing because of the prevalence of cameras on most spacecraft and surface explorers (Jalobeanu, Kuehnel, and Stutz 2004). Activi-

ties of interest include tracking landmarks to infer vehicle motion or guide it to a target (Pedersen et al. 2003), detecting obstacles, and generating three-dimensional terrain models from multiple images of a landscape (note that active sensing systems such as laser range finders can do this directly and with less computation but in the past have been too massive and power hungry to deploy on rovers).

The Mars exploration rovers (MER) made significant use of computer vision both for landing and for navigation. During terminal descent, the descent image motion estimation subsystem (DIMES) (Chang, Johnson, and Matthies 2005) used images of the Martian surface from a downward-facing camera to estimate the horizontal velocity of the vehicle. This in combination with radar altimetry and the attitude measurements yielded a transverse impulse rocket subsystem (TIRS) firing solution to reduce vehicle velocity at landing to within the airbag design threshold of 24 meters per second, in spite of high winds that would have otherwise prevented landing at that site.

The MER mission has conclusively demonstrated the importance of surface mobility for accomplishing science objectives. The vehicles drove up to rocks within view to get contact measurements from targets that would otherwise have been beyond manipulator range. Over a longer time, the vehicles explored more interesting terrain and surveyed larger areas. For example, volcanic rocks of uniform type were first found at the *Spirit* landing site in Gusev Crater, without much to indicate a watery past. Pushing hard, the vehicles were able to drive several kilometers to the Columbia Hills with a greater diversity of rocks that showed clear evidence of an aqueous history.

One key to MER surface mobility was stereovision-based obstacle avoidance and motion estimation (Cheng, Maimone, and Matthies 2006), enabling it to traverse long distances, escape from craters with slippery slopes, and shorten the time required to approach science targets for in situ measurements.

The MER examples mentioned illustrate situations where autonomous, intelligent sensing is essential to mission success because the long communications delay (20–40 minutes each way) precludes direct teleoperation from Earth. However, even when signal latency is not the limiting factor, the sheer volume of sensor data makes it hard for mission controllers and science teams to identify and respond to interesting events in a timely fashion without automated assistance. This ranges from the ASE software (Chien et al. 2005a) on the *Earth Observing 1* (*EO-1*) satellite (figure 3) for autonomously detecting and responding to interesting events on Earth, to ground-based tools for visualizing rich datasets (Edwards et al. 2005), content-based image retrieval Meyer 2005), automatic crater counting, and rock frequency estimation algorithms.

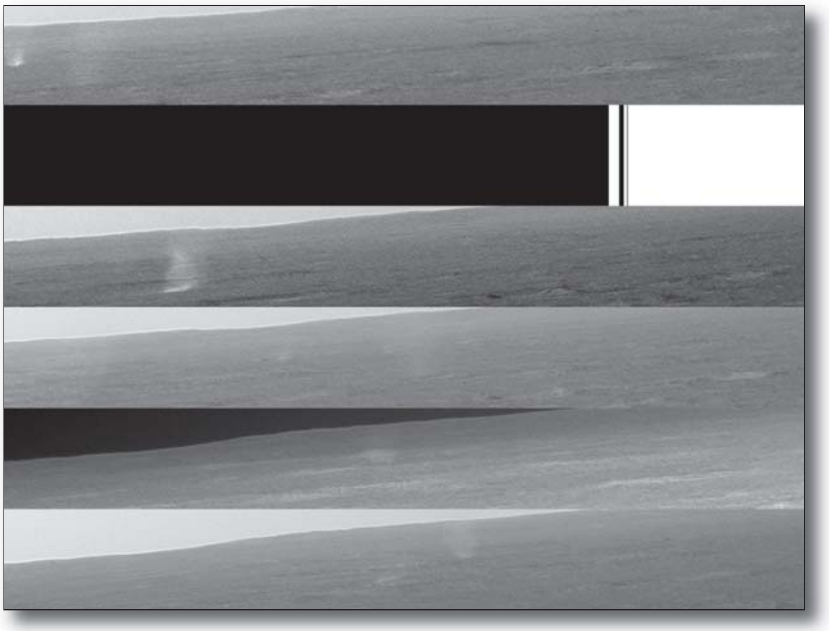

Figure 2. Dust Devils on Mars, Observed by the MER Vehicles.

The 2006 MER software update includes algorithms (Castaño et al. 2007) to detect these in low-resolution navigation images so that immediate high-resolution images can be obtained with the PanCam imagers.

Planning and Execution

Planning is the process of decomposing high-level goals into a sequence of activities that satisfy temporal, resource, and other constraints. The plans generated for space exploration typically must manage the uncertainty inherent in operating in space. This uncertainty manifests itself in how long it takes to perform an activity, and in how much resource (power, memory, CPU time) is required to perform tasks. For example, the time it takes to drive from one location to another depends on wheel slippage and path length, which in turn depend on the characteristics of the soil and terrain, and the presence of (unseen) obstacles.

Traditionally, mission planning is performed by a myriad of ground resources that generate a relatively inflexible sequence of commands that is uplinked to the spacecraft. Often, this sequence must be highly conservative to ensure that the plan can be executed. While planning and scheduling systems are being deployed to support ground operations, seldom have these systems been used in a closed-

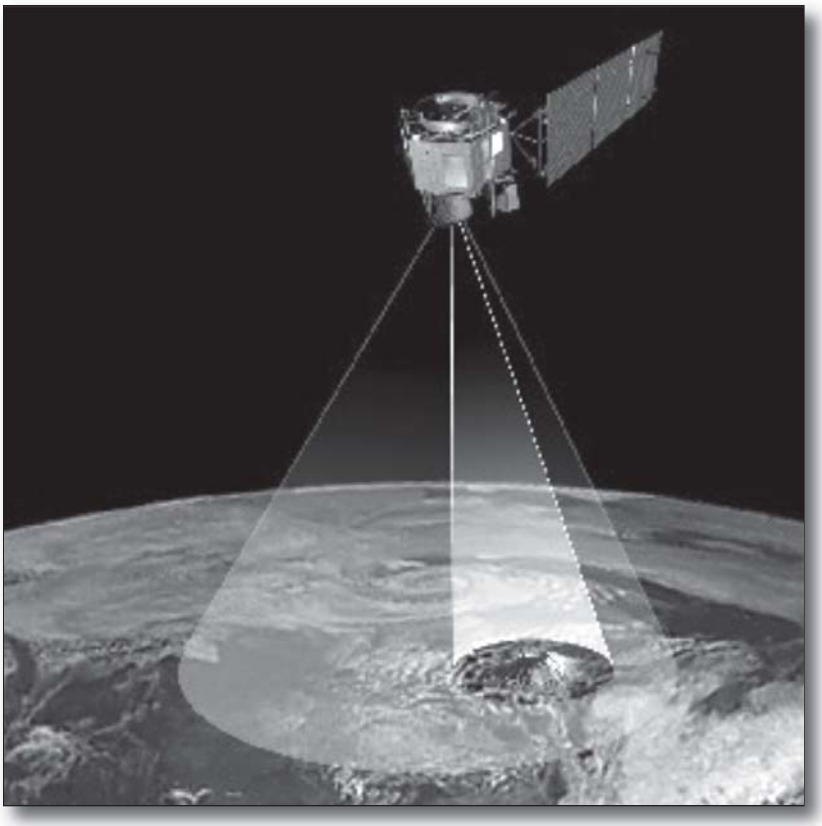

Figure 3. Earth Observing 1 Satellite EO-1.

On-board software combines machine learning and pattern recognition to identify terrestrial regions of interest or recent changes such as flooding, ice melt, or volcanic eruptions. On-board decision making allows retargeting the imager to these regions during subsequent orbital passes.

loop fashion to control an autonomous system, with the notable exception of path and motion planning (an exception is the *EO-1* ASE experiment [Chien et al. 2005b]). Furthermore, the existing technology is limited in its ability to generate a flexible sequence that allows for a variety of execution paths based upon an evaluation of the current state of the environment at the time of execution.

Path and motion planning is a special case of mission planning that must often be performed on board the vehicle because of frequent state updates that must be accommodated in real time (Stentz 1995). These include vehicle position updates, detection of new obstacles, and reaction forces encountered by vehicle wheels or manipulator device (Tompkins, Stentz, and Whittaker 2004).

An execution system is responsible for dispatching a sequence of commands while monitoring and responding to failures within the system. Given a specification of the sequence of tasks to be performed, an executive is responsible for reactively selecting the next action to take based upon the sensory inputs available at that time. The key challenge to be addressed in building execution systems is providing a guaranteed real-time response given a flexible sequence of tasks while still reasoning about the systemwide interactions and the future ramifications of an action (Williams and Nayak 1996).

Fault Protection and Health Management

The fault protection and health management decision-making capability addresses the need for detecting, diagnosing, and reacting to events occurring inside a system, given streams of observations, through the use of models about the behavior of the system. The models predict expected future states of the system, and comparisons between predictions and observations allow inferences about the system's state.

Currently, health status information for spacecraft is often obtained through limit thresholds and simple algorithms that reason locally about an individual component. For autonomous health self-management, the integration of lower-level precise models with system-level models is required to determine the global state of the device based upon an evaluation of current systemwide interactions, as well as to provide automated fault localization. This capability will limit the need for redundant sensors and will increase the overall robustness of the vehicle. Furthermore, the system will be able to perform many of these tasks within the real-time execution loop, thus permitting the system to respond quickly based on a true identification of the source of the anomaly as opposed to limited local information (Robertson, Effinger, and Williams 2006).

Traditional AI-based diagnosis systems typically use a combination of a declarative representation of the physical system and monitors to translate continuous-valued sensors into discrete-valued variables. These systems work reasonably well for diagnosing spacecraft navigating through space, in which the state of the environment does not typically change dramatically over time. By contrast, rover diagnosis depends significantly on environmental interactions and potentially noisy data. Consequently, the diagnostic system must be smart enough to distinguish between changes caused by faulty components from transient state changes caused by the environment. The intimate coupling of physical device with environment puts a strain on the ability of deliberative diagnostic systems to perform well in this domain. A sensor reading may be normal in

some contexts and indicate a fault in others; this tends to push the decision making towards the monitors, who must be sophisticated enough to interpret the environmental context producing the observation when translating the continuous valued data into discrete values. In other words, with rovers, diagnosis becomes more of an intelligent sensing problem.

An alternative approach is to have a hybrid diagnostic system, one that can reason directly with the continuous values received from sensors, as well as perform discrete mode identification (Hofbaur and Williams 2004). The system model used by a hybrid diagnostic system consists of three parts: a set of discrete modes and a transition function that describes how the system changes from one mode to another; a set of equations that, for each mode, describes the dynamics of the system in terms of the evolution of the continuous state variables; and an observation function that defines the likelihood of an observation given the mode and the values of the state variables. The diagnosis problem becomes that of identifying the mode of the system, as well as the values of all the state variables (Hofbaur and Williams 2002).

Distributed Decision Making

Within the past few years NASA has formulated mission scenarios that involve scientific assets distributed across multiple spacecraft. Examples include the terrestrial planet finder (TPF), the laser interferometry space antenna (LISA), and the TechSat21. The TPF consists of an ultraprecise formation of four collector and one combiner spacecrafts that combine the high sensitivity of space telescopes with sharply detailed pictures from an interferometer. With LISA, gravitational waves from galactic and extragalactic sources will be observed using three spacecraft flying 5 million kilometers apart in the shape of an equilateral triangle. The TechSat21 Air Force Research Laboratory (AFRL)–led microsatellite program would validate a distributed sparse aperture radar concept; understand formation flying, fine position sensing/control, and a number of microsatellite technologies; measure ionospheric small-scale structure in situ and propagation; and demonstrate lightweight microsat hardware. In addition to distributed spacecraft, NASA plans surface exploration missions, which require mixed teams of humans and robots.

Autonomous capabilities for coordination address the need for cooperation between independent autonomous agents to achieve a common goal. Effective cooperation requires resources to be shared across systems and the assignment of roles and responsibilities to minimize the coupling between agents while still ensuring coordination in the attempt to satisfy the higher-level mission goals. Distributed decision making is critical for missions such as a robotic colony on Mars, deployment of a fleet of sensing devices orbiting Earth, or within an armada of cooperating deep space probes.

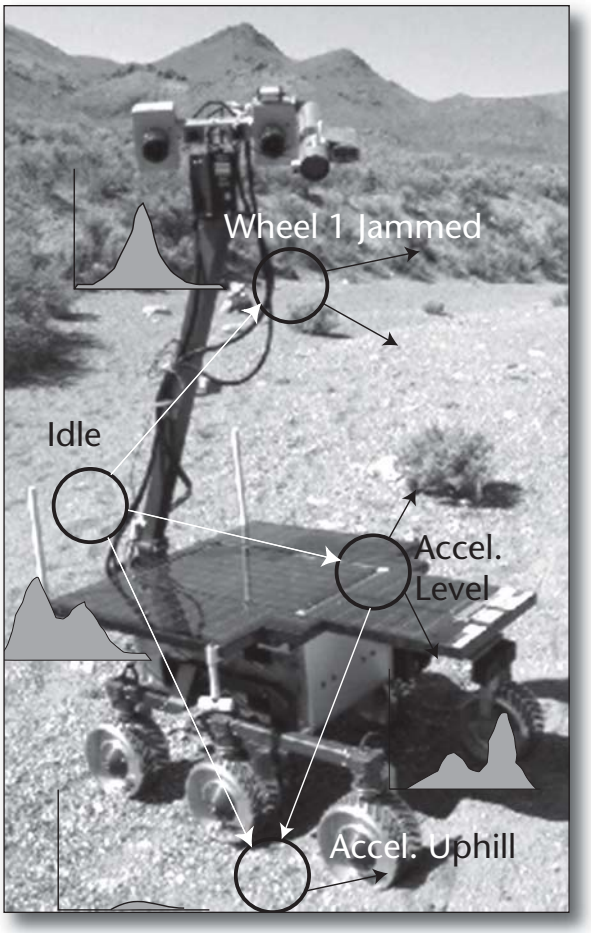

Figure 4. Reasoning for Fault Protection on the K9 Rover.

Teams of specialized robots offer the potential for accomplishing complex tasks better and more reliably than do independent, general-purpose decision makers. Among other things, teams have the capability to work in parallel, combine capabilities to achieve new competence, and monitor and repair one another. This advantage is especially important for long-term missions, where single points of failure can be disastrous. The challenge in achieving the benefits of collaborative teams of robots, however, resides in the ability to coordinate their activities effectively.

Building from research in autonomous robot architectures, multirobot coordination, plan-

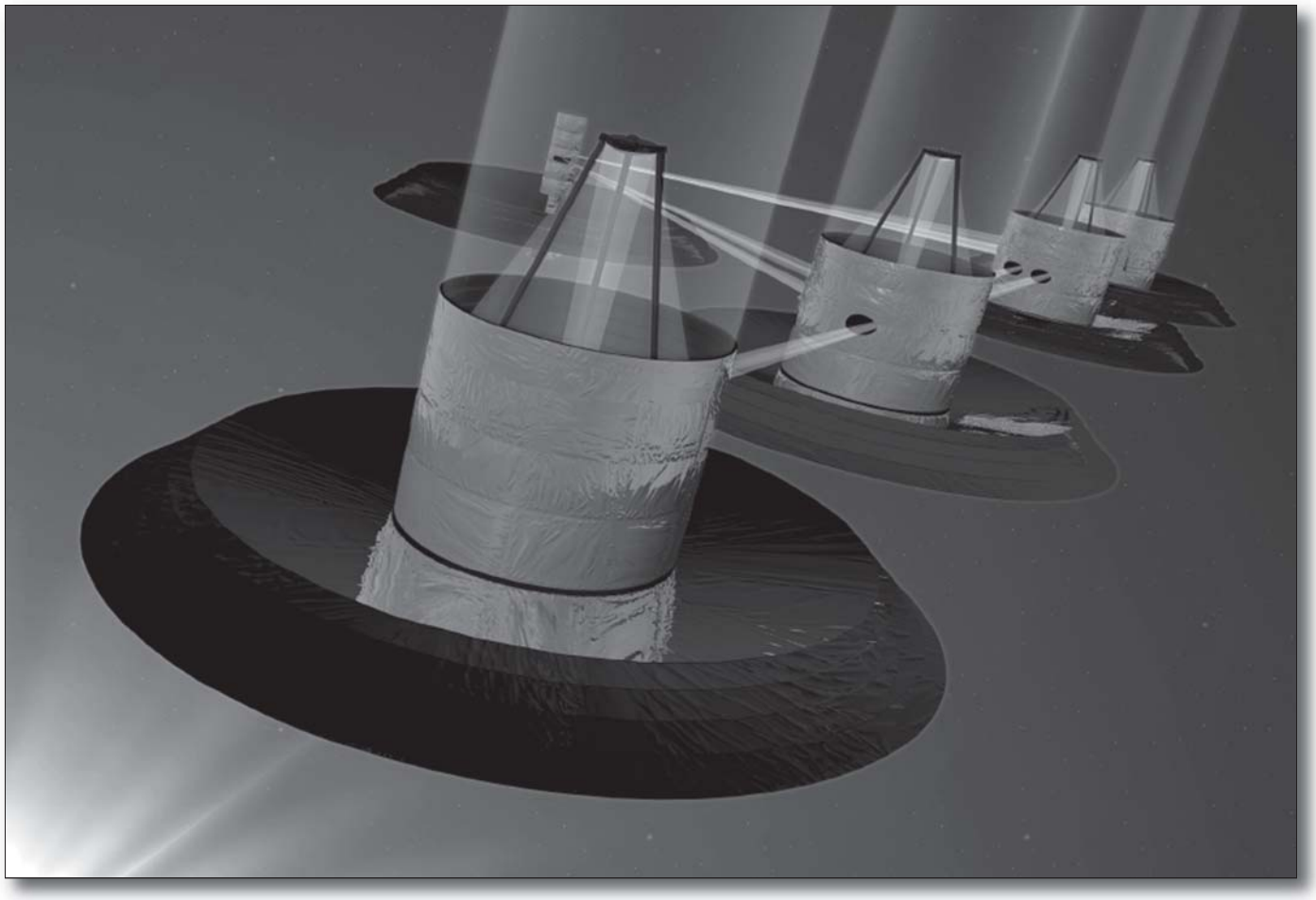

Figure 5. Terrestrial Planet Finder Observatories Will Study All Aspects of Planets Outside Our Solar System.

ning and scheduling, task execution, distributed control, probabilistic reasoning, machine learning, and monitoring and fault diagnosis, the problems of heterogeneous coordination are addressed on several levels.

First is coordination at the planning and task allocation level (Chen and Decker 2004, Clement and Barrett 2001). This involves building an overall architecture for dynamic team formation and includes distributed or centralized approaches to planning of team activities, as well as methods for agents to learn to better estimate their own capabilities, and those of others, and to better negotiate task assignments.

Second is coordination at the execution and control level, which includes developing techniques for independent, heterogeneous agents to explicitly coordinate with one another to achieve tasks. Examples include synchronized localization (Fox et al. 1999) and coordinated mobility (for example, to enable formation flying or coordinated assembly of a remote outpost).

Third is coordination at the health-management level (Tumer, Uckun, and Agogino 2005), which consists of developing techniques that enable robots to monitor one another's progress and to assist each other in diagnosis and repair.

Capabilities in each of these categories combine to create autonomous spacecraft. In the "What Makes a System Autonomous?" section, we will describe principles for organizing these capabilities into an integrated system. In the next section we provide examples of operating flight systems that differ in the degree to which these autonomous capabilities are exhibited.

Case Studies in Space Flight Systems

In this section, four space missions, past and present, are examined. For each mission, we

describe and compare the mission objectives, mission operations, spacecraft subsystems, and strategies for onboard control.

Cassini-Huygens

Cassini-Huygens is the first mission to explore Saturn's system of rings and moons from orbit. After a seven-year voyage that includes four gravity-assist maneuvers, *Cassini* entered Saturn's orbit in June 2004. A four-year mission includes more than 70 orbits around the ringed planet and its moons. Pointing its various instruments at carefully calculated scientific targets, *Cassini* collects detailed data on Saturn. The European Space Agency's *Huygens* probe dove into Titan's atmosphere in January 2005.[1] The 12 science instruments on board the *Cassini* spacecraft carry out sophisticated scientific studies of Saturn, from acquiring data in multiple regions of the electromagnetic spectrum, to studying dust particles, to characterizing Saturn's plasma environment and magnetosphere. The instruments gather data for 27 science campaigns, providing scientists with enormous amounts of information.

Mission control for *Cassini-Huygens* is coordinated in the form of a human operator, called the Ace, who monitors the ground data system and the spacecraft during the periods when the Deep Space Network (DSN) is tracking the spacecraft. Dozens of engineers and scientists communicate with the Ace by voicenet, telephone, or e-mail while checking system status or uplinking command sequences to the spacecraft. The Ace ensures that all the spacecraft's data are acquired, checked, stored, and distributed. The Ace is also responsible for correcting problems that sometimes occur with the hundreds of computers, programs, data files, and processes that enable the flight team to communicate with the *Cassini* spacecraft. Workstations display the spacecraft's health and safety, as well as the real-time status of DSN antennas and systems that link together to provide two-way communications with the spacecraft.

From a command and control perspective, *Cassini-Huygens* illustrates a fairly traditional spacecraft and mission design. Although commanding and decision making for *Cassini-Huygens* are virtually completely done by humans, a limited amount of self-commanding is conducted on board. In particular, the spacecraft contains on-board fault protection software, which is able autonomously to place the spacecraft in a safe, stable state and able to receive commands should an equipment failure occur. The software also responds automatically to faults requiring immediate action.

Figure 6. TechSat 21 Uses Clusters of Microsatellites That Operate Cooperatively to Perform the Function of a Larger, Single Satellite.

The spacecraft also conducts autonomous attitude determination and control in the attitude and articulation control subsystem (AACS), using three inertial reference units (IRUs) and a star tracker. Reaction wheel assemblies (RWAs) are one of two systems used to provide pointing control of the spacecraft in flight (with the thrusters of the propulsion module subsystem as the other). The reaction wheel assemblies contain electrically powered wheels. They are mounted along three orthogonal axes aboard the spacecraft.

Mars Exploration Rovers

The twin *Spirit* and *Opportunity* Mars exploration rovers were successfully landed on Mars in January 2004. The goal of the mission was to discover evidence of past water at Gusev Crater and Meridiani Planum. Each rover is equipped with a multifilter and stereo camera pairs and a thermal emission spectrometer and in situ measurements (with a five degree of freedom arm for deploying a rock abrasion tool). The rovers also consist of a microscopic imager, an alpha particle X-ray spectrometer, and a Mossbauer spectrometer (Biesiadecki, Leger, and Maimone 2005).

The MER vehicles are typically commanded once per Martian solar day (sol) using MAPGEN, an automated constraint-based planning system (Ai-Chang et al. 2003). A sequence of commands sent in the morning specifies the sol's activities: what data to collect, how to position the robotic arm, and where to drive. The uplinked commands are selected based on what is known—and what is unknown—about

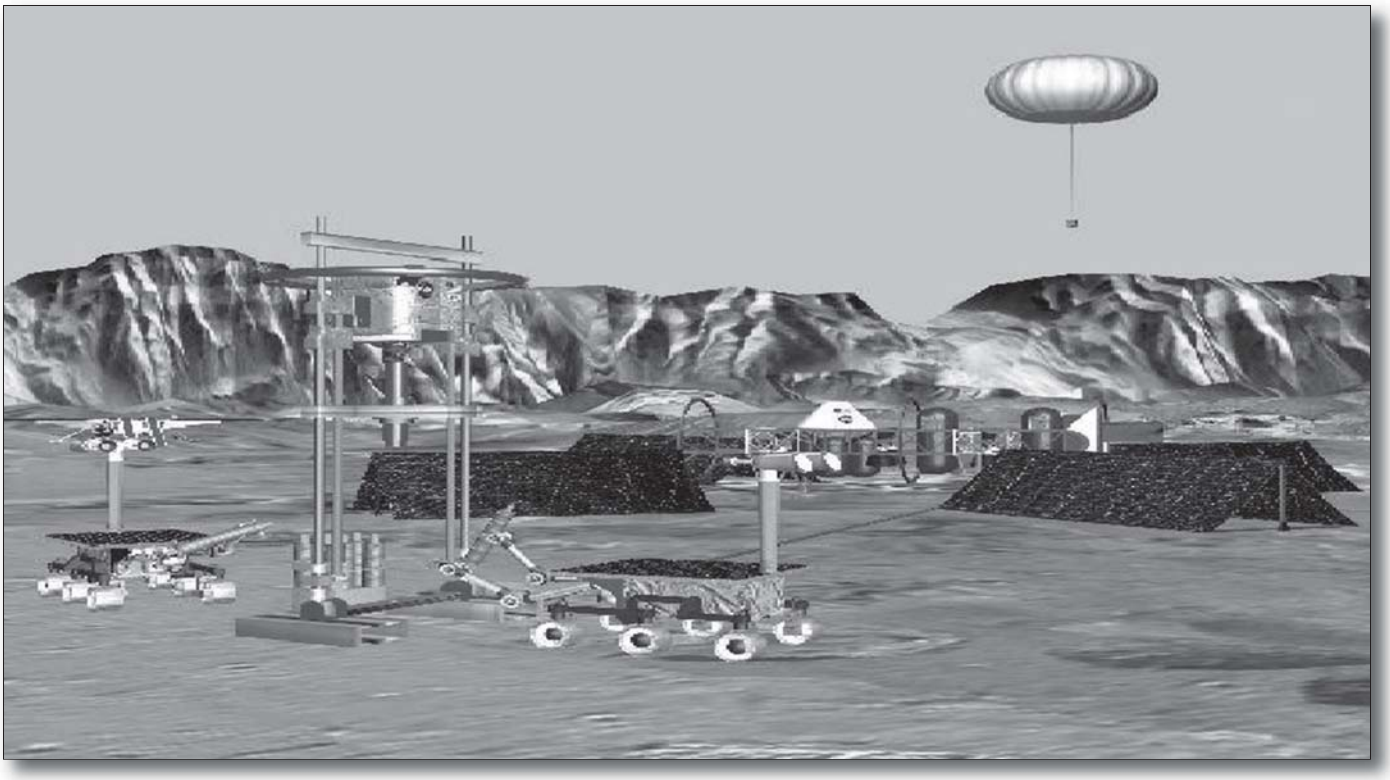

Figure 7. Artist's Conception of a Team of Rovers Performing an Assembly Task on Mars.

the terrain ahead. At the end of each sol, the rovers send back the images and data used by ground operators to plan the next sol's activities.

There are two basic driving modes on the MER vehicles. Traverses cover long distances; approaches are short-term drives to a specific target, like an interesting rock. There is also a fine positioning mode, which involves placing a robotic arm in close proximity to the desired target. The rovers have autonomous navigation capabilities. They can be commanded by specifying a waypoint. The rovers will generate a path and autonomously traverse to the waypoint. The rovers have onboard hazard detection using stereo vision pairs taken by any of the three types of stereo camera pairs found on MER vehicles to locate traverse hazards and avoid them. They have two kinds of obstacle avoidance: reactive and proactive. In reactive obstacle avoidance, real-time interrupts check for orientations that may put the rover at risk. The rovers can also do terrain analysis proactively to evaluate future risk.

In addition to autonomous navigation, the rovers can use visual odometry to gauge vehicle motion accurately. This capability has proved essential for egressing steep crater slopes where significant wheel slippage makes traditional dead-reckoning-based navigation impractical. In addition, this capability has been demonstrated to be effective in cases where precision positioning of an instrument on a target is required (Biesiadecki, Leger, and Maimone 2005).

It should be noted that recently a number of autonomy enhancements have been developed and demonstrated on the MER vehicles. Among them are the use of D* path planning to assist in hazard avoidance,[2] the use of visual target tracking for more accurate target approach,[3] and decision making for on-board science.[4]

Remote Agent

Remote agent (RA) is an AI software system for spacecraft goal-based commanding, robust execution, and fault recovery. The remote agent experiment (RAX) on *Deep Space 1* (*DS1*) demonstrated RA capabilities in May 1999. RA allowed ground operators to command the system at the goal level rather than through issuing individual commands and by the automation of flexible failure detection, isolation, and recovery (flexible in contrast, for example, to *Cassini-Hugyen*'s traditional rigid safing[1] response to all anomolies).

During RAX, RA controlled a number of *DS1* subsystems, including the camera for use in autonomous navigation, the solar electric propulsion (SEP) subsystem for trajectory

adjustment, the attitude control system for turns and attitude hold, the navigation system for determining how the actual trajectory is deviating from the reference trajectory and the SEP thrusting profile required to stay on the reference trajectory, and the power amplification and switching module (PASM), for use in demonstrating fault protection capabilities (Jónsson et al. 2000).

RA is one of the original examples of a layered autonomy architecture (more on this later). The primary components of RA are a planner/scheduler, a reactive plan execution system, a diagnosis and repair system (MIR) composed of mode identifier and mode recovery components, and a mission manager that selects goals for each planning horizon (figure 8). A RAX manager provided a messaging conduit between RA and the real-time execution flight software (Bernard et al. 1998). RA issues commands to the flight software to modify the system state and receives data from *DS1* through a system of monitors that discretize sensor values. First, the mission manager selects goals for the next commanding horizon (typically, several days in length) and sends them to the planner/scheduler, which then plans activities for the accomplishment of the goals.

RA elements relied on relatively abstract declarative models to generate plans from goal commands, synchronize the parallel execution of plan activities, and respond to faults. Models of the ion propulsion system, the attitude control system, the miniature integrated camera and spectrometer (MICAS), power, reaction control system, the data system, sensors, and the RA itself were developed (Bernard et al. 1998).

The planner/scheduler is composed of a heuristic chronological backtracking search engine managing a temporal database. The planner/scheduler continuously expands partial plans into complete plans by posting constraints derived either from spacecraft models or from planning goals. The executive is a reactive, goal-oriented control system, managing the execution of plans requested from the planner/scheduler, as well as system-level fault protection. A goal is a state of the system being controlled that must be maintained for a period of time, such as keeping a device on for a requested duration. Exec coordinates the execution of multiple goals simultaneously, by transforming goals into command procedures.

MIR performs model-based diagnosis using declarative spacecraft component models to determine whether the current state of each component (mode) is consistent with the current observed sensor values. If failures occur, MIR uses the same model to find a repair or workaround procedure that allows the current plan to continue to execute.

Because of the departure of RA from traditional flight software, the integration of RA with the *DS1* flight software offered unique challenges, among which was the fact that RA was implemented in Lisp, whereas the flight software was implemented in C or other lower-level languages. Interfaces between RA and flight code needed to be designed and implemented. In addition, memory and CPU fraction constraints required a thorough analysis and transduction (removal of unneeded components of the development environment from the final image).

RAX operated *DS1* for several days. Scenarios that were tested included executing an IPS thrust arc, acquiring optical navigation images, and responding to several simulated faults. RA successfully accomplished all of its validation objectives during the experiment.

Autonomous Science Experiment on *EO-1*

The autonomous science experiment (ASE) on NASA's *EO-1* Earth observing spacecraft demonstrates autonomous control of science activity. *Earth Observing 1* is the first satellite in NASA's new millennium program Earth observing series, launched on November 21, 2000.The core autonomy capabilities of ASE are on-board science analysis, replanning, and robust execution. A later extension to the experiment has included the incorporation of model-based diagnostic capabilities (Hayden, Sweet, and Christa 2004).

Image data acquired from the Hyperion instrument on EO-1 are analyzed on board to detect the occurrence of something with science value. Science data are downlinked only when either change is detected from previous observations, or when events of interest such as volcanic eruptions, flooding, ice breakup, and presence of cloud cover are detected. The results of analysis also provide inputs to onboard decision-making algorithms that then update the current observation plan, derived initially from goals uplinked from the ground, to capture high-value science events. This new observation plan is executed by a goal and task execution system, able to adjust the plan to succeed despite runtime anomalies and uncertainties (Chien et al. 2005a).

The autonomy software is organized into a three-layer architecture. At the highest level of abstraction, the continuous activity scheduling planning execution and replanning (CASPER) system is responsible for mission planning. CASPER uses resource and spacecraft constraint models to generate feasible schedules of science activities. Activities are sent to the spacecraft command language (SCL) execution system, which generates detailed sequence commands. At the bottom tier, the *EO-1* flight software provides low-level control of the spacecraft and provides a layer of independent fault protection. The science analysis software is scheduled by CASPER and executed by SCL in batch mode. The results from the science analysis provide new observation requests for integration into the mission plan.

The ASE software has flown in a series of tests beginning in March 2003. Fully autonomous science operations were first demonstrated in January 2004. As of June 2007, ASE software had been used to successfully acquire more than 10,000 science images and has flown for years continuously. The value added to NASA has already reached millions of dollars, and it is now the primary mission operations software for *EO-1*. The increase in science return has been documented to be over 100 times by the ASE science team.

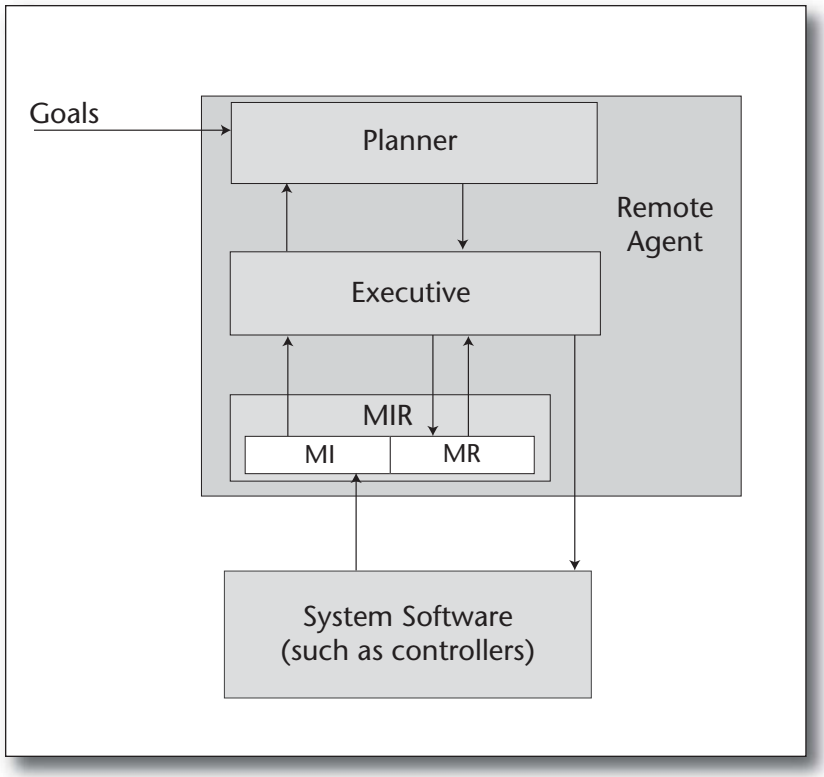

Figure 8. Remote Agent Architecture

What Makes a System Autonomous?

The preceding examples of spacecraft and methods to control them serve to illustrate the range of behaviors that could be made autonomous through the application of technologies related to intelligent sensing, planning and execution, and health management. In this section, we address the broader question of how to classify and characterize autonomous behavior.

Specifically, we propose three necessary conditions for autonomy:

> 1. Autonomy describes a range of behaviors associated with agents, systems that can sense the world around them as well as their own state, can make decisions about what to do, and can carry out their decisions through their own action.

Autonomy is associated with traditional concepts of agency, the ability of a system to sense, think, and act. Furthermore, autonomy is about control authority, the range of capabilities that a system can exhibit. The chess-playing program Deep Blue is not an autonomous system by this definition, even if it exhibits intelligence in chess playing. Its range of behaviors is limited to moves of a chess game, and it does not exhibit the necessary conditions for agency.

> 2. An autonomous system can be controlled by commanding it to achieve a set of goals; the system itself transforms the goals into sequences of actions that accomplish each goal.

A teleoperated robot can exhibit a range of goal-directed behaviors, but it is not autonomous because it is command driven rather than goal driven. It should be further noted that goals can be specified at increasing levels of abstraction. The goal "Find an interesting rock" is more abstract than the goal "Drive to location (x, y)", which is in turn more abstract than the command "Turn right 90 degrees."

> 3. An autonomous system flexibly responds to off-nominal situations by adjusting its activity sequence to attain the high-level goals and maintain system safety.

This condition refers to a system's robustness with respect to the environment within which it is acting, that is, the envelope of conditions under which it can accomplish its goals. This condition ensures that a system that is goal-directed but is susceptible to failure in accomplishing its goals due to unpredicted changes is not considered autonomous. With respect to the ability to remain safe in the presence of anomalies, for example, the *DS1* spacecraft (Jónsson et al. 2000), with its capability to "fail flexibly," is more autonomous than the *Cassini-Huygens* spacecraft.

It follows that autonomy describes a set of capabilities, not a set of technologies. For each contributing capability to autonomy, there is a range of technologies that have been proposed and developed or applied to realize the capability. Second, humans are never out of the loop when it comes to autonomous artifacts for space. Although the evil autonomous system HAL from *2001: A Space Odyssey* can be imagined, it is clearly not the intention to devise space systems that are beyond the reach of human control (although, to be fair to HAL, it had been given high priority instructions to keep the true mission secret from the crew. It reasoned that killing them was the only remaining option to accomplish this). Finally, autonomy is not an all-or-nothing characteristic of a system, but something that admits of degree. Systems can be compared and ordered with respect to the degree to which they satisfy each of the three autonomy criteria just defined. The philosophical concern about whether a rock or a thermostat is autonomous can in this way be reduced to happy hour entertainment (or, equivalently, to an academic philosophy journal paper).

Autonomous Systems Architectures

Space-exploration systems that exhibit autonomy consist of software and hardware integrations of the set of capabilities described above. These capabilities must be organized in an effective manner in order to meet the performance requirements of space exploration. An autonomy system architecture describes the structure underlying this organization, as well as a description of how the different components communicate among themselves and with the external world (Coste-Maniere and Simmons 2000).

This section discusses general principles underlying autonomy architectures for space systems in more detail. We will focus on three different facets of an autonomy architecture: the layers of software that are typically built on the underlying hardware, the sense-think-act cycle that governs the behavior of different elements, and then the reasoning techniques that are typically involved. We will then briefly touch on distributed autonomy architectures, an extension of the core architectures to support interactions among multiple spacecraft and systems.

Layered Architecture

Over the years, a consensus has been reached that architectures for autonomy should consist of a planning layer, a task sequencing layer, and a reactive layer (Nilsson, 1984; Alami et al. 1998). These are also often referred to as deliberative, executive, and functional layers. The three layers can be distinguished in terms of their abstraction from the hardware and their response-time requirements. The response-time limitations usually translate into limits on the ability to deliberate and limits on the time horizon being considered. The functional layer has the fastest response requirements, as it must keep up with the hardware, and each component typically considers only a single task or subtask at a time. The executive layer manages a set of tasks at a time, and needs only respond fast enough to keep up with task activations and terminations (Verma et al. 2005). Finally, the deliberative layer considers multiple tasks and multiple options, taking impacts far into the future into account. It need only respond fast enough that it can provide additional task sets, or plans, to the executive when needed.

But the layers do not simply represent an increase in capabilities; there are trade-offs. The functional layer has access to detailed data from the hardware and often performs complex numerical calculations on those to determine responses or provide data to the layers above. The executive layer usually has contingency handling and control capabilities that are not in the deliberative layer.

Sense, Think, Act

Each layer in an autonomy system performs a variation of a sense-think-act cycle. The overall autonomy system then has an overall sense-act-think cycle as well. Sensing involves getting data from lower layers or the hardware and mapping that to a representation that the software can use. Thinking involves considering the sensory data, information about the spacecraft, and desired objectives and then arriving at a result about what should be done. Finally, acting involves carrying out the decisions reached in the think cycle.

The reactive, or functional, layer comprises a set of elementary actions that perform specific tasks using a predetermined algorithm or perform predetermined calculations to provide results to other components or layers. The functional layer is thus a real-time distributed system, usually implemented in a modular fashion, interacting with an environment through raw data sensors. The thinking is limited to direct deterministic calculations, based on the sensor input, and the acting is done with direct commanding of underlying hardware. For example, a functional layer pressure control action handles pump and valve control to maintain a certain pressure in a fuel system. It reads pressure sensors and calculates how trends should be mapped to valve and pump control. A functional layer computation, which typically only involves sensing, may monitor raw pressure indications and provide an indication of over or under pressure to higher-level layers.

The executive layer manages a set of tasks. Its primary responsibilities are to monitor the systems and the progress of tasks, through its sensing abilities, determine what tasks need to be activated, interrupted, resumed, or terminated; and then carry out that task control. Acting as a layer between the reactive and deliberate layers, the executive is often perceived as being very simple. But most deliberative layers provide only abstract decisions and plans, leaving the executive layer to fill in those plans and to respond when the decisions or plans do not lead to the expected results. For example, when executing a plan involving a fuel system, the executive will monitor the plan and the system, determine when to initiate an action such as maintaining a given pressure, and then send those instructions to the functional layer. But it will then continue to monitor the execution, and respond if unexpected issues such as over- or underpressure arise.

Finally, the deliberative layer is responsible for making the high-level decisions and thus is often the focus of AI research in these areas. Like other layers, the deliberative layer has its own sense-think-act cycle. A deliberative layer comes in many forms, and the wide range of capabilities and techniques that have been developed is much too large to cover here. Therefore, we will focus on the commonly used combination of state identification and decision making as the building blocks for a deliberative layer. State identification is often referred to as state estimation, diagnosis, mode identification, or even systems health management. The decision making invariably involves some sort of planning, but this capability can vary greatly, and in some cases the decision making is done by a collection of planners. These planners can differ in terms of reaction times, as some are highly reactive while others consider very long-term goals, as well as in the level of detail they consider, whether they consider contingencies, and so forth.

State identification modules read

sensor data, which can be raw output from hardware, calculated results from functional layer modules, or even summaries from the executive layer. From this information, they determine the most likely state of the system. For example, when diagnosing the fuel system, a model-based diagnosis system will map sense values to variables in a model and then reason to derive what states, for example, pump failures, leaks, and so on, match the observed data.

Planning modules read current state information from lower-level layers and from state-identification software. They also have specifications of desired goals or objectives that drive the decision making, which involves finding ways to achieve desired goals, given the current state and the limitations of the system. For example, a planner might look at a state where a pump has failed and then build a plan that achieves the desired trajectory maneuvers in that state.

The layering architecture and the multiple sense-think-act loops within the layers are designed to provide an overall sense-think-act loop that controls the underlying system. The sensing combines low-level calculations, methods to abstract and summarize data, tracking of plan execution progress, and desired objectives. The thinking resides primarily with the deliberative layer, which uses this information to determine what the software should be doing. Finally, the acting is handled by the executive and reactive layers, turning the high-level decisions into reality.

Distributed Autonomy

An autonomy architecture provides constraints on the organization, integration, and coordination of a set of components of a single goal-directed system. The same principles underlying autonomy architectures can be generalized to enable the coordination of distributed systems (Simmons et al. 2002). Coordination of distributed systems differs from the coordination of the components of a single system by virtue of the fact that the components of a distributed system may themselves be goal directed. In space applications, distributed coordination may occur in the context of remote sensing (Morris et al. 2005) or construction tasks, such as building a permanent outpost on the moon or Mars (Larson and Wertz 1999). Furthermore, the agents being coordinated may include a mixture of human and robotic agents.

The goal in developing autonomy architectures for distributed systems is to enable collective autonomy; that is, the collective as a whole should exhibit a range of robust goal-directed behaviors. This implies, in addition to the capabilities discussed earlier for single-agent autonomy, the following: (1) Models to support coordinated activity, including information sharing, communication protocols, and so on; (2) Coordinated mission planning, potentially including the ability to form teams to accomplish a mission goal that requires coordinated actions and to generate plans that can be executed effectively by a team of agents; (3) Robust coordinated execution, including timely communication of state information among team members, and adapting to unexpected events.

Benefits of Autonomy

Space exploration is by its very nature an expensive and risky endeavor. The reason for the high expense may need no explanation, but it is worth noting that it goes up rapidly with increased mass and increased distance. The costs involved can typically be separated into the expense of the spacecraft systems development, the transportation (launch) costs, and the operations costs. The addition of crew members increases costs by orders of magnitude because of lower risk tolerance, significantly greater payload mass, and need to return them safely.

The risk stems from a number of factors. The environment is hostile and unforgiving, which requires designers and operators to guard against dangers and factors that are not found on Earth. Spacecraft are also complex machines that are in most cases designed and manufactured in very small numbers, often as a single copy; this reduces the benefit of lessons learned and requires unique expertise for design and operations.

Finally, space exploration involves great distances, which in almost all cases eliminate any possibility of physical access for repairs or adjustments; almost any mistake or failure can lead to an unrecoverable situation.

The high cost and high risk involved in spacecraft operations drive a desire to achieve as much as possible during a given mission. This makes performance a very important factor in spacecraft operations. Autonomy technology has clear benefits in these three areas; risk, cost, and performance.

A space-exploration mission invariably progresses through four phases (Larson and Wertz 1999): The first is an initial study phase, resulting in a broad definition of the mission. Next, comes a detailed development phase, which results in a detailed definition of the mission. The third phase is a production and deployment phase, from construction of the ground and flight hardware and software to launch. Finally, there is the operations phase, the day-to-day operation and maintenance of the space system after launch until de-orbit or recovery of spacecraft. The benefits discussed here apply to many of the phases. However, in order to limit the scope of this article, we focus primarily on the mission operations phase.

Risk Reduction

Space-exploration safety is typically addressed by a combination of design and operational decisions. Spacecraft hardware is designed to handle likely problems or environmental factors and is often made redundant. However, these designs must be traded off against factors such as cost, increased mass, and possibly increased risk, due to the increased complexity. The design of on-board software also plays a key role, providing capabilities for responding to fault situations. In most cases, these capabilities are limited to identification of faults and simple responses that either abort specific activities or put the spacecraft into a safe state and await instructions.

In operational terms, safety has almost exclusively been completely in the hands of human operators. These operators digest and analyze telemetry

data from the spacecraft, determine the state of the spacecraft, decide what needs to be done, and then generate the exact sequences of commands that accomplish their decisions. In this process, flight rules, cross-checking processes, detailed reports, and other safeguards are used to document decisions and avoid human errors and mistakes.

There are three key areas where this approach can be improved. One is that it is fairly costly in terms of manpower and expertise needed for operations. We discuss that aspect here below. A second is that it limits the range and level of detail of fault types that can be handled properly. Finally, it severely restricts the possible onboard responses in the face of faults.

Current spacecraft fault identification and response software is some of the most impressive software ever built. It is carefully designed to provide layers of monitoring capabilities, fault determination, and responses. However, it is written in terms of triggers that kick off predetermined sequences of actions. It does not examine all the available data to reason what the state is, and the conditions for executing recovery actions are predetermined and do not take into account the complete situation in terms of spacecraft state and other ongoing activities. The fallout from this is to limit the reasoning used to identify faults and to severely restrict possible responses. The software cannot identify novel faults and is usually limited to responding by halting operations and calling for help from Earth.

In contrast to traditional software, autonomy technology can provide a much more flexible approach to onboard fault identification and response. Model-based state determination techniques use a model of the spacecraft systems to determine possible states causing a given signature. These can subsume the predetermined triggers and add identification capabilities for a great deal many more situations than the traditional software can identify. In addition, considering additional information when making a fault determination will reduce the risk of incorrectly identifying a fault, something that can have disastrous consequences.

When it comes to response to faults, the autonomy technology can not only take into account the full range of possible fault states but can also take into account what is ongoing or planned and then formulate a safe response based on all those factors. Without the ability to combine all this information, traditional triggered responses will almost always have to halt any and all operations; furthermore, the operations plans may be limited so as to ensure that fault responses won't impact critical activities.

Cost Reduction

The high cost of space exploration is primarily driven by high launch costs, which to a large extent are driven by payload mass. Together, these push the need for very reliable and therefore expensive hardware and software. Finally, spacecraft operations are themselves expensive due to the large number of personnel needed for the duration of a mission. Autonomy technology can do little to directly address launch cost issues, but it can have some impact on hardware and software development cost, and it can support a significant reduction in operations costs.

Spacecraft hardware must operate in conditions that are unlike most any place on Earth. Radiation, temperature extremes, lack of gravity, and dust, in the case of surface spacecraft, have significant impact on hardware and computer systems. But spacecraft are also made more robust to make up for the lack of repair options. This is typically done with redundancy in systems. More adaptable software could, in some cases, reduce the need for redundancy.

Spacecraft flight software is complex and unique. This invariably entails high development and testing costs for flight software. Reuse of tested and certified software can reduce the cost of flight software development. Autonomy software, driven by declarative models, lends itself very well to reuse, and at the same time provides a great deal of capability and flexibility in how the software is used.

The most significant cost-saving impact of autonomy software is in the area of mission operations. Traditionally, mission operations rely on raw telemetry and data being sent to ground staff, who then analyze it, decide on a course of action, encode that course of action in detailed sequences of commands, and then transmit those sequences to the spacecraft that executes them step by step. Autonomy technology can help with operations costs, both by reducing the need for ground involvement and by assisting ground staff with their tasks.

Onboard autonomy can significantly reduce operations costs by off-loading work done by ground staff. At one extreme, a fully autonomous spacecraft requires virtually no interaction with ground staff. However, there are relatively few spacecraft for which that is a feasible paradigm. In the case of science mission spacecraft, operations are invariably driven by scientists' needs, which change based on the data provided by the spacecraft. But data can sometimes be analyzed on board, and suitable response for additional data gathering can be formulated without having to involve ground staff. The autonomous science experiment software onboard EO-1 can accept requests and insert them automatically into the operations plan (Chien et al. 2005a). Even when data cannot be analyzed automatically, autonomy software can be used to allow scientists to specify the science data products they desire and then have the software achieve those while maintaining safety. This reduces the need for staff specialized in the spacecraft itself, as opposed to scientists. In many spacecraft, engineers and spacecraft experts are essential for safe operations; nonetheless, autonomy software can reduce the workload by handling common and noncritical housekeeping tasks and failure responses.

Autonomy can also play a key role in ground operations. Tools based on planning and scheduling techniques are being used to save time and cost in operations. For example, the MAPGEN tool (Jónsson et al. 2000) is used to build activity plans for the Mars exploration rovers. The tool can use plan-

ning and search methods to build plans or suggest improvements, while ensuring that flight rules and constraints are satisfied. The result is less time spent building activity plans, or, more room to explore options for maximizing the value of the plan.

Performance Increase

One of the problems with traditional spacecraft commanding is the time that passes between an event being observed by the spacecraft, be that a fault or an interesting piece of science, and the execution of a response. This delay will often mean lost opportunities and wasted time. As noted above, autonomy technology can reduce or eliminate this delay. This may reduce costs, but more importantly, it increases the time spent on useful operations, as opposed to downtime or waiting time, enabling more to be done in the same amount of time.

Onboard autonomy need not involve fully automated closed-loop decision making and execution. Command sequences have typically very limited levels of complexity, so that ground staff can identify the expected spacecraft state at any point in time. Complex plans, involving event-driven commanding, plan contingencies, fault recoveries, conditional branches, and loops, make this very difficult to accomplish. To address this, autonomy-based techniques can be used both on the ground, to evaluate the safety of such plans, and to watch over execution on board and ensure that flight rules are not violated, resources are not depleted, and so on.

At the other end of the extreme, there are other kinds of missions where communications with Earth simply cannot keep pace with events that occur on or around the spacecraft and that require an immediate response. In such circumstances, the increased performance provided by autonomy software is essential to making the mission possible. For example, an exploration under the Europa ice cap will require fully autonomous event-driven operations from insertion to extraction.

It is worth noting that in some cases, the impact of a given autonomy application can be used to increase safety, reduce cost, or improve efficiency. For example, consider an application that makes it possible to reduce the time needed to build a sequence for a spacecraft. The saved time can be used to reduce manpower and thus cost. The time can be used to explore more options and increase efficiency by getting more done. Finally, the time can be spent analyzing contingencies and fault scenarios for the plan, thus improving safety.

Challenges

Over the last decade, there have been notable successes and advances in autonomy software for spacecraft operations. Nonetheless, many challenges remain to be overcome in this area. These challenges largely break down into technical and cultural issues.

It is worth noting that the very capabilities offered by autonomy software and some of the key motivations for using those capabilities in spacecraft operations give rise to many of the challenges faced. Cost and complexity of space exploration lead to risk aversion, which in turn leads to favoring established approaches over new technologies. The capability of autonomy software to handle more situations and perform more sophisticated operations puts greater demands on computational power and on methods for ensuring correctness.

Technical Challenges

There are some notable technical challenges that remain in the development of spacecraft autonomy software. The core computational challenges involve the capabilities of the software, performance, and correctness assurances. But spacecraft operations invariably involve humans as well, which adds the requirement of working effectively with humans.

The capabilities required for autonomous spacecraft operations are far from being as demanding as general artificial intelligence. Nonetheless, the software must handle noisy sensor data, an uncertain environment, limited resources, complex systems that may not behave as specified, and more. Planning with time and resources, reasoning about uncertainty, automated learning, computer vision, and other AI techniques solve some of these problems, but each has limits, and often the techniques are focused on a type of problem in isolation.

Most AI techniques also require a great deal of computational resources, something that is typically scarce on board spacecraft. On one hand, methods like current vision-based terrain modeling methods are inherently computationally expensive. On the flipside, automated planning need not be too expensive on average but has a very heavy-tailed distribution, which means that at times the computational expense becomes very high.

The complexity of autonomy software gives rise to a very difficult verification problem. Whereas most traditional software can be verified against calculable expected results, the results of autonomy software depend on a much larger input set and involve methods that search through large sets of options before making a choice.

Finally, most autonomy software development has focused on techniques that work in complete isolation. This makes it challenging to adapt the techniques to work in situations where humans are also involved. Mixed-initiative planning techniques are an example of solutions being developed to tackle part of this challenge, but even in that area, a great deal of work remains to be done.

Cultural Challenges

Cultural challenges arise from both the autonomy software developer community and the spacecraft mission community.

The spacecraft mission experts are faced with daunting tasks: designing, planning, and operating missions of great complexity and expense. A great deal of their efforts goes into managing and minimizing risks. And one way to avoid perceived risk is to eliminate new technology and use only proven approaches. This leaves many technology developers frustrated, feeling that the only way to have their technology used in a mission is already to have been part of a mission.

A closely related issue is that auton-

omy software is often seen as replacing human decision making and even pushing out human participation. This does not exactly endear those involved in spacecraft operations to autonomy technology.

Autonomy software developers have also been reluctant to fully understand and acknowledge the complexity of the domains they tackle, and few domains are as complex as spacecraft operations. Many technologists focus on well-defined problem classes that lend themselves to clear and concise technical approaches. The real world, and in particular, spacecraft operations, rarely maps into clear and well-defined algorithmic problems. Nonetheless, technology developers are prone to request that spacecraft operators simply define their problems as AI problems and then all their problems are solved.

Finally, a shared cultural issue is that of distrust of technology promises. Spacecraft operators are skeptical of claims made by autonomy technology developers, in some cases, with good reasons as autonomy technology developers often have more faith in their technology than reality bears out.

Promising Future

These challenges are not insurmountable obstacles but point the way to where technical developments are needed. New ideas and "out of the box" thinking have been and continue to be behind successful examples of autonomy capabilities in space exploration. There is promising work on autonomy capabilities in almost all fields of spacecraft operations, from Earth-orbiting satellites to human spaceflight and to very distant deep space-exploration missions. Examples of intelligent autonomy software applications in spacecraft operations will become more and more frequent in the next decade, as these thorny problems are tackled and the cultural obstacles are overcome.

Acknowledgements

This article is a revised and expanded version of a paper that appeared in the Proceedings of the 2007 IEEE Aerospace Conference, Big Sky, Montana 2007. Those portions that were previously published have been reused here with permission from IEEE.

Notes

1. See saturn.jpl.nasa.gov.

2. See D* Integration into MER, www-robotics.jpl. nasa.gov/tasks/showTask.cfm?FuseAction=ShowTask&TaskID=45&tdaID=2735.

3. See Visual Target Tracking in MER, www-robotics.jpl.nasa.gov/tasks/showTask.cfm?FuseAction=ShowTask&TaskID=18&tdaID=2468.

4. See On Board Science on MER, www-robotics.jpl. nasa.gov/applications/applicationArea.cfm?App=13.

5. Spacecraft "safing" is a general purpose safe-state response that is initiated by both system and subsystem internal fault protection.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Jónsson, A.; Hsu, J.; Kanefsky, B.; Maldague, P.; Morris, P.; Rajan, K.; and Yglesias, J. 2003. MAPGEN: Mixed Initiative Planning and Scheduling for the Mars 03 MER Mission. Paper presented at the Seventh International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS 2003), Nara, Japan, 19–23 May.

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An Architecture for Autonomy. Special Issue on Integrated Architectures for Robot Control and Programming, *International Journal of Robotics Research* 17(4): 315–337.

Bernard, D. E.; Dorais, G. A.; Fry, C.; Gamble, E. B., Jr.; Kanefsky, B.; Kurien, J.; Millar, W.; Muscettola, N.; Nayak, P. P.; Pell, B.; Rajan, K.; Rouquette, N.; Smith, B.; and Williams, B. C. 1998. Design of the Remote Agent Experiment for Spacecraft Autonomy. In *Proceedings, 1998 IEEE Aerospace Conference*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Biesiadecki, J.; Leger, C.; and Maimone, M. 2005. Tradeoffs Between Directed and Autonomous Driving on the Mars Exploration Rovers. Paper presented at the Twelfth International Symposium of Robotics Research, San Francisco. 12–15 October.

Castano, A.; Fukunaga, A.; Biesiadecki, J.; Neakrase, L.; Whelley, P.; Greeley, R.; Lemmon, M., Castano, R.; and Chien, S. 2007. Automatic Detection of Dust Devils and Clouds at Mars. *Machine Vision and Applications* 18(3).

Chang, Y.; Johnson, A.; and Matthies, L. 2005. MER-DIMES: A Planetary Landing Application of Computer Vision. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Los Alamitos, CA: IEEE Computer Society.

Chen, W., and Decker, K. S. 2004. Managing Multi-Agent Coordination, Planning, and Scheduling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2004)*. New York: Association for Computing Machinery.

Cheng, Y.; Maimone, M. W.; and Matthies, L. 2006. Visual Odometry on the Mars Exploration Rovers. *IEEE Robotics and Automation Magazine* 13(2) (June): 54–62.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castaño, R.; Davies, A.; Mandl, D.; Frye, S.; Trout, B.; Shulman, S.; and Boyer, D. 2005a. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. *Journal of Aerospace Computing, Information, & Communication*, April 2005.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castaño, R.; Davies, A.; Mandl, D.; Frye, S.; Trout, B.; D'Agostino, J.; Shulman, B.; Boyer, D.; Hayden, S.; Sweet, A.; and Christa, A. 2005b. Lessons Learned from Autonomous Sciencecraft Experiment. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2005)*. New York: Association for Computing Machinery.

Clement, B., and Barrett, T. 2001. Using Abstraction in Multi-Rover Scheduling. Paper presented at the Sixth International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-Sairas 2001), Montreal, QB, Canada, June 2001.

Coste-Maniere, E., and Simmons, Reid. 2000. Architecture, the Backbone of Robotic Systems. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*. Piscataway, N.J.: Institute of Electrical and Electronic Engineers.

Edwards, L.; Bowman, J.; Kunz, C.; Lees, D.; and Sims, M. 2005. Photo-Realistic Terrain Modeling and Visualization for Mars Exploration Rover Science Operations. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics (IEEE SMC 2005)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Fox, D.; Burgard, W.; Kruppa, H.; and Thrun, S. 1999. A Monte Carlo Algorithm for Multi-Robot Localization. Technical report, CMU-CS-99-120, Carnegie Mellon University, Pittsburgh, PA.

Hayden, S. C.; Sweet, A. J.; and Christa, S. E. 2004. Livingstone Model-Based Diagnosis of Earth Observing One. In *Proceedings of AIAA First Intelligent Systems Technical Con-*

ference. Arlington, VA: American Institute of Aeronautics and Astronautics.

Hofbaur, M. W., and Williams, B. C. 2002. Hybrid Diagnosis with Unknown Behavioral Modes. Paper presented at the Thirteenth International Workshop on Principles of Diagnosis (DX02), Semmering, Austria, May 2002.

Hofbaur, M. W., and Williams, B. C. 2004. Hybrid Estimator of Complex Systems. *IEEE Transactions on Systems, Man and Cybernetics*. Part B, 34(5): 2178–2191.

Jalobeanu, A.; Kuehnel, F. O.; and Stutz, J. C. 2004. Modeling Images of Natural 3D Surfaces: Overview and Potential Applications. In *Proceedings of the 2004 IEEE Computer Science Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'04)*. Piscataway, N.J.: Institute of Electrical and Electronic Engineers.

Jónsson, A. K.; Morris, P. H.; Muscettola, N.; Rajan, K.; and Smith, B. 2000. Planning in Interplanetary Space: Theory and Practice. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*. Menlo Park, CA: AAAI Press.

Larson. W. J., and Wertz, J. R., eds. 1999. *Space Mission Analysis and Design*. Dortrecht, Holland: Kluwer Academic Publishers, 1999.

Maimone, M.; Leger, P. C.; and Biesiadecki, J. 2007. Overview of the Mars Exploration Rovers' Autonomous Mobility and Vision Capabilities. Paper presented at the IEEE International Conference on Robotics and Automation (ICRA) Space Robotics Workshop, Rome, Italy, 14 April.

Meyer, C. 2005. Classification and Content-Based Retrieval of Images for Planetary Exploration. Master's thesis, Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, Switzerland, April 2005.

Morris, R.; Dungan, J.; Edgington, W.; Williams, J.; Carlson, J.; Fleming, D.; Wood, T.; and Yorke-Smith, N. 2005. Coordinated Science Campaign Scheduling for Sensor Webs. In *Proceedings of the Eighth International Symposium on Artificial Intelligence, Robotics, and Automation in Space*. ESA SP-603 August 2005. Noordwijk, The Netherlands: European Space Agency.

Nilsson, N. J. 1984. Shakey the Robot. Technical Report 223, SRI International, Menlo Park, CA.

Pedersen, L; Sargent, R.; Bualat, M.; Kunz, C.; Lee, S.; and Wright, A. 2003. Single-Cycle Instrument Deployment for Mars Rovers. In *Proceedings of the Seventh International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS 2003)*. Noordwijk, The Netherlands: European Space Agency.

Robertson, P.; Effinger, R. T.; and Williams, B. C. 2006. Autonomous Robust Execution of Complex Robotic Missions. In *Proceedings of the Forty-First IEEE Industry Applications Society Annual Conference,* 595–604. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Simmons, R.; Smith, T.; Dias, M. B.; Goldberg, D.; Hershberger, D.; Stentz, A.; and Zlot, R. M. 2002. A Layered Architecture for Coordination of Mobile Robots. In *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 NRL Workshop on Multi-Robot Systems*. ed. A. Schultz and L. Parker. 2002. Berlin: Springer.

Stentz, A. 1995. The Focussed D* Algorithm for Real-Time Replanning. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*. San Francisco: Morgan Kaufmann Publishers.

Tompkins, P.; Stentz, A. T.; and Whittaker, W. R. L. 2004. Mission-Level Path Planning for Rover Exploration. In *Proceedings of the Eighth Conference on Intelligent Autonomous Systems (IAS-8)*. Piscataway, N.J.: Institute of Electrical and Electronic Engineers.

Tumer, K.; Uckun, S.; and Agogino, A. 2005. A Distributed and Adaptive Health and Mission Management Architecture. Paper presented at the Integrated Systems Health Management Conference, Cincinnati, OH, August.

Verma, V.; Estlin, T.; Jónsson, A.; Pasareanu, C.; Simmons, R.; and Tso, K. 2005. Plan Execution Interchange Language (PLEXIL) for Executable Plans and Command Sequences. In *Proceedings of the Eighth International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS 2005)*. Noordwijk, The Netherlands: European Space Agency.

Wagstaff, K. L.; Castaño, R.; Chien, S.; Ivanov, A. B.; and Titus, T. 2005. Towards Orbital Tracking of Seasonal Polar Volatiles on Mars. Abstract presented at the Thirty-Sixth Lunar and Planetary Science Conference, League City, TX, 14–18 March.

Williams, B. C., and Nayak, P. 1996. A Model-Based Approach to Reactive Self-Configuring Systems. In *Proceeding of the Thirteenth National Conference on Artificial Intelligence,* 971–978. Menlo Park, CA: AAAI Press.

Ari Jónsson is the dean of the School of Computer Science at Reykjavik University. He received his Ph.D. in computer science from Stanford University in 1997. For the next 10 years, he worked at NASA Ames Research Center, in the Intelligent Systems Division. His research and development efforts include work on constraint reasoning, planning and scheduling, robust plan execution, mixed-initiative planning, autonomous operations, and validation of autonomous systems. Jónsson has served as the principal investigator on multiple technology research and development projects, focusing on intelligent automation for manned and unmanned spacecraft operations, both in terms of onboard autonomy and ground-based mission operations support. He has received a number of recognitions and awards for his work, including awards for his contributions to the remote agent experiment, which took place on board Deep Space 1 in May 1999, as well as recognition for his work on the infusion of automated reasoning and planning technology into the Mars Rover Exploration mission in 2004.

Robert Morris is a researcher in the automated planning and scheduling group in the Intelligent Systems Division at NASA Ames Research Center. For six years he has served as principal investigator on projects dealing with applying automated information technologies for sensor web coordination. He is currently PI or co-PI on a number of Earth science projects for building applications of automated planning and scheduling technologies. His research interests include representing and reasoning about time, constraint-based planning, and developing search techniques for constraint optimization.

Liam Pedersen (pedersen @email.arc.nasa.gov) is a robotics researcher at NASA Ames Research Center and Carnegie Mellon University, with interests in rover navigation and rover-based exploration of planetary and other extreme environments. He holds a Ph.D. in robotics from Carnegie Mellon (2001) for the robotic identification of meteorites in Antarctica. His team developed single-cycle instrument placement for MER-class vehicles. Current projects include low-cost aerial mapping and robotic exploration of the permanently shadowed lunar south polar craters.