# SPACECRAFT AUTONOMY ISSUES: PRESENT AND FUTURE

**Joan C. Horvath**
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109 USA

Operating robotic scientific spacecraft is a complex process. To reduce operating costs and risks, there has been much discussion of the requirements for automating spacecraft operations as well as automating the spacecraft themselves. It is not always easy to determine which features to automate in space, which ones to automate on the ground, and which functions would best remain in human hands. Higher-capability commercial and military satellites will also have technology needs of their own, and more synergy between these developments and the development of scientific satellites will also be a future trend. This paper discusses the driving forces behind spacecraft automation, and identifies relevant technology areas and needs.

## 1. Introduction

The future of spacecraft operations will be a challenging one as missions are developed to reconnoiter the remaining yet-unexplored bod ies in the solar system, and to go back to planets that were previously targets of quick flybys. These missions will give us a more complete understanding of the other planets in the solar system to understand what is common across all planets and what is unique about Earth. More sophisticated astronomical and Earth-observing scientific spacecraft will be providing us with enormous amounts of data to allow an understanding and accurate modelling of large-scale climactic and geologic phenomena, In turn, this will enable prediction of the short-term local results of such phenomena (such as drought, floods, hurricanes, earthquakes, volcanic activity, and the like).

In the face of national and world needs, fewer and fewer resources are available to be dedicated to scientific spacecraft. Yet, the need to understand natural phenomena becomes steadily more pressing, and requires more and better data. Thus, future spacecraft will need to be flown more cheaply without unacceptable compromise of science goals or reliability. Cost tradeoffs will be made carefully, and the price comparisons between technology development and application versus the use of labor- intensive "known" solutions will need to be analyzed for each case individually to simultaneously optimize price and science return, Higher-capability commercial and military satellites will also have technology needs of their own. There will be more synergy between these developments and the development of scientific satellites.

One major factor in technology requirements for the next set of spacecraft is that many missions have already been done that are, relatively speaking, technologically straightforward, leaving the more difficult missions for future engineers. There are two categories of "difficult" missions: spacecraft to return data from targets that are difficult to reach for some reason, and spacecraft to return in-depth data from those targets that previously had simple reconnaissance. The technological challenges to fill these two gaps are somewhat different.

For those targets that have not had simple reconnaissance, the targets themselves are difficult to reach for some reason. For example, temperatures might be extreme and distances for the spacecraft to travel might be great -- for example, a mission to Pluto. Alternatively, the

target might be "hard to get to" in dynamic terms, requiring a very light spacecraft, a very large launch system, or both. Comets, the poles of the sun, and other bodies requiring high energy trajectories fall into these categories. This implicit] y gives ı ise to several cost and technology drivers: future missions to these targets might be very long and might have to return large amounts of data at a very low data rate (due to small antennas and power limitations onboard). This requirement for smaller, lighter and more reliable spacecraft will in turn drive many other technologies, particularly in the areas of electronics and autonomous operation.

Spacecraft which will perform detailed studies of previously-surveyed targets have somewhat different drivers. Missions falling into this category are the current Galileo mission to Jupiter; the 1996-launch Mars Geoscience Surveyor (MGS) and Mars Pathfinder (fig. 1 and its rover, fig. 2) missions to Mars; and the 1997-launch Cassini mission to Saturn. These missions are planned to orbit (or in the case of Mars Pathfinder, land on) planets for a substantial period of time, taking data with a variety of instruments. Recent earth-surveying and astronomical satellites, which are trying to take more types of data and larger quantities of it, wi ll have similar requirements. These missions will also have the problems of planned long data-returning lifetimes, as well as the need to return very large amounts of data,

In the past it was frequently the case that two spacecraft would be sent to fulfill essentially the same mission. This would ensure that if one spacecraft failed for whatever reason, the other would go on to return at least the most critical science data. Alternatively, spacecraft were flown solely to demonstrate new technology, which would then be used in a subsequent scientific spacecraft. The NASA "New Millennium" program will attempt to fly "engineering demonstration" spacecraft, to relieve the inherent problems when "new technology" has to work the first time and consistently on any and all spacecraft for which it is used.

"New technology" can involve hardware, such as a sophisticated onboard computer. Of equal importance, new software technology can fundamentally change the operational characteristics of a spacecraft. Current emphasis on making spacecraft smaller and more autonomous is one example of a technology push requiring an overall systems approach to developing both new hardware and software. In some ways, a return to a more hands-off relationship with a small spacecraft is a return to the past. Early spacecraft were tiny -- Ranger 1 was a cylinder eighty inches long and six inches across [1] -- and largely autonomous as far as their capabilities went. A "computer - sequencer" onboard dictated actions with some limited commandability (e.g., Ranger 6 and 7) [2]. Miniscule downlink rates, such as the 8 bits per second science data for early Mariner Mars and Venus spacecraft [3] also minimized the ability to interact with spacecraft. A paper in a 1967 conference on the future of spacecraft and ranging stated that, "Projected scientific instrumentation complements require, even in extreme cases, a maximum data transmission rate of about 25 bits/see (provided storage of several thousand bits of data onboard the spacecraft were available.)" [4] The gradual improvement in capability of onboard computers allowed for the current ability to actively control spacecraft with complex stored "sequences" and force the decisions faced today on how much to exercise those command links.

## 2. Anticipated Future Problem Areas

There are two areas that drive spacecraft cost that will need to be addressed in the future. The first onc is the cost to develop and built the spacecraft itself. The second major driver of cost besides the hardware itself is the cost of "flying" the spacecraft, including the software systems to support flight operations. "Concurrent engineeri ng" of both the design of the spacecraft and the design of the operations system are expected to lower the overall costs of future missions

by taking into account operations costs from the very early stages of a project.

During development and construction of a spacecraft a focus on hardware (and secondarily, on onboard software) is expected and the spacecraft operations "problems''(if they are considered at all) are usually considered with respect to whatever hardware difficulty seems to be the most important at the time. To anthropomorphize, when the contractor is "pregnant" with a spacecraft being built, usually the major concerns about its future day to day routine are about any defects that might appear at "birth" or any "genetic defects" that might appear later, and what those defects might mean in "day to day living." However, once the spacecraft has been launched the preoccupation is with honing its daily routine and tasks and understanding its actual intellectual capacity and work habits, This changes if it becomes "sick", particularly in a way that might leave it permanently unable to perform its planned routine,

Thus, it is often not until a scientific spacecraft is flying that its true identity as a robot run by software onboard and on the ground becomes more dominant than its identity as a piece of hardware per se. This has the result that except for dealing with the occasional hardware failure (and this exception drives many design parameters) the process of flying a spacecraft is in large part the process of writing "programs" to run it day to day. As such the process is not too different from the process of software development and shares many of the same problems. Many software development tools have applicability y to flight operations of robotic spacecraft.

In many cases, the utilization of new technology will be driven by the planned amount of autonomy either onboard the spacecraft or enabled by its ground support system. To discuss this further, we need to define autonomy. Most planetary spacecraft are flown by sending onboard programs , or "sequences" which can control the spacecraft with varying degrees of ground intervention. However, normally these sequences are deterministic, sometimes to a fanatical degree, in that extensive ground simulation of every aspect of the execution of the sequence is modeled and tested on the ground sometimes even before launch. This type of spacecraft is not truly autonomous, by the definition to be used for the remainder of this paper. However, these same spacecraft in the face of an anomaly will execute onboard "emergency" algorithms called "fault protection". These algorithms will execute. autonomously and asynchronously in reaction to a perceived anomaly. The interaction of fault protection and sequences does fit the definition of autonomy to be used subsequently in this paper.

"Autonomous spacecraft" do not necessarily need to have their autonomy onboard, however. Highly automated ground operations can also be developed. Essentially, this implies leaving the spacecraft onboard control algorithms simple but increasing the development of automated tools both to prepare commands and to watch telemetry, only requiring human intervention when something highly unusual occurs. One can think of this as the ultimate in distributed computing, with the Deep Space Network and a few billion miles of vacuum taking the place of the more conventional networking cables. Extreme versions of this paradigm have the advantage that it allows humans to intervene more easily than onboard autonomy would allow, and the disadvantage (particularly for long-light-time spacecraft) that a human must intervene in all but the most basic quick-response situations. Ground-based autonomy, however, is not as practical as onboard autonomy in fast-changing spacecraft environments.

Figure 3 diagrams the interrelation of onboard automation and staffing required on the ground. Since it costs money to develop the automation of spacecraft, software development costs of the spacecraft itself will increase as one goes to the right side of Figure 3. Since onboard and/or ground automat ion is relatively new, it is assumed that risk to the mission also increases somewhat as one goes to the right. This risk can be mitigated if there arc also people watching in

parallel with the automation, but this somewhat defeats the purpose (leading to point E on the graph,) A complex spacecraft like Voyager that was flown very deterministically would be an example of point D during its encounters. However Voyager for its Interstellar Mission has staffed down and made operations more routine and streamlined, moving closer to point A on the graph. The Magellan mission to Venus druing its prime mission would be an example of point B: it had a sophisticated set of anomaly resolution algorithms, but under normal conditions it was flown by a moderate-sized team. In this era of low budgets and high science expectations, the current trend is to go towards point C: a capable spacecraft that can "fly itself."

Figure 3 should really have a third axis: spacecraft capability. It is obviously easier to get to point A (or C) for a spacecraft with one instrument than for one with twenty. If any individual spacecraft is smarter and more complex, it will probably also be expensive. This means there will be very few flown, which will tend to drive a project towards point D or possibly E. This cycle can feed on itself, requiring more and more rigorous testing for more and more detailed flight scenarios as a spacecraft becomes more complex and expensive, and as it is more and more inconceivable that the spacecraft might fail.

Alternatives are for spacecraft individually to be less complex and less smart, with more redundant spacecraft, or for spacecraft to be less complex and smarter, again with more redundant spacecraft. Each of these will be discussed below, in the light of what can be thought of as "Murphy's Law of Automation":

> *Spacecraft need to become smarter to become mor e autonomous. However, the more autonomous and complex the spacecraft becomes, the harder it is to determine what is wrong if something does go wrong, and the harder it is to test all paths.*

Automation has different benefits and drawbacks for different types of missions and would have different faces for different types of spacecraft. For example, for "focused missions" where each spacecraft is relatively simple (although there many be a number of spacecraft flying and gathering complementary data at the same time) relatively straightforward automation of many onboard tasks might be possible. When automating a spacecraft, however, it must be remembered that a large part of the operations of any spacecraft must be driven by the resources available to communicate with the ground. For example, the amount of science data that can be stored onboard a spacecraft will be driven in part by how often it is feasible to play it back to earth. The exact times when it is possible to play data back to the ground will in turn drive how many science observations may be taken at a given time. 1 f the spacecraft cannot communicate with earth and take data at the same time (which is often the case) a complex cascade of activity constraints can result which ultimately are driven by the communication schedule of the system that is used on the ground. Hence for a simple mission automation onboard might drive towards repetitive sets of commands that are centered on the times when ground-based tracking is available.

An alternative is for the automated spacecraft to assume that tracking is always available and to have the spacecraft take its data and return it accepting that some fraction of the time communication will in fact not be available. Science data would then go into the "bit bucket." Since current spacecraft are complex and expensive, the latter solution is rarely used (intentionally!) today. However if many spacecraft are flying, and each of thcm is light and simple, one might want these spacecraft to "fly themselves" and return data on a deterministic schedule, which then communications network schedulers (automated or manual) could optimize around the best they could. Autonomy of this type would be relatively low-risk, but would work best with a very repetitive mission. This is in fact not very different from the Magellan Venus mission, which took and returned data on a very rhythmic schedule. There are interesting issues, however, about

what happens in a scheme like this when things go wrong; see the discussion of "Category 2" automation later in this paper.

For these autonomy scenarios, another very important parameter is one-way light time, the time that it takes a signal to go one-way from a spacecraft to the ground or vice-versa. If one-way light time is short, then it is feasible (thought not necessarily advisable) to "joystick" a spacecraft and to count on all anomaly recovery to be conducted from the ground. One-way light time for planetary spacecraft vary widely (fig. 4). All spacecraft at some point have a one-way light time of zero and progress out to their later large values. Hence a given spacecraft might be in a different operations regimes at different points in its life cycle. A second related parameter is the "view period", the amount of time that a spacecraft is in view of a ground station or relay satellite from the time it "rises" to when it "sets." For spacecraft close to earth the light time is negligible but viewperiods might be quite short. This means that unless a quite sophisticated automated scheduling system that understands tracking station rise and set is used, it wi ll be unlikely that a random time to send back data will in fact be in view of a tracking station. Also, since signal strengths are relatively high, earth orbiters create more interference with each other than do planetary spacecraft if they transmit high powered data at a tracking station that some other spacecraft is trying to use.

For earth orbiters, the difference between pointing at, for example, Canberra and Madrid is a very large angle, and if the spacecraft is using a steerable high gain antenna it is necessary to know not only that tracking coverage is available but on *which* ground (or space) antennas it is available. However, for long-light-time spacecraft, the earth is essentially a point source and target, so the spacecraft simply needs to point to "earth". Antennas on the ground usually have a distant planetary spacecraft in view for some number of hours. Signal strength is weak at planetary distances, and there are fewer spacecraft, so transmitting on top of another user when an antenna is not allocated is not as much of a problem as it is for earth orbiters. However, the long light times make scheduling complex (for Voyager, for example, uplink and downlink activities are skewed by the round-trip light time of over fourteen hours).

For complex missions, both for long and short light-time scenarios, the problems of full onboard automation become more complex. Since a variety of instruments are being scheduled, most likely with geometrical constraints and constraints with respect to each other, the need to schedule tracking coverage as another of the constraints probably will raise the complexity level too high to realistically be performed onboard, unless all spacecraft communicate with each other in an automated fashion to get tracking coverage unless radically ncw mission operations scenarios are developed, However, automation tools on the ground (to be discussed in the following sections) can reduce the risk, cost and complexity of flying these spacecraft.

Highly redundant fleets of spacecraft have received both commercial and scientific interest recently. Design of software for these spacecraft has its own particular challenges. For example, the Iridium commercial system is in the production phase for a fleet of 66 spacecraft to provide cellular phone service. Teledesic [5] is proposing a constellation of nearly one thousand spacecraft for wireless communications of various sorts. To take advantage of the commercial investment in this area there has been substantial interest in the scientific community for arrays of spacecraft as well. In the case of a constellation of sophisticated autonomous spacecraft, if one of them failed the others would either need to be sophisticated and fault-tolerant enough to manage around the failure, or the whole system would need to "go down" or operate in a degraded mode while it waited for ground intervention. High-reliability automatic routing around failures and generalized redundancy management is a sophisticated process which can fail in a variety of ways, inchrding''Byzantinc failure." An example of a Byzantine failure is a spacecraft in a constellation

that has an onboard failure, but tells healthy spacecraft that *they* have failed but that it (the failed spacecraft) is just fine. Redundancy management and message routing design to prevent or minimize the effects of these difficulties are areas of cm-rent research for a variety of other critical applications [6].

## 3. Technology for solutions

Based on the above spacecraft taxonomy (simple/complex; short/long one way light time; single vehicles/cooperating fleets) current and projected technology trends can be examined to determine which technologies will be available to assist future spacecraft designers and operators,

The first, and most straightforward, solution to many technical problems can be borrowed from the insurance industry: namely, to spread cost and risk of future spacecraft among a bigger group of investors. It will be easier to take technological risks if those risks are small to any of the individuals investing in the spacecraft. Unilateral efforts by one nation or group to build and fly spacecraft might have a similar ultimate consequence as does the decision by an individual to buy earthquake insurance from a small California-based company or hurricane insurance from a small Florida-based company: if they really want to be covered when a disaster comes, their premiums better be higher than they would if the risk were spread across a national group. International cooperation on large spacecraft spreads the risk among enough nations that the loss of one is less disastrous to any one set of taxpayers. Alternatively, as was the case when Comet Halley was explored, a fleet of complementary small spacecraft can be sent, one from each participating country, and any individual spacecraft can bc innovative without risking the loss of all data if it fails.

Hardware risk is not the only technological risk that a flight project must manage. Software, both onboard and on the ground, poses its own set of issues. Emerging technologies offer opportunities to automate many aspects of space operations. However, the appropriate mix of automated and manual functions is a complex tradeoff that is somewhat different for each mission, particularly when it comes to management of risk in different parts of the spacecraft's software system. Tools are evolving to make software safer, but it is still necessary to ask: "How much automation is too much (or not enough?)"

### 3.1 Artificial intelligence and expert systems

"Artificial intelligence" and "expert systems" are fairly broad terms for automating functions normally associated with people that are expert in some user domain. A variety of expert systems arc used for ground support either as prototypes (running in parallel with flight operations conducted manually) or in some cases, in actual flight operations. Expert systems are in development or in use both for planning and scheduling (uplink tasks), and for telemetry monitoring (downlink tasks).

The possibility of automating parts of the uplink and downlink process is very attractive when missions become very long and it is difficult to keep experts onboard for the duration of a very long mission. It is also expensive to staff a variety of analyst positions all the time. Much planning and scheduling for spacecraft tends to be very repetitive; automating these repetitive portions tends to make these tasks more error-free and enjoyable for the scheduler. There are several approaches to automation in the ground support environment, which will be split into three categories (fig. **5).**

*Category 1: Automate the repetitive, time-consuming "general cases" of-flying the spacecraft, using a fe w simple rules which flag unusual cases. Use people to recognize pathological cases and to manage planning in the face of unusual circumstances and anomalies.*

Using Category 1 automation techniques has implications for the makeup of the flight team to run the system. These teams will require a few highly-skilled operators who will run an "expert-friendly" system. These operators will by and large only make very sophisticated decisions, and will need substantial authority to effectively manage the unusual cases and use their judgment. Most flight project automation tools today fall into this category. This category is particularly useful in cases of planetary orbiters with long-duration science-data-taking missions. Here, much data-taking is repetitive, but usually just enough is happening that requires unusual solutions and tweaks to the pre-launch plan to keep the experts busy. Precisely because software like this tends to be somewhat prescribed in its scope, it is relatively easy to exhaustively test and understand. This type of automation is sometimes called "incremental automation" since tasks can be automated in order of decreasing arduousness to the staff of the project. Many traditional definitions of "artificial intelligence" and "expert systems" might dismiss these applications as not part of the domain, but in reality they are capturing knowledge, albeit simple knowledge. However, since much professional engineering time is spent dispensing routine knowledge over and over again, simple tools like this if carefully designed can make enormous productivity increases.

As a simple example, at the time of its 1992 launch the TOPEX/Poseidon Earth orbiter oceanographic project had approximately one contact with a communications relay satellite (a Tracking Data Relay Satellite, or TDRS) per hour. Explicit spacecraft antenna pointing to an appropriate TDRS satellite needed to be provided in the spacecraft sequence for each contact, each of which required three commands per contact. A small program called MAKER was developed which automatically looked at the allocated relay satellite coverage and generated these commands (plus some other associated tape recorder commands) thereby saving large amounts of sequence engineers' time. MAKER was delivered just before launch to remove this most tedious sequence development task. Gradually, MAKER was been adapted to handle more cases, such as the fact that communication contacts using TOPEX's omnidirectional antenna (instead of its steerable high gain) do not require pointing commands.

*Category 2. Automate the recognition and management of special cases of spacecraft operations, so that scarce and expensive expertise can be released after the spacecraft hardware has been completed and launched.*

Software in this category captures expert knowledge in its rule base but, to make the system testable, limits its user interface to a few simple operator actions. This can be thought of as the "fast-food cashier" model of flying spacecraft, insofar as operators can punch in standard operations in a relatively foolproof way but cannot override the tool easily to create a special set of commands. Operators in this scenario will be limited in their actions, with the rule base essentially having been encoded with the limits of their authority. Any system like this which is actually producing spacecraft commands constrained by rules or generated in accordance to software rules must make allowances for overrides by a "super user" for handling of unanticipated situations. Whether the usual operators will be the ones to override the system or whether an off-line expert will be called back in to authorize and make changes will drive the type of user interface and operators a system like this will require. It is crucial that these staffing assumptions be explicitly discussed and documented at the software requirements stage for such systems, so the developers can all have in mind a consistent picture of "their user" (or the privilege and access differences between their two types of user) from the very beginning of the software life cycle.

A second type of "category 2" system can be an advisor to operators making decisions. In its simplest form, on-line help files for a spacecraft command system's command functions can be thought of as a primitive system of this type. Since these systems are only advisory, they do not need extensive user override capability since they are not in-line with the production of spacecraft commands or processing of data.

An often-unexpected implication of the insertion of a system like this into a process (particularly if the process was manual before) is that the automated system, unlike a human expert, cannot gather information by running into people in the hallways. An automated system often requires much more explicit and precise information than would a human expert, who can make assumptions. This more explicit and precise information if regular] y given to a human "expert system" would probably also result in fewer problems "slipping through the cracks." However, care must be taken that the effort to gather information for the automated system to make its decisions is not greater than the effort to make the decisions manually in the first place would have been. This frequently involves careful coordination of previously disparate electronic systems to be sure that the information that would have flowed to an expert formally or informally flows between electronic counterparts in an accurate way. Setting up a system like this is particularly cost-effective for lengthy missions, particularly ones with very long low-activity or repetitive periods. It is also advisable for the whole ground system to be engineered to allow this type of automation from the beginning. Increasing sophistication of knowledge capture and testing techniques will allow these systems to be easier to build and hence to be worthwhile for smaller and smaller problems. They will also evolve to have more powerful and flexible user interfaces, ultimately evolving into the next category.

Recent discussions of automation in aircraft "glass cockpits" [7] on which type of Category 2 assistance (advisory or controlling) is more appropriate for pilots have resulted in various different design philosophies at major airframe manufacturers. Airbus has trended more towards preventing inappropriate pilot actions while Boeing has used automation more as a reducer of pilot workload. Since pilots can be confused about what an autonomous aircraft is doing, the more aggressive Airbus automation has been cited as a possible cause of accidents [8]. Airbus designers counter that pilots were not using the automation adequately [9] . Boeing has tried to keep the "feel" of its traditional aircraft, even in its new 77'/, to give pilots as many instinctive cues as possible in case they need to take over from the automat ed system [10].

*Category 3. Build a Category 2 system to talk to a Category 1 system more or less directly to completely automate the process of building commands or watching telemetry, with infrequent human intervention.*

Tools in Category *3* would take the place of large numbers of human operators, and might enable one expert operator to manage several spacecraft at once, only intervening in highly unusual situations requiring sophisticated judgment. A problem for critical Category-3 software is that test and validation tools need to be developed that make project management comfortable with completely trusting such tools, particularly as stand-alone tools without human intervention. Efforts in this regard will be discussed under the topics of specification and verification, simulation, and test below. The knowledge-gathering and encoding issues for Category-2 software arc amplified even further in this type of tool.

A variety of tools for mission operations that are mixes of Category 1 and 2 have been developed, some of them with eventual aspirations towards Category 3. These tools are intended for usc in the areas of planning, scheduling, and telemetry modeling and analysis. Many

excellent groups are working in this area, and any sampling will be incomplete. However, the following touches on most of the major categories of tool and the issues involved in generating tools that users will accept and not defeat. The impact a variety of automation tools on the office environment [11 ] has some interesting parallels that should be noted by managers deciding on the correct level of autonomy for their space and ground systems.

An expert system for planning and scheduling of sequences, PLAN-IT-2, is under development at JPL [ 12]. This system (along with its predecessors, PLAN-IT and DEVISER) was intended to create software which approached planning and scheduling in the same way that a human scheduler does. In particular, it attempted to simplify the process of resolving conflicts. PLAN-IT-2 had as a major goal the avoidance of the phenomenon common to some automated systems that they tend *to* either do "too much" or "too little" to the schedule generated by the tool in the face of a new activity added to the schedule. PLAN-IT-2 also has as a design goal that it be relatively simple to adapt for different missions, which will be very important when a variety of small missions are all flying at once and a system must be easy to modify from mission to mission to make automation and knowledge capture for this Category 2/3 system "worth it." Planners need tools with "what-if' type capability as well, and tools like PLAN-IT-2 make it less arduous to try a variety of approaches for optimum science data return, resource usage., or other goals.

In the real-time control area, a set of tools that started out as Category 1 incremental automation tools are evolving into Category 3 connected uplink and downlink tools. This object-oriented system, called MGDS (Multi-Mission Ground Data System) [13] initially consisted of a few programs to generate sequences of events and timelines for real-time operations personnel (as opposed to tools for planners.) The real-time area has somewhat different drivers than the planning area, and prior to the development of these tools there was a largely mission-specific set of utilities that were somewhat of a patchwork. The Operat ions Engineering Laboratory (OEL) at JPL has gradually built up the MGDS set of tools to include editors, programs to strip out items of interest to certain users of a sequence of events, and the like. Eventual l y, they hope to link monitoring of telemetry (downlink) with command functions to truly automate many of the functions of real-time operations.

Spacecraft experts also need tools to assist them with understanding exactly what is planned to occur on the spacecraft as well as what is actually occurring. Several systems to support some combination of real-time on-line spacecraft controllers and subsystem analysts are currently being pursued at JPL. One, the Multimission Automation for Realtime Verification of spacecraft Engineering Link (MARVEL) system [ 14] is designed to monitor spacecraft telemetry and flag departures from expected behavior. MARVEL has been used in spacecraft operations since 1989 on the Voyager project at JPL. MARVEL. combines the use of AI programming techniques with conventional software methodologies, using the technique that works the best in given situations. Initially, MARVEL modules monitored telemetry on a subsystem-specific basis. The program, however, is evolving into being able to flag more system-wide anomalies and to create more links between uplink and downlink tasks to reduce analyst workload.

SELMON ("Selective Monitoring'')[l 5] is a software monitoring system that can be used either on the ground or onboard a spacecraft to detect anomalies more complex than checking that a value is not out of predefine constant "alarm" bounds. SELMON looks for unusual readings from sensors ("unusual" being defined against a variety of modeling and characterization techniques). Once a possible anomaly has been detected, SELMON determines what parts of the spacecraft have likely been affected by an anomaly. SELMON is currently being used in parallel with a flight system in Johnson Space Flight Center's new mission control center, and is under evaluation for a variety of other applications, including onboard applications for the proposed Pluto

Express spacecraft.

The Engineering Analysis Software Environment (EASE) [16] is an environment for development of subsystem-level engineering support tools to analyze telemetry and to review commands before they are sent to the spacecraft. EASE is intended to be common across several missions to enable specialist subsystem engineers to support more than one mission at a time, thus reducing costs to any given mission.

## 3.2. Scheduling Network Resources

The above tools are aimed largely at the users of a particular spacecraft, as they perform the tasks required to fly their machine. However, automation is also crucial in scheduling tracking resources. As has been mentioned above, the availability of resources to return scientific and engineering data or tracking data to the ground often drives schedules for the spacecraft itself. Tools exist both at the user level for determining what resources are desired and for processing the returned data, and at the level of the networks themselves. The rhythms of the major tracking networks' scheduling process drives many other processes in the spacecraft mission operations arena. Two major networks in use today, the TDRSS (Tracking Data Relay Satellite System) and Deep Space Network (DSN) have fundamentally different approaches to scheduling antenna time for their users. The differences are illustrative and bring out important distinctions between different types of appropriate automation under different circumstances, even though superficially the problem being solved might be the same.

The DSN has evolved a system of what can be thought of as "distributed scheduling," Initial forecasts (from requests made years or months in advance) are put together central] y and distributed. The initial forecast in its entirety is then sent to all schedulers for all projects using the network. The schedulers then trade amongst themselves for any other coverage they may need. If disputes arise, the project line managers become involved, not the DSN. This works for planetary spacecraft where coverage times are very long (eight hour viewperiods are not uncommon), needs are frequently known very large periods of time in advance, and all users are civilian. The automation tool to manage the schedule at the DSN so that the antennas know their schedule is called RALPH [17]. RALPH generates the initial schedule from agreed-upon user rules, and users interface with RALPH to change schedules if they have agreed to swap time or give it up. Late-breaking changes are usually handled verbally. A DSN contact cannot be scheduled too close to its use, (typical project requirements for emergency passes being in the hour or two range) since the process of setting up for a contact is quite manual and different for each spacecraft user, although normally users will stay within special "keywords" that define a standard configuration for them. A given DSN contact may have a large variety of activities in it, specified by a combination of "keywords" . However, this largely manual process makes the system able to deal with emergencies, unusual situations, and the occasional very demanding signal-to-noise situations of planetary spacecraft that might be hours and hours of one way light time away.

Scheduling TDRSS for earth-orbiting spacecraft is fundamentally different. Users schedule time on one of two active TDRS satellites in earth orbit. Since there is a requirement to support both open and military users [18] all users cannot see the. whole schedule. Thus, the schedule stays centralized at all times. Further, since earth orbiter view periods are often quite short, earth orbiters use many contacts per day. The closeness of earth orbiters (with negligible one-way light time) also leads to a more volatile scheduling environment. The system is also shared with the Space Shuttle, which leads to even more variability if Shuttle launches slip. The centralization and sheer volume of contacts, and the tendency for change have led to a very different environment from the DSN. TDRSS users develop their scheduling requests and send

them in to a central point electronically. TDRSS'S internal scheduling then develops a forecast schedule, with some combination of verbal and electronic interaction with users. Users must ask for each and every contact explicitly . If changes are desired to a scheduled event, a user talks to a central scheduler and then electronically adds or deletes events as appropriate, which are then centrally confirmed or denied. These events must be taken from a pool of acceptable "configuration codes" which are agreed upon in advance. However, changes to a schedule using these acceptable codes can be made in emergencies up to just a few minutes before an event [19].

A user scheduling resources with the DSN will need different tools than will one scheduling TDRSS, since the software systems on the other end are so different. Users interfacing with the DSN's RALPH to date have had relatively simple tools, since the number of contacts is manageable and RALPH requires a straightforward ASCII request line per contact. Most of the negotiation process for DSN takes place off-line, and the lightness of the electronic interface reflects this. Real-time detailed instructions are somewhat separated from the scheduling process: these instructions are prepared in a "keyword file".

For TDRSS scheduling, more information is provided by the user, the format for submission is complex, and the users work in isolation from each other. One tool for users to schedule TDRSS has been developed by University of Colorado at Boulder. The TDRSS Resource Users Scheduling Tool (TRUST) [20] evolved from the OASIS planning and scheduling system tool [21 ] and is currently in use by the TOPEX/Poseidon project at JPL. TRUST allows a user to create and manage a set of TDRSS schedule requests (for the first six months of the mission, about 160 contacts per week, each contact made up of three separate TDRSS services), and checks user schedules against a set of rules in an expert system. It has an electronic "mailbox" that holds schedule deletions coming in from the electronic interface on the TDRSS side of the system, and alerts a user that there is incoming traffic if necessary. Additional interface tools have largely automated the generation of the initial input schedule, except for the occasional "special request" for a specific spacecraft activity.

Recently, it has been discussed that it would be to everyone's benefit if all tracking networks could be "interoperable", meaning a user would schedule more or less functionally and any available tracking resource would fill the need. Obviously, many of the issues involved in building such a system will require careful thought and analysis of the different requirements that have driven the current variety of scheduling systems, TOPEX/Poseidon during its launch period used both the DSN and TDRSS, with contacts from the two networks interleaved, and interesting scheduling issues arose because of the differing approaches and lead times for the two networks. Since data coming into the project was quite different in format from the DSN and TDRSS, complex communications networking and configuration issues needed to be addressed and tested. Writing a "script" and schedule for all parties to accommodate the differing approaches to launch slips and contingency plans showed that while a given spacecraft under special circumstances can be "interoperable", users will need to be flexible and be able to think in several vernaculars to do so for the foreseeable future,

A problem that arises when one tries to create encoded rules for scheduling any restricted resource (such as antenna time, science observations, etc.) is that the rule developers need to get the users of the resource to agree to the written. down, encoded rule that will always be followed by the software in generating priorities, Frequentl y human schedulers and negotiators will "horse-trade" resources to each other with the understanding that "next time they will be paid back." This is difficult to encode in some absolute form, and is a closely related problem to the difficulty mentioned above for fully automated systems that "can't run into the other automated systems in the hall." Finding the correct level of automatic while understanding that at some

point human intervention may be necessary to soften some of the "absolute encoded rules" is a key to user acceptance and long-term viability of such a system.

### 3.3. Onboard automation issues

What does automation ultimately mean for a "spacecraft user"? If spacecraft is "fully automated" who is "the user"? Should an autonomous spacecraft "do science" and just "call in" if it finds "something interesting"? (How would communication antennas on the ground know this?) These questions have partially been addressed previously in this paper. However, any attempt to truly automate onboard will need to address these issues early in their overall software design. The ground system automation, if any, will need to flow information smoothly from the onboard system.

One particular issue out of these general ones is: what at does it mean to "sequence" an automated spacecraft, particularly one in an unpredictable environment? A regime in which this is particularly interesting is for a planetary rover at some appreciable light-time distance. For rovers, some of the when-is-automation-worth-it arguments become somewhat reversed from other spacecraft. For many spacecraft in a relatively benign and predictable environment, onboard autonomy means a lot of information needs to be provided to the system from the ground to achieve goals, However, a rover which is spending a lot of its time dealing with a local environment in turn would have to pass a high bandwidth of information back to earth (thus requiring a lot more tracking network scheduling, thus requiring more scheduling on the ground....). However, rovers are also operating in a more difficult-to-predict environment than is a spacecraft in orbit and so it is more difficult to define all cases, and so automation tradeoffs must be undertaken particularly carefully here.

Ironically, in current spacecraft, there is one area of onboard "autonomy" which is more or less generally accepted: the "save-yourself" routines of onboard fault protection. This means that most onboard automation is designed for the times when the spacecraft is the least predictable (i.e., in the presence of at least one and possibly more faults.) If onboard automation can be trusted during anomalies, why not for routine work? Fault protection is supposed to be "reflexive" act ion of the spacecraft for which there is no time for ground intervention before spacecraft damage would occur. Other spacecraft actions are "voluntary" and as such can wait for human intervention. A future trend might be to allow more and more routine spacecraft operation to fall into the realm of "reflexes" and fewer and fewer under direct ground control. The fact that onboard automation has dealt with many spacecraft anomalies over the years is also a powerful argument to counter the legitimate concern that in the presence of a cascade of anomalies the onboard system might not be able to cope. A spacecraft, no matter how autonomous, will always have to have an "I give up" mode in which it determines it is hopelessly confused and waits for help, and if it is talking to an autonomous ground system that system as well probably should kick in a "human needed" alarm at that point as well. The discussion of what "that point" is in an individual case will drive many ground and spacecraft design discussions in the coming years.

### 3.4. Navigation and autonomy

For current interplanetary spacecraft, it is frequently the case that navigation (knowing where the spacecraft is) is done on the ground, while attitude determination (knowing how the spacecraft is pointed in three-dimensional space) is done largely onboard or a mixture of onboard and ground-based. Particularly for spacecraft with high navigation precision requirements, large amounts of tracking data may be requ i red. An autonomous spacecraft would probably still need some navigational updates from the ground, although there is now discussion

of various techniques for purely onboard navigation. However substantial requirements for onboard navigation capability or complex attitude determination situations might drive up onboard computing requirements and complexity, with consequent increases in complexity of other onboard software.

### 3.5. Simulation

Since many spacecraft to date have been flown in a very non-autonomous way, it has been very valuable to simulate very precisely what is about to occur onboard the spacecraft, perhaps at the bit level. Spare onboard computers can also be used to run sequences before they are sent to the spacecraft to check for logic errors. This sort of very detailed simulation is excellent for ensuring the success of one-shot encounters that must not fail if science data is to be taken (such as the Voyager planetary encounters, or the orbital insertions of the Magellan Venus orbiter.) It is also useful for validating a routine set of repetitive commands that will then be used over and over again for taking data. As onboard processors get more and more powerful, however, it gets harder and harder for a ground based processor to run faster than real-time and to "keep up with" simulating all processes before they are sent *to* the spacecraft for execution.

Simulation for an autonomous spacecraft might be thought of as more of a "trainer" for the spacecraft than as a "simulator" per se. The current practice of trying to simulate every spacecraft action is like trying to simulate flight 847 from Los Angeles International to New York's Kennedy Airport on Tuesday December 12-- it cannot be done to any valid level if the environment is not fully predictable. Flight simulators try to go through every "path" a pilot's "software" might reasonably be expected to experience, which might be a better model for autonomous spacecraft. This means that simulators would be used mostly pre-launch, or to test changes in onboard autonomous algorithms.

### 3.6 Software test and validation

To determine the correct amount of automation it is necessary to define the areas in which software is a possible risk and to examine whether that risk is acceptable. Software by its very nature is difficult to test completely, since it is very difficult (if not impossible) to anticipate every situation that will occur in the life of the software, including all the possible overlaps and combinations of special cases. At the moment, since software tends not to be fault-tolerant and to be very non-physical, the safety of complex systems can hinge on single logic branches that are incorrect. A variety of researchers have worked for some time to understand the problems inherent in real-time control software (such as a spacecraft flight software and sequence.) There are discussions of the issues and relative merits of different testing techniques for various situations in Nancy Leveson's work[22].

SOmc other techniques with application to determining the correctness and safety of a set of spacecraft commands is described have been developed by Robyn Lutz [23]. These techniques allow algorithmic checking of one or more asynchronous processes against time-dependent constraints. That is, in a spacecraft example, if a sequence (one process) was running and three or four fault protection routines (three or four other processes) kicked in for some reason asynchronously, these techniques could tell a user if any user-specified time constraints would be violated. The user would specify constraints in the nature of "component A must not turn on within five minutes of Component B". These techniques could determine if there were any combinations of start times of the processes which would then violate these constraints. Test techniques like this may make spacecraft designers more comfortable with more asynchronous, autonomous onboard operations.

Another validation technique is formal specification of a flight system. Projects typically develop "flight rules" (a subset of which are the type of temporal constraints alluded to above) which are then checked by some means before commands are sent to a spacecraft. Techniques for specification and verification of software are being developed in many quarters to deal with issues of validating critical software, These techniques are being implemented to check flight rules of spacecraft [24], A finite-state-machine model of the instrument is developed. The states of the spacecraft are the states of the finite-state machine, and spacecraft commands cause a spacecraft to transition between these states. The tinite-state machine representation can be made into a table of states and commands, called an "action table", which can then show what happens for each command in each state. Using these representations, constraints can simply be stated in an English-1ike form. This representation also lends itself to ease of entry of constraints and models, making it possible that there could be personal automation tools which each analyst could easily modify to do his or her own checks, analogous to a spelling checker in a word processor which has everyone's own particular words, acronyms, etc. encoded in it. A version of this system, called the Specification and Verification Environment (SAVE) has been implemented on a parallel computer and is currently being implemented on a sequential computer to verify TOPEX/Poseidon sequences before uplink [25].

### 3.7. **Distributed and parallel computing and advanced networks**

Automation tools on the ground or in space may require substantial computing resources. Higher computing speed in ground systems allows users a means to get away from "batch processing" of spacecraft command files. If verification and validation tools for spacecraft commands are fast enough to be interactive, then users will tend to allow a machine to do checking for them and will be able to use their own time more effectively. This is analogous to the days of programming on cards. When a large batch run would take overnight, users would spend a lot of time finding their own syntax or logic errors on cards before handing them in. Now, most programmers let compilers find most of their syntax errors without lengthy searches through the code.

Recent trends indicate that a cost-effective, fault-tolerant method for obtaining high computing power is to use distributed and/or parallel computing. Distributed computing uses several workstations or other computers, each of which is somewhat autonomous, to solve a computing problem in concert. Parallel computing also uses several computers in concert to solve a problem, but usually in much more tight synchronization, and usually with just one interface for a given user, Many computing problems in mission operations lend themselves to parallel and distributed implementations [26] [27] and some implementations of prototypes have been achieved. More will be expected as more programming tools are available and wider acceptance of these computing systems occurs.

There has been much recent interest in distributed mission operations as well [28]. In a distributed mission operations environment, different portions of the flight team reside in different geographical areas. If the operations team becomes very scattered and many projects begin to operate this way, a demand for high-speed and high-bandwidth networks will arise to carry all the data and commanding, Electronic mail and other efficient means of carrying information quickly to dispersed individuals will also see an increase in demand in this environment. Security issues will become very important as well to prevent unauthorized commanding or interruption of the legitimate commanding process. For small missions, to distribute or co-locate the operations team will begin to be a very important trade and cost driver since even a very efficient distribution system costs something and requires some amount of central

coordination. It may evolve that small missions will all share common central coordination facilities, with some of their operations teams remaining at their home institutions.

A final challenge to the automation community is the need to come up with newer and faster algorithms for autonomous systems. The computational fluid dynamics community has for some time been limited by the hardware available for compute-intensive three-dimensional modeling. It has often been pointed out that computing speed due to hardware improvements alone has been nearly exponentially increasing for many years. However, it is less well known that in the scientific computation community, computing speed improvements have been increasing independent y more quickly due to algorithmic improvements alone [29]. Search and reasoning algorithms need to have the same kind of "grand challenge" emphasis placed on them as has been placed on the more traditional finite-difference and finite-element methodologies in "scientific computing" to enable autonomy with reasonably sized computing hardware, particularly fault-tolerant parallel and distributed hardware,

Novel hardware and new microelectronics also may have revolutionary impacts on future spacecraft. For example, neural networks computing chips might allow for innovative reasoning and sensing algorithms. High-speed massive memory onboard might allow for high flexibility of data return compared to the current slow tape-management systems required. More fault-tolerant hardware and software might enable missions that might be too high-risk otherwise.

### 3.8. **Data archive, retrieval and visualization**

One frequently-neglected technology area that can make an enormous cost difference to a project is the choice of technology to save science data from the spacecraft and make it accessible-to scientific users. As more and more spacecraft start flying, and each of them produces more data, reliable and inexpensive storage technology will become a larger fraction of the cost of an individual project. Database management techniques and high-speed access devices, as well as compact media, will be essential to avoid drowning future generations in seas of obsolete-format magnetic tape !

### 3.9 **Microspacecraft**

Another technology area is the use of microspacecraft (fig. 6) . Since these spacecraft have very severe limitations on power, mass, and volume, much advantage can be taken of judicious usc of new technology. Since part of the idea is that there bc large numbers of these spacecraft flying at once, it would be helpful if each of thei n did not need excessive "care and feeding." This would tend to drive toward some autonomy, and since the spacecraft will be simple quite high degrees of autonomy might work well in this area.

## 4. **Conclusions**

This paper has summarized some existing trends in spacecraft mission operations, particular in the areas of autonomy and ground software. Some major themes have arisen in the course of this discussion. One is that it is essential when developing a mm-e automated or partially automated system that the entire end-to-end design be considered, including designing spacecraft to be easier to fly (and to diagnose during failures). Although this paper has focused more on the uplink operations, it will also ease automation of downlink operations if data collection can be designed so that data is easier to process, analyze, distribute and archive. This link between uplink

and downlink design will be crucial in new, complex yet cost-conscious spacecraft operations, and will make it more and more essential that all hardware and software requirements are clearly defined early on in the design process.

As robot spacecraft get smarter, flying them gets more and more like training a pilot than programming a computer. The community must establish that the pilot is "trained" adequately and have the "training" and "continuing certification" costs be more like that of a Cessna pilot than a space shuttle pilot. Interfaces with any remaining human users must be considered as well [30]. The research community needs to concentrate effort on the technology areas outlined in this paper, as well as many others, for this to happen.

Finally, the research community cannot develop technology alone. The flight community must make their technology needs clear, and then accept solutions as soon as they are reasonably mature. A similar relationship exists between clinical medicine practitioners and the research biomedical community. Normally, a general practitioner is a long time removed from current laboratory results, and is somewhat dependent for dissemination of results from laboratories and approval of new treatments after clinical trials by the FDA. In fact, it is difficult if not impossible and illegal (particularly in the current malpractice environment) for a clinical practitioner to simply try most new techniques, even if he or she were aware of them. The biomedical community relies heavily on programs to conduct clinical trials that cross over from the research community into clinical practice. Recently, the customers of the clinical world have complained that the process of moving new technology is taking too long, is too unwieldy, and too expensive.

Flight operations needs to develop its own high speed system of allowing "clinical trials" of new technology. There must be new technology incentives for flight projects, so that managers are not afraid of "malpractice" problems should they try a new approach. Right now, crossover research and operations organizations tend to be somewhat homeless and lacking in real positions on either side of the fence. There are several reasons for this, in addition to the direct analogies to the medical situation. Research organizations tend to be small organizations with much emphasis on publication and on novelty of algorithms and concepts. There is a strong connection of individuals with concepts, and usually only funding for small prototypes of any idea. Competition for funding is intense, and hierarchy and formality of organizations tend to be minimal.

Operations organizations, however, are designed to be hierarchical and to somewhat rigidly define functional boundaries to avoid chaos, since most flight projects are very large compared to most research endeavors and hence need more structure. However, this means that researchers with ideas that would, for instance, consolidate five flight functions must gain acceptance by all users all the functions. Then, the different organizations would need to decide which of the five organizations would now run the consolidated tool. Needless to say, this is not easy, and automation efforts in particular sometimes die since they are not accepted by enough parts of a system to be "worth it."

Similarly, flight projects tend to think in terms of "positions" instead of individuals, and frequently documents are unsigned or signed by an organization. This means that publishable work (usually, the only recognition of legitimacy in the academic world) will frequently be lost by a research person who lives as part of the necessarily-large team on a flight project for any length of time. Likewise, a flight project person who moves to do research for a while may be viewed as without adequate "real" experience to enter flight operations at a level similar to the one he or she occupied in an academic setting. This makes it very difficult for people to transition easily back

and forth to make academics aware of "real problems" and flight projects aware of the availability of technology that could be transferred.

There are no easy answers for these cultural difficulties, and the only long term solution is open-mindedness and willingness to take chances on both sides. It is essential for the future of mission operations that these cultural differences be worked out. Similar dichotomies exist in most industries, and the competitiveness of our space program and the rest of our economy depend upon it as we move into the next century.

## 5. The Future

Automation is an area of significant effort in a variety of sectors, and hence it is difficult to reliably predict where the field will be in five, ten, and fifteen years. However, the sheer number of spacecraft being launched in the commercial sector would tend to indicate that some spacecraft-management software will be developed in the short term (5 years) to allow large constellations to be managed by a reasonable number of people. Whether this software will be ground or space-based is less clear, however. Commercial satellites are in earth orbit, where the light time is short and the disadvantages of ground-based automation are few. For planetary spacecraft, the trend is for more aggressive onboard automation. Some of the technology developed over the next five years for planetary spacecraft -- for onboard navigation and attitude control, for example -- may then migrate to the earth-orbiter sector over the following five years. If significantly more compute power is available over time, more automation might begin to migrate onboard, leading to the possibility of "no-uplink" spacecraft for some applications in the fifteen-year timeframe.

## *6.* Acknowledgments

The future of mission operations will be enhanced with the work of many researchers and flight operations personnel working together. This paper drew heavily on the work of the authors cited below, many of whom made time available to discuss their work in addition to their published descriptions. Several contacts essential to the development of the paper were made on the "Systers" computing electronic mail forum, managed by Anita Borg of the Network Systems Laboratory of the Digital Equipment Corporation. Finally, invaluable discussions with Kathryn Weld, Sanford Krasner, Richard Doyle, Dennis Page, Robert Gustavson, Leon Alkalaj, Karl Schneider, Joseph Spitale, Bridget Landry, Robert Stiver, Frank Salamone, Kevin Miller, and David Collins, all of JPL; with Gregory l'icard of Telos; and with a variety of others too numerous to mention were helpful in solidifying the ideas presented in this paper. Any incorrect interpretation of their views of mission operations are those of the author alone. Some of the work in this paper was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration.

## 7. References


1 Burrows, W. E., *Exploring Space: Voyages in the Solar System and Beyond.* Random House, NY. (1990) p.78.

2 Ibid., p. 147-174.

3 Hunter, M. W., and Moll, R. L., "Future Interplanetary Unmanned Space Missions." in *Future Space Program and Impact on Range and Network Development, v.* 15 of AAS Science and Technology Series, Morganthaler, G.W. (cd) p. 114. (1967)

4 Ibid.

5 McCaw, C., Tuck, E. F., Daggatt, R. W., Davidson, T. W., Tobey, M. L., "Application of Teledesic Corporation for a Low Earth Orbit Satellite System inthe Fixed Satellite Service." U.S. Federal Communications Commission (FCC) frequency license filing, Washington DC. March 21, 1994.

6 Owre, S., Rushby, J., Shankar, N., von Henke, F., "FME '93: Industrial-Strength Formal Methods," p. 482-500, Odense, Denmark. Volume 670 of Springer-Verlag *Lecture Notes in Computer Science.* (April 1993).

7 Hughes, D., Dornheim, M. A., "Accidents I Direct Focus on Cockpit Automation" p. 52-60; Hughes, D. " Incidents Reveal Mode Confusion", p. 56; Dornheim, M. A., "Dramatic Incidents Highlight Mode Problems in Cockpits." p. 57-59; Dornheim, M. A., "Modern Cockpit Complexity Challenges Pilot Interfaces.", p. 60-62; Scott, W. B., "Certification Officials Grapple with Flight Deck Complexity." p. 64-65. All in *Aviation Week and Space Technology,* **142:5** *(30* January 1995).

8 Phillips, E. H., "NTSB: Mode Confusion Poses Safety Threat". *Aviation Week and Space Technology,* **142:5**, *63-64 (30* January 1995).

9 Sparaco, P., "Airbus Seeks to Keep Pilot, New Technology in Harmony". *Aviation Week and Space Technology, 142:5,62-63 (30* January 1995).

10 Hughes, D., "Fly-by-wire 777 Keeps Traditional Cockpit." *Aviation Week and Space Technology,* **142:18**, *42-48* **(1** May 1995).

11 Hirschman, R. A., "The Effect of A Priori Views on the Social Implications of Computing: The Case of Office Automation." *ACM Computing Surveys,* **18:2,** 165-196 (June 1986).

12 Eggemyer, W. C., and Cruz, J. W., "PLAN-IT-2: The Next Generation Planning

and Scheduling Tool." *Telematics and Informatics*, **7:3/4**, 189-207 **( 1990).**

13 Murphy, S. C., Miller, K. J., and Louie, J. J., "Object-Oriented Technologies in a Multi-Mission Data System." Presented at SpaceOps92 conference, Pasadena CA Nov 17-20, 1992; in conference Proceedings (1993).

14 Schwuttkc, U. M., Quan, A. G., Angelino, R., Childs, C. L., Veregge, J. R., Yeung, R, Y., and Rivera, M.B., "Marvel: A Distributed Real-Time Monitoring and Analysis Application." in Scott, A .C., and Klahr, P. (eds) *Innovative Applications of Artificial Intelligence 4*, AIAA Press and MIT Press joint publication, 1992.

14 Doyle, R. J., Chien, S. A., Fayyad, U. M., Wyatt., E. J., "Focused Real-time Systems Monitoring Based on Multiple Anomaly Models. " International Qualitative Reasoning Conference (QR 93) Eastsound, WA, May 1993.

16 Bahrami, K. A., and Harris, J., "An Advanced Environment for Spacecraft Engineering Subsystem Mission Operations." Paper 929101, *Vol. 1: Aerospace Power, SAE/AIAA 27th Intersociety Energy Conversion Engineering Conference Proceedings,* IECEC 1992, San Diego, CA 3-7 August 1992.

17 Durham, R., Reilly, N.B, Springer, J. B., "Resource Allocation Planning Helper (RALPH): Lessons Learned." *Telematics and Informatics*, **7:3/4**, 189-207 (1 990).

18 Happell, N., Moe, K. L., Minnix, J., "Scheduling the Future NASA Space Network: Experiences with a Flexible Scheduling Prototype." Presented at SpaceOps92 conference, Pasadena CA Nov 17-20, 1992; in conference Proceedings (1993).

19 Networks Technical Training Facility, "TDRSS Constraints and Scheduling Procedures." STDN document 1153.4, Goddard Spaceflight Center, January 1992.

20 Laboratory for Atmospheric and Space Physics, "TDRSS Resource User Scheduling Tool (TRUST) User's Guide." University of Colorado at Boulder, Boulder Colorado (1992).

21 Thalman, N., Sparn, T., Jaffres, Gablehouse, D., Judd, D., and Russell, C., "AI Techniques for a Space Application Scheudling Problem." 1991 Goddard Conference on Space Applications of AI, NASA Conference Publication 3110.

22 Leveson, N. G., "Software Safety: What, Why, and I1ow." *A CM Computing Surveys,* **18:2, 125-164 (June 1986).**

23 Lutz, R. R., and Wong, J.S.K, "Validating System-Level Error Recovery for Spacecraft." American Institute of Aeronautics and Astronautics AIAA-91-3714-CP, 1991.

24 Alkalaj, L. J., "Towards a Specification Language and Programming Environment for Concurrent Constraint Validation of Spacecraft Commands," JPL Internal Report, July 1992.

25 Horvath, J. C., Alkalaj, L, J., Schneider, K. M., Amador, A. V., and Spitale, J. N., "The 'Instant Sequencing' Task: Toward Constraint-Checking a Complex Spacecraft Command Sequence Interactively." Presented at SpaceOps92 conference, Pasadena CA Nov 17-20, 1992; to appear in conference Proceedings (1993).

26 Horvath, J. C., and Perry, L. P., "Hypercubes for Critical Spacecraft Command Verification." AIAA Paper 90-5095, presented at AIAA/NASA 2nd International Symposium on Space Information Systems, Pasadena, CA , 17-19 September 1990. Also Schwuttke et.al., ibid.

27 Hull, L., Hansen, E., and Sparn, T., "Distributed Planning and Scheduling for Instrument and Platform Operations." AIAA Paper 90-5089, presented at AI AA/NASA 2nd International Symposium on Space information Systems, Pasadena, CA , 17-19 September 1990.

28 Wall, S. D., and Ledbetter, K.W. *The Future of Mission Operations Systems.* Taylor and Francis, London, 1991, Chapter 8.

29 Office of Science and Technology Policy, Federal Coordinating Council for Science, Engineering and Technology, Committee on Physical, Mathematical, and Engineering Sciences, "Grand Challenges: High Performance Computing and Communications: A supplement to the Fiscal Year 1992 Budget of the President of the United States." Published by National Science Foundation, Computer and Information Sciences and Engineering, 1800 G St NW, Washington DC. (1992).

30 Hartson, H.R., and Hix, D., "Human-Computer Interface Development: Concepts and Systems, " *ACM Computing Surveys,* 21: 1, 5-92 (March 1989).

FIGURE CAPTIONS:

Figure 1. The Mars Pathfinder lander and rover.

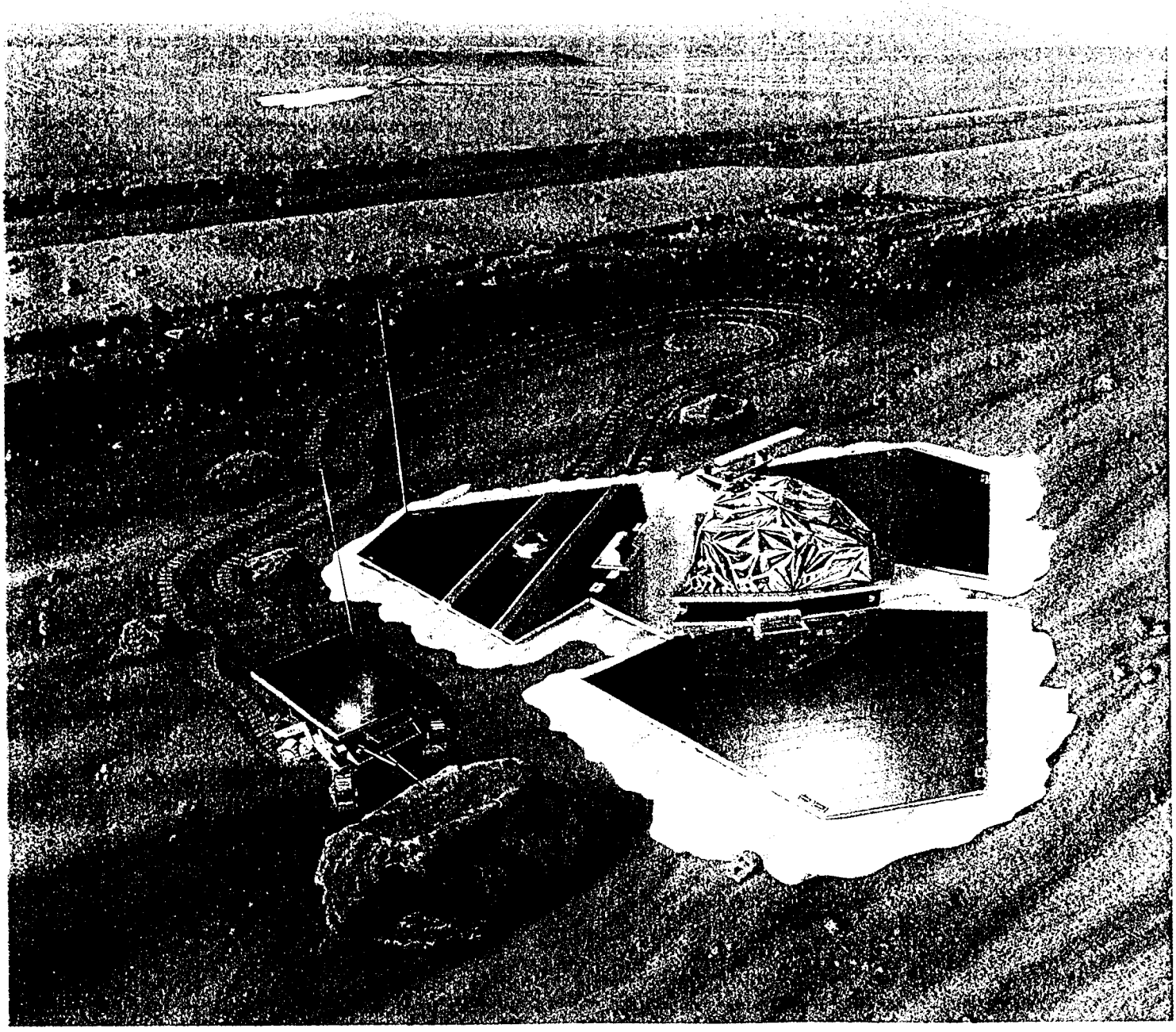
Figure 2. Closeup of the the Pathfinder rover.

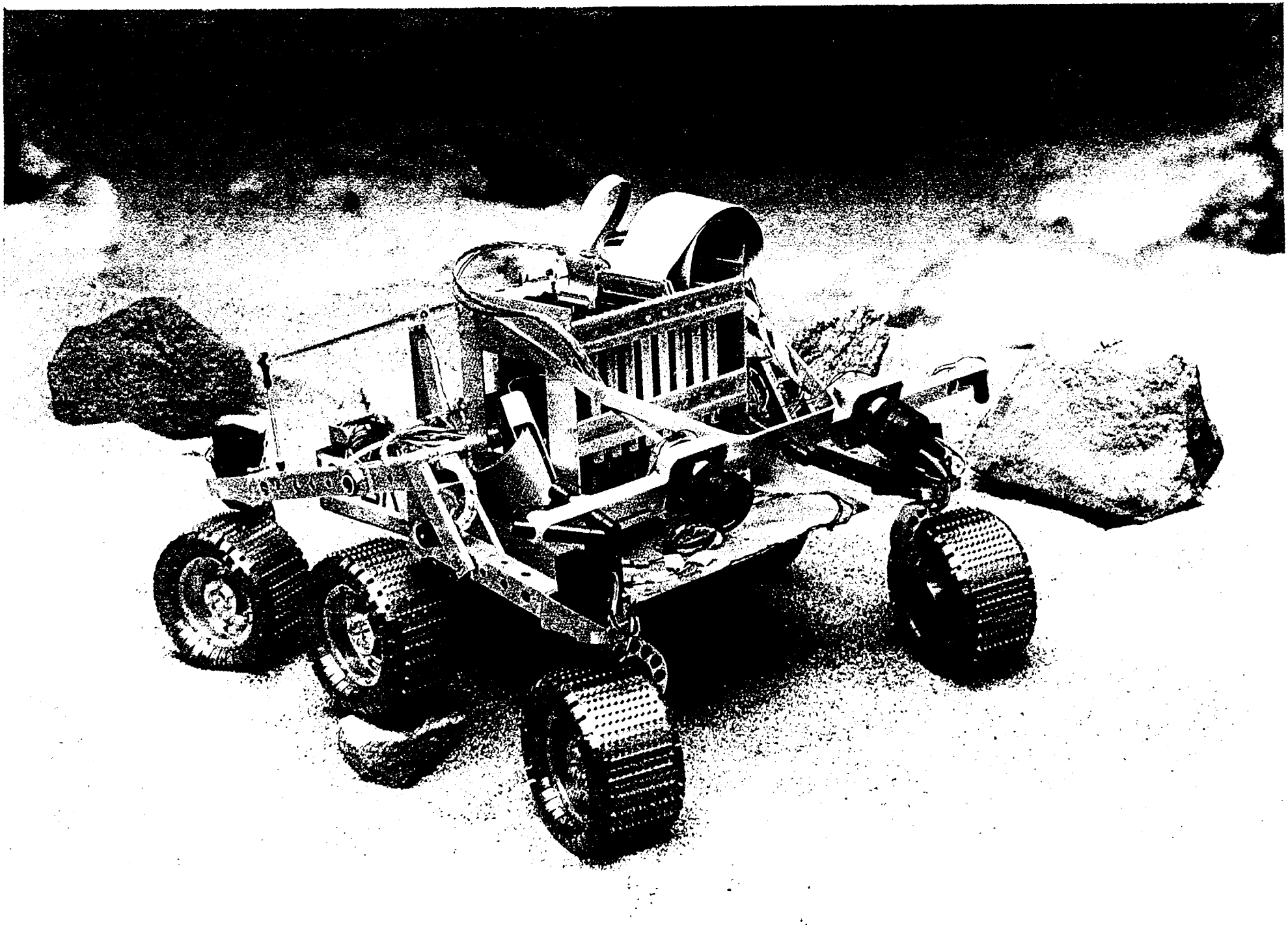
Figure 3. Trends in automation level and associated parameters.

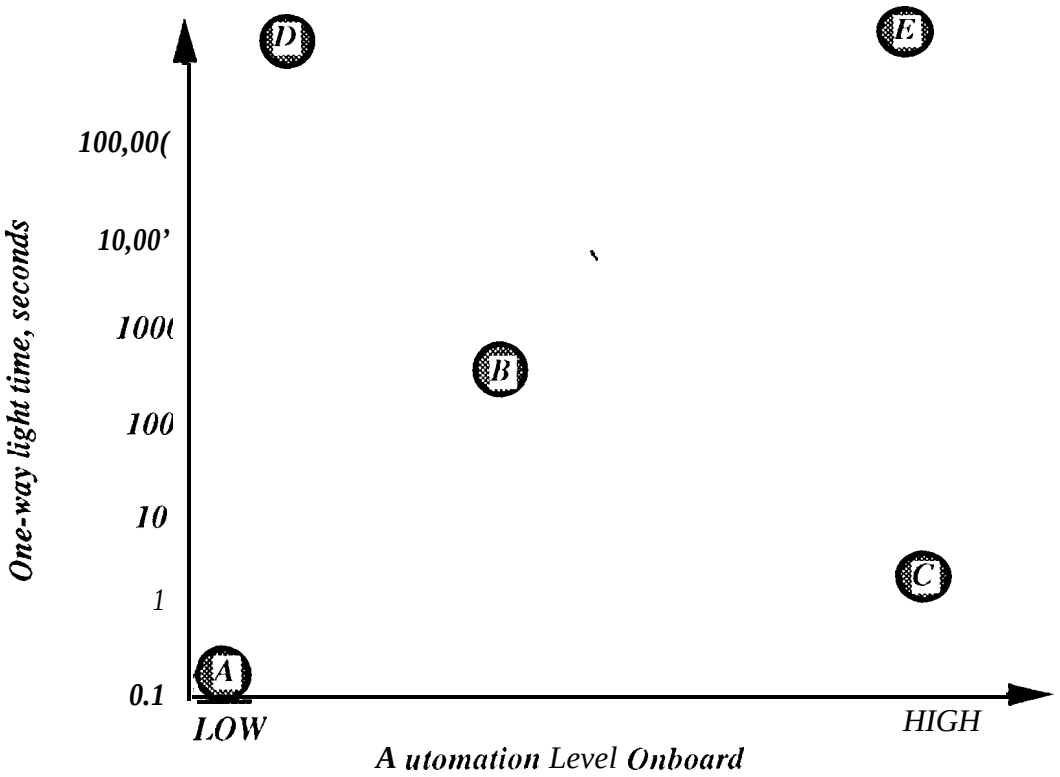
Figure 4. One-way light times at some mission times of interest.

Figure 5. Categories of automation systems.

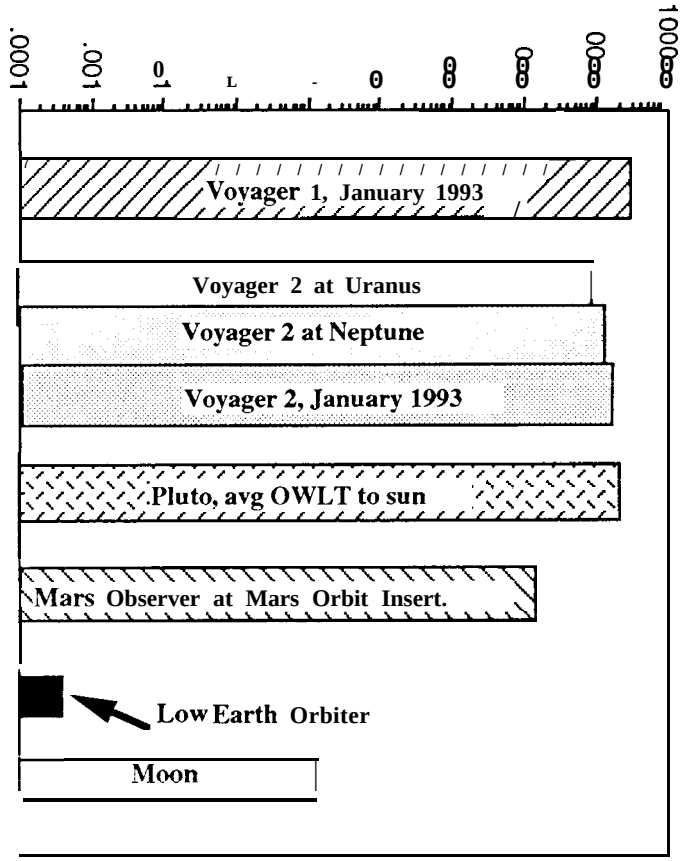
Figure 6. A microspacecraft.

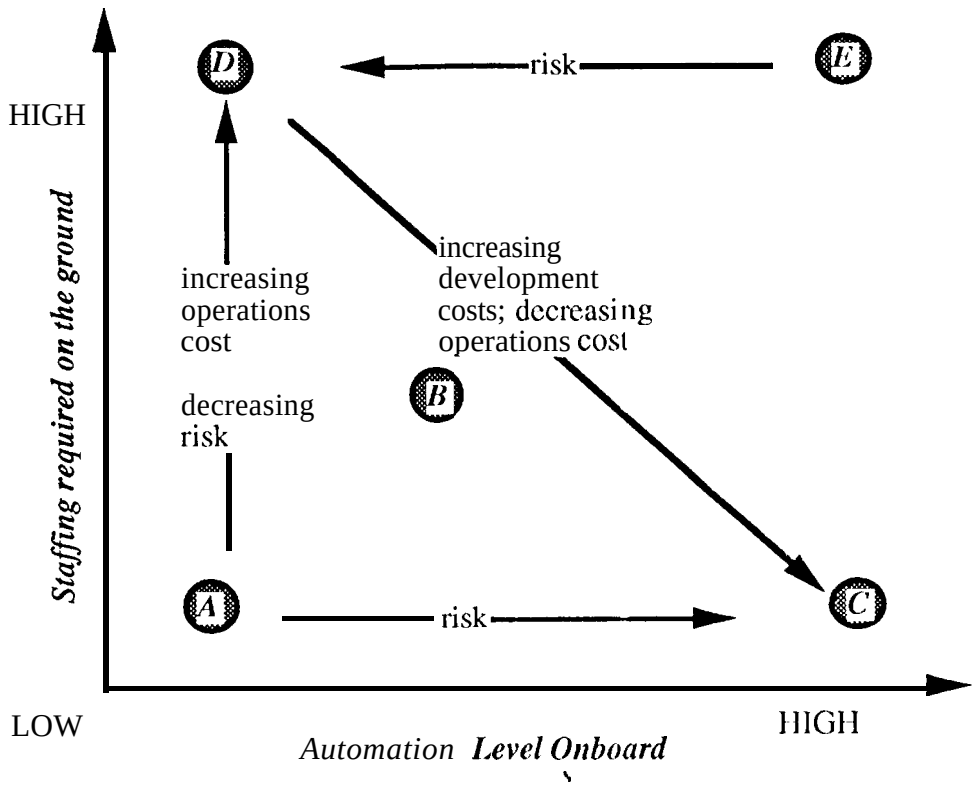







**NOTE 2.8** hours = 10,000 seconds, **27.8** hours = 100,000 seconds


One-way light time, seconds
.0001
.001
0 1
L
0
0
0
0
100000
Voyager 1, January 1993
Voyager 2 at Uranus
Voyager 2 at Neptune
Voyager 2, January 1993
Pluto, avg OWLT to sun
Mars Observer at Mars Orbit Insert.
Low Earth Orbiter
Moon


D
risk
E
HIGH
Staffing required on the ground
increasing
operations
cost
increasing
development
costs; decreasing
operations cost
decreasing
risk
B
A
risk
C
LOW
HIGH
Automation Level Onboard

| Category | Type Of knowledge encoded | Type of user expected | User override frequency |
|---|---|---|---|
| 1 | Simple | Sophisticated | Frequent |
| 2A | complex (control) | Unsophisticated | Seldom |
| 2B | Complex (Advisory) | Sophist icated | Not applicable |
| 3 | Complex and simple | Sophisticated | Seldom |