To be published in the proceedings of the founding convention of the Mars Society, August 1998, Boulder CO.

Model-based Autonomy for Robust Mars Operations

James A. Kurien† P. Pandurang Nayak* Brian C. Williams

NASA Ames Research Center
MS 269-2, Moffett Field, CA 94035
{kurien, nayak, williams}@ptolemy.arc.nasa.gov

ABSTRACT— Space missions have historically relied upon a large ground staff, numbering in the hundreds for complex missions, to maintain routine operations. When an anomaly occurs, this small army of engineers attempts to identify and work around the problem. A piloted Mars mission, with its multiyear duration, cost pressures, half-hour communication delays and two-week blackouts cannot be closely controlled by a battalion of engineers on Earth. Flight crew involvement in routine system operations must also be minimized to maximize science return. It also may be unrealistic to require the crew have the expertise in each mission subsystem needed to diagnose a system failure and effect a timely repair, as engineers did for Apollo 13.

Enter model-based autonomy, which allows complex systems to autonomously maintain operation despite failures or anomalous conditions, contributing to safe, robust, and minimally supervised operation of spacecraft, life support, ISRU and power systems. Autonomous reasoning is central to the approach. A reasoning algorithm uses a logical or mathematical model of a system to infer how to operate the system, diagnose failures and generate appropriate behavior to repair or reconfigure the system in response.

The "plug-and-play" nature of the models enables low cost development of autonomy for multiple platforms. Declarative, reusable models capture relevant aspects of the behavior of simple devices (e.g. valves or thrusters). Reasoning algorithms combine device models to create a model of the system-wide interactions and behavior of a complex, unique artifact such as a spacecraft. Rather than requiring engineers to envision all possible interactions and failures at design time or perform analysis during the mission, the reasoning engine generates the appropriate response to the current situation, taking into account its system-wide knowledge, the current state, and even sensor failures or unexpected behavior.

1. Introduction

Exploring and ultimately settling Mars will be a milestone in the development of our civilization and an uncompromising measure of our courage, cleverness and resolve. Accordingly, it will also be an unprecedented technical challenge, involving multiple interdependent mission elements, multiyear duration, incredible budgetary pressure and the duty to protect human lives in a harsh environment millions of miles from Earth. Evidence of the utility of highly capable, robust and coordinated autonomous systems in meeting this challenge pervades mission scenarios such as Mars Direct [1] and the NASA Mars Reference Mission [2].

Model-based autonomy involves the use of automated reasoning engines and high level models of the system being controlled to generate correct system behavior on the fly, even in the face of failures or anomalous situations. This approach is proving to be a robust and cost effective method for developing more highly capable autonomous systems than have been deployed in the past and might prove invaluable to the development of piloted missions to Mars.

The next section of this paper describes why autonomous systems are needed to explore Mars. Section 3 briefly discusses the varieties of model-based autonomy research going on at NASA Ames Research Center. Section 4 discusses how this work can contribute to cheap, safe, robust, and minimally supervised systems on Mars. Section 5 describes Livingstone, one of the reasoning engines developed at Ames that will be tested onboard a spacecraft next year. Section 6 describes a number of Mars-related testbeds that are making use of model-based autonomy technology.

This paper is meant to serve as an introduction to the concepts behind model-based autonomy for those who are not computer scientists and as a rough position paper regarding how those concepts might assist in a journey to Mars. The References section contains pointers to a number of papers on model-based autonomy with more technical detail and concrete explication.

---

†Caelum Research Corporation

* Recom Technologies

2. THE UTILITY OF AUTONOMY ON MARS

The need for robust, inexpensive and productive operation of remote assets on Mars appears throughout both the Mars Direct scenario and the Mars Reference Mission. In both of these mission designs, initial mission elements such as in-situ propellant production (ISPP) plants and the crew return vehicle must be able to operate for a period years in a harsh environment with limited downlink capabilities and a reduced set of ground control personnel. Such systems must maintain efficient operation in spite of unexpected failures, novel environmental phenomena and degraded system capabilities. Safety places high demands on system robustness: the crew cannot depart Earth if propellant plant down time results in inadequate production or if the return vehicle cannot verify nominal operation.

Once the crew does depart Earth, they will be travelling two orders of magnitude farther from home than the Apollo crews. They will be separated from mission control by thirty-minute communication delays and potentially multi-day communication blackouts imposed by the relative positions of Mars and the Earth. There will be a number of systems upon which the crew's ability to reach Mars or survive an abort to Earth will depend: life support, attitude control, propulsion, communications and power generation are examples. While only life support might seem to require immediate response to anomalies, many other situations require on board response as well: losing attitude control during an aerobreaking maneuver, failures which need to be quickly safed, and loss of communications with Earth are all cases in point.

Once on the Martian surface, maximization of exploration becomes a focus in addition to safety. We do not yet have the resources to send crews of fifteen to Mars to run a Martian science outpost and support system. Hence crew involvement in routine operations such as controlling the life support system or maintaining rovers must be minimized and minor anomalies must be resolved locally rather than awaiting ground analysis. In addition, to maximize science return in this unknown environment, operations on Mars must be able to rapidly adapt to take advantage of new science opportunities or make the best of degraded capabilities.

These challenges to maintaining safety and productivity on Mars from Earth for several years are daunting when one considers the current state of mission operations. Current piloted missions rely upon near-instantaneous contact with hundreds of engineers and operators on the ground. In addition, recent attempts to teleoperate relatively simple systems for ninety days on Mars resulted in a considerable fraction of the mission being used to determine the state of the remote system and return it to productive operation, often over the course of a day or more [3].

The Reference Mission therefore explicitly calls for autonomous systems on Mars to allow unmanned systems to robustly prepare for human arrival, to protect crew and resources by rapidly responding to critical failures, to free explorers from routine operations and to control operations costs for this complex, multi-year mission. In this context, autonomy means the ability to correctly react to a wide range of circumstances, both usual and anomalous, without the need for direct human supervision. If available, a robust onboard autonomy capability would enable safer, more affordable missions to Mars by allowing complex systems such as life support systems or spacecraft attitude control systems to operate for extended periods of time without supervision over a wide range of nominal and anomalous operating conditions. The benefits would be increased safety and reduced downtime for mission critical systems, leverage of scarce human skills by automation of routine tasks, and reduced ground operations due to unattended recovery from anomalies and less detailed commanding requirements.

Currently, NASA's operational experience with the type of high capability, failure-tolerant autonomy described in the Reference Mission is low. To date, no fully automated power plants, life support, or cryogen plants have been deployed. Some automated planning and scheduling has been used to pre-compute command sequences for spacecraft and to schedule space shuttle refurbishment, but no deployed system has autonomously replanned its mission activities in the field. In addition, the robotic systems that have been deployed in space have been almost entirely dependent upon pre-computed command sequences relayed from Earth controllers, and have not been highly autonomous in the sense conveyed above.

Of course, every unmanned system sent into space has required some level of autonomy: if a spacecraft cannot at least point its antenna at Earth and wait for help after the expected kinds of failures, it is likely to be lost. Currently programmers and mission control operators use their commonsense understanding of hardware and mission goals to produce code and control sequences that will allow a spacecraft or other system to achieve some goal while allowing for some (usually very small) amount of uncertainty in the environment. This has the disadvantages of being relatively time intensive, error prone, and not particularly reusable. Because of the amount of analysis involved, such systems usually allow for uncertainty by being extremely conservative and provide the minimal amount of adaptability necessary to raise the likelihood of survival of the spacecraft. If an anomaly occurs the spacecraft or other system typically halts all activity, achieves a safe mode, and awaits further instructions. One notable exception is the attitude and articulation control system on the Cassini spacecraft, which represents the state of the art in deployed spacecraft autonomy [4] and which has not been replicated on the "faster, better, cheaper" missions which have followed.

The cost to develop highly robust autonomous control software and the ability of such systems to improve safety and productivity of assets deployed on Mars (or deep space or Europa for that matter) are significant risk factors that impact NASA's ability to accurately plan and scope future missions. One intent of the work described in the paper, model-based autonomy, is to demonstrate that highly robust autonomous systems can be developed more easily and more cheaply than the more modest systems which have been deployed to date.

What is model-based autonomy?

Model-based autonomy refers to the achievement of robust, autonomous operation through a growing set of reusable artificial intelligence (AI) reasoning algorithms that reason about first principles models of physical systems (e.g. spacecraft). In this context, a *model* is a logical or mathematical representation of a physical object or piece of software. A *first principles* model captures what is true about behavior or structure of the object (e.g. fluid flows through an open valve unless it is clogged). This is as opposed to traditional programs or rule-based expert systems, which capture what to do (e.g. turn on valves A, B, & C to start fuel flow) but unfortunately work only in certain implicit contexts (e.g. valves A, B, & C are working and A, B & C happen to control the fuel flow).

Since model-based autonomous systems do not contain an encoding of what to do in each situation, they must reason about the appropriate action to take or conclusion to draw based upon their models and the currently available information about the environment. The past few decades of AI research have produced reasoning engines that can plan a course of action to achieve a goal, identify the current state of a physical system, reconfigure that system to enable some function (e.g. make the engine thrust) and so on from a first principles model.

Reasoning directly from the model, the current observations of the world and the task at hand provides many large advantages over traditional software development. Not least among these are that the system is robust in uncertain environments since it was not hard coded to respond to certain situations, the models and inference engines can be reused, and the models explicitly capture the assumptions about the system that are being relied upon to control it.

3. Model-Based Autonomy at NASA Ames

The Autonomous Systems Group within the Computational Sciences Division at NASA Ames Research Center focuses on pursuing basic computer science research to increase the competence of autonomous systems and on providing basic autonomy technologies to NASA mission centers.

Many of the approximately twenty researchers within the group focus on model-based approaches to autonomy. As described above, a model-based autonomy architecture uses a declarative specification of a system's components and their interconnections to reason about the system as a whole and to provide general capabilities such as resource planning, execution monitoring, fault diagnosis and automated recovery. These compositional (plug-and-play) models adapt the generic architecture to a specific platform, enabling rapid development of autonomous control and health maintenance software for new systems.

Briefly, here is a small sampling of the many autonomy technologies being investigated within the Autonomous Systems Group that could be considered model-based. In Section 5, we will focus on one technology in slightly more detail.

♦ Model-based discrete controllers

  This line of research seeks to create a general engine for providing discrete control of a hardware and software system using only a declarative model of that system's components. This reasoning engine must infer the most likely state (referred to as a *mode*) of each component of the controlled system (including failures), accept a high level configuration goal (e.g. make the spacecraft produce thrust), and return a set of commands for the system's components that will achieve the configuration goal. The engine must use all available data to correctly handle sensor failure, multiple faults and novel recoveries.

  The Livingstone system for state identification and reconfiguration is such an engine, which has been developed at NASA Ames [5] and is being deployed in a number of NASA projects described below and in Section 6.

♦ Model-Based Decompositional Learning (MDL)

  This research uses a model of a system's structure to decompose complex parameter estimation problems into largely independent and solvable estimation subproblems. The Moriarty system is being developed at Ames to provide this capability and has been applied to developing a highly efficient office building environment controller [6].

♦ Planning & Scheduling

  Planning and scheduling research develops algorithms that accept a goal some system must achieve and emit a plan for causing the system to achieve the goal within the bounds of the resources allotted. The plan might be a simple sequence of commands or more generally it might be a partially ordered set of commands, a recurring schedule, or a plan with contingent branches which are executed based upon the outcome of previous actions. Unlike the simple configuration achievement of the discrete controller mentioned above, a planning and scheduling system must deliberate more thoroughly to achieve goals that involve oversubscribed resources, irreversible actions, or time-based constraints such as "take pictures of the following ten asteroids in the next hour".

  There are several planning and scheduling systems being pursued at NASA Ames, including the PS planner/scheduler [7]. Previous applications of NASA Ames planning technology include scheduling of observation requests onto an array of automated telescopes and scheduling of ground processing for the space shuttle.

The Livingstone engine, the PS planning and scheduling system and the Smart Executive plan execution system [8] have been combined to form the Remote Agent autonomy architecture. NASA Ames and the New Millennium Program at the Jet Propulsion Laboratory (JPL) developed the Remote Agent architecture and will flight-test it during a weeklong autonomous operations experiment on the New Millennium Deep Space 1 spacecraft. During this week, control of the spacecraft will be turned over to the Remote Agent system. PS will generate plans that achieve high-level mission goals specified by the ground controllers. The Smart Executive will decompose those partially ordered plans into a sequence of commands to the spacecraft and execute them. Livingstone will determine the state of the spacecraft at each command and notify the Smart Executive if any failures or unexpected results have occurred. If any such failures occur, Livingstone will be used to find a repair or workaround that allows the plan to continue execution. Simulated failures will also be injected for the purpose of testing, as one cannot rely on a spacecraft failing during a particular week. If the plan cannot be executed, the Smart Executive will inform the planner of the degraded capabilities of the spacecraft and ask for a new plan that still achieves the mission goals. Much more detail on the Remote Agent and its experiment on Deep Space 1 can be found in [9] and [10].

In addition to developing the core automated reasoning engines that enable model-based autonomy, NASA Ames is also pursing a number of research directions intended to further reduce the cost and effort required to develop and operate a model based autonomous system. Below two representative examples are described.

- Model-based Programming Environments

  This task provides engineer-friendly tools to visually develop the models used by the above reasoning engines, generate test suites from the model definitions, and enable collaboration during model development.

- Human Centered Autonomous Agents

  This task draws upon the lessons learned during development of Remote Agent. It seeks to develop methods to smooth collaboration between humans and autonomous systems, providing variable levels of autonomy and enabling cooperation between humans and autonomous systems. The intent is to minimize the need for human involvement in routine operations, but to avoid interfering with it when it is needed or simply desired. The goal is to allow a small team to interact with and direct one or more autonomous systems and to allow humans to quickly assimilate the situation should an autonomous system realize it is out of scope and request human intervention.

4. Benefits of Model-Based Autonomy

Our initial experience has been that model-based autonomy can drastically reduce cost compared to traditional software development while increasing robustness, safety and maintainability of the autonomous system. We believe this experience will carry through to additional NASA missions that use this type of technology to achieve high capability autonomy. We discuss a number of areas where model-based autonomy can provide benefits below.

Safety and Reliability

Model-based autonomous systems increase safety and efficiency primarily via timely and correct response to anomalies. They can detect failures as soon as or very shortly after they occur and automatically reason through system-wide ramifications. They can then immediately notify personnel or autonomously attempt to mitigate the effects of the problem. A rapid, well-reasoned response minimizes the impact of failures or unforeseen scenarios.

Reliability is also increased by raising the visibility of the mapping between the control system and the apparatus it is meant to control. For example, with hard-coded fault protection designs knowledge about the controlled system is implicit rather than explicit. This means that we require the fault protection software developers to understand the system, understand the system-wide ramifications of various symptoms that might arise and actions that might be taken in response (taking into account the system might have experienced previous failures by the time the current problem occurs). The results of that understanding must then be encoded in a low-level procedural programming language. If the system is modified, as is often the case with one-of-a-kind hardware, that mapping from understanding to procedural code must be reconstructed and the necessary modifications to the code made, a likely time for the insertion of errors. Under these conditions even developing a fault protection system which has been scaled back to the minimal essentials can be a challenging task.

In contrast, with model-based fault diagnosis the fault protection software engineers explicitly model how the system behaves in nominal and (if known) failure cases. Relevant assumptions about the behavior and structure of the controlled system are explicitly stated in a declarative model, which is easily inspected and understood. Appropriate behavior is generated by operation of the heavily tested inference algorithms on the explicit models, not by low level statements of standard procedural software codes, which have an implicit and tenuous correspondence to written requirements or the software developer's evolving mental model of the system.

Finally, an approach which features reuse of reasoning engines and component models over multiple mission elements benefits reliability, cost and capability in the same manner that standard engines for database or graphics functionality benefit other software developers. Cost is reduced as development of the autonomy capability is amortized over several applications. Reliability is increased as the architecture's inference engines are invoked thousands of times before flight and as testing aggregates over multiple applications of the system. Capability is increased as reasoning engine improvements made and verified for one application are available for use in subsequent applications.

Development Effort

Model-based systems have proven extremely easy to design, implement and maintain. Applications consist of explicit, easily acquired “common-sense” level models of the controlled system. Gone is the need to develop procedural code that mingles implicit system models with implementation details, complicating development and maintenance. There is also no need to constantly reason through the interactions of the system’s many components in order maintain code or rules for every possible scenario or spacecraft configuration.

Declarative, high level models of the structure and behavior of relevant device in the system can be quickly developed, unit tested and plugged together. Appropriate system-wide behavior is then generated. Device models capture only information about the local device, and not procedures for controlling or recovering the device within the context of some specific spacecraft or system. This makes them both reusable to model different systems and easily modified or replaced as a specific system being modeled is revised.

As the models are declarative and contain only first principles knowledge rather than context-dependent procedures or rules, the model of a single system can also be reused in a number of contexts. In the Remote Agent experiment, a single set of device models is used to monitor execution of commands, diagnose failures, provide recoveries, generate discrete event simulators and automatically produce descriptions of the spacecraft. Reuse could be taken much farther than time allowed during the Remote Agent experiment. One could use the same model to compile a static fault tree, compile interface definitions for communicating with the Remote Agent, and so on. In addition by historical accident the high level planning and scheduling component of Remote Agent did not accept the same model as was used for all of the above tasks, but an effort is underway to unify the two languages before Remote Agent is for additional mission operations.

Operations Cost

Cost goals demand that mission planning be streamlined and mission control intervention in routine spacecraft and Mars surface operations are minimized. Systems that can autonomously monitor, adjust and repair themselves, even in the face of novel situations, obviously decrease the time the ground controllers or crew must devote to such tasks. In addition, when human involvement is required or desired, our goal is to drastically reduce operator effort, allowing an operator to control more systems or a crewmember to operate a system more quickly or with less detailed training. Model-based autonomy allows very high level interaction. The current state of the system can be presented in hierarchical diagrams generated from the models. Commands can be goals to be achieved rather than detailed instructions. Significantly, when a user wants to interact with an model-based system, its explicit internal representation allows it to explain why it has made certain inferences, why it decided to take certain actions or what it would do in a hypothetical situation. These features combine to make operating a model-based system potentially far more intuitive, cooperative and efficient than current spacecraft operations.

5. LIVINGSTONE

As mentioned in Section 3, Livingstone is a model-based discrete controller. Its function is to infer the current state (mode) of each relevant device making up the system being controlled and to recommend actions that can reconfigure the system so that it achieves the currently desired configuration goals, if possible. In practice, these configuration goals could be provided by a human operating some apparatus by issuing high level configuration commands, or by some automated system such as the Smart Executive (Exec) mentioned above, which decomposes a high level plan into a series of configuration goals to be achieved. Purely for the sake of the discussion below, we will assume the Exec is providing the configuration goals and that the system being controlled is a spacecraft.

To track the modes of system devices, Livingstone eavesdrops on commands that are sent to the spacecraft hardware by the Exec. As each command is executed, Livingstone receives observations from spacecraft’s sensors, abstracted by monitors in the real time control software for the Attitude Control Subsystem (ACS), communications bus, or whatever hardware is present. Livingstone combines these commands and observations with declarative models of the spacecraft components to determine the current state of the system and report it to the Exec. A pathologically simple example is shown schematically in Figure 1. In the nominal case, Livingstone merely confirms that the commands had the expected effect on spacecraft state. In case of failure, Livingstone diagnoses the failure and the current state of the spacecraft and provides a recovery recommendation. A single set of models and algorithms are exploited for command confirmation, diagnosis and recovery.

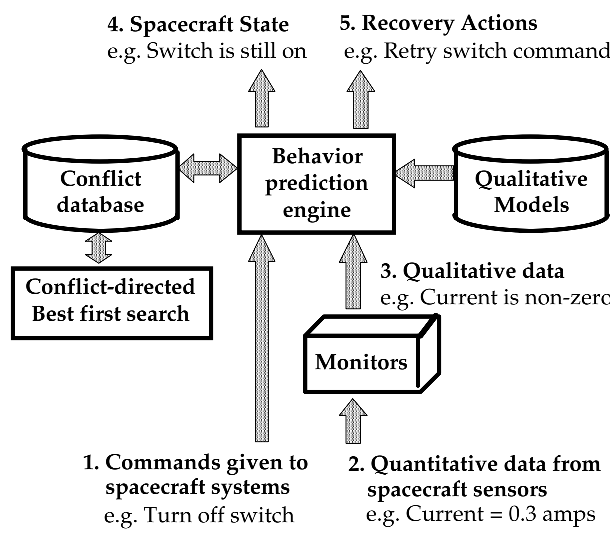

Figure 1. Information Flow in Livingstone

The capabilities of the Livingstone inference engine can be divided into two parts: mode identification (MI) and mode reconfiguration (MR). MI is responsible for identifying the current operating or failure mode of each component in the spacecraft. Following a component failure, MR is responsible for suggesting reconfiguration actions that restore the spacecraft to a configuration that achieves all current configuration goals required by the Exec. Livingstone can be viewed as a discrete model-based controller in which MI provides the sensing component and MR provides the actuation component. MI's mode inference allows the Exec to reason about the state of the spacecraft in terms of component modes or even high level capabilities such as "able to produce thrust" rather than in terms of low level sensor values. MR supports the run-time generation of novel reconfiguration actions to return components to the desired mode or to re-enable high level capabilities such as "able to produce thrust".

Livingstone uses algorithms adapted from model-based diagnosis [11, 12] to provide the above functions. The key idea underlying model-based diagnosis is that a combination of component modes is a possible description of the current state of the spacecraft only if the set of models associated with these modes is consistent with the observed sensor values. Following de Kleer and Williams [13], MI uses a conflict directed best-first search to find the most likely combination of component modes consistent with the observations. Analogously, MR uses the same search to find the least-cost combination of commands that achieve the desired goals in the next state. Furthermore, both MI and MR use the same system model to perform their function. The combination of a single search algorithm with a single model, and the process of exercising these through multiple uses, contributes significantly to the robustness of the complete system. Note that this methodology is independent of the actual set of available sensors and commands. Furthermore, it does not require that all aspects of the spacecraft state are directly observable, providing an elegant solution to the problem of limited observability.

The use of model-based diagnosis algorithms immediately provides Livingstone with a number of additional features. First, the search algorithms are sound and complete, providing a guarantee of coverage with respect to the models used. Second, the model building methodology is modular, which simplifies model construction and maintenance, and supports reuse. Third, the algorithms extend smoothly to handling multiple faults and recoveries that involve multiple commands. Fourth, while the algorithms do not require explicit fault models for each component, they can easily exploit available fault models to find likely failures and possible recoveries.

Livingstone extends the basic ideas of model-based diagnosis by modeling each component as a finite state machine, and the whole spacecraft as a set of concurrent, synchronous state machines. Modeling the spacecraft as a concurrent machine allows Livingstone to effectively track concurrent state changes caused either by deliberate command or by component failures. An important feature is that the behavior of each component state or mode is captured using abstract, or qualitative, models [14]. These models describe qualities of the spacecraft's structure or behavior without the detail needed for precise numerical prediction, making abstract models much easier to acquire and verify than quantitative engineering models. Examples of qualities captured are the power, data and hydraulic connectivity of spacecraft components and the directions in which each thruster provides torque. While such models cannot quantify how the spacecraft would perform with a failed thruster for example, they can be used to infer which thrusters are failed given only the signs of the errors in spacecraft orientation. Such inferences are robust since small changes in the underlying parameters do not affect the abstract behavior of the spacecraft. In addition, abstract models can be reduced to a set of clauses in propositional logic. This form allows behavior prediction to take place via unit propagation, a restricted and very efficient inference procedure.

It is important to note that the Livingstone models are not required to be explicit or complete with respect to the actual physical components. Often models do not explicitly represent the cause for a given behavior in terms of a component's physical structure. For example, there are numerous causes for a stuck switch: the driver has failed, excessive current has welded it shut, and so on. If the observable behavior and recovery for all causes of a stuck switch are the same, Livingstone need not closely model the physical structure responsible for these fine distinctions. Models are always incomplete in that they have an explicit unknown failure mode. Any component behavior that is inconsistent with all known nominal and failure modes is consistent with the unknown failure mode. In this way, Livingstone can still infer that a component has failed, though the failure was not foreseen or was simply left unmodeled because no recovery is possible. By modeling only to the level of detail required to make relevant distinctions in diagnosis (distinctions that prescribe different recoveries or different operation of the system) we can describe a system with qualitative "common-sense" models which are compact and quite easily written.

6. MARS RELATED APPLICATIONS

The intent behind model-based autonomy is to create generic, high capability reasoning systems that can be adapted to a wide range of applications simply by writing the appropriate models. As such, model-based autonomy might be able to contribute to the control of a variety of elements of a piloted Mars mission. In this early stage of Mars mission definition, model-based autonomy is involved in the prototyping of a number of specific mission elements.

Closed-Loop Ecological Life Support Systems (CELSS)

In order to transport and support humans for Mars expeditions, NASA's Human Exploration and Development of Space (HEDS) requirements state a need for autonomous operation of life support, ISRU and transport equipment. During a Mars expedition, autonomous plant operations would allow unmanned systems to prepare for human arrival, protect crew and resources by rapidly responding to critical failures, and free humans from routine operations, allowing greater exploration.

At NASA's Johnson Space Center (JSC), a closed loop life support testbed called Bioplex has been constructed. The Bioplex consists of three sections: a three story cylindrical living quarters similar to the Mars habitats discussed in various mission proposals; a plant chamber where wheat is grown to provide food and exchange CO_2 for O_2; and an incinerator chamber used to eliminate solid waste and produce CO_2. The most recent Bioplex testing is referred to as the Product Gas Transfer phase as it concentrates on generation and distribution of product gases (CO_2 from the crew and incinerator and O_2 from the plants) and does not yet address issues such as waste water recycling or power management.

A JSC advanced development group has developed an autonomous control system to operate the product gas transfer phase of Bioplex [15]. This system, based upon the 3T autonomy architecture [16], maintains the appropriate atmosphere in each chamber by extracting and storing product gases and coordinating activities such as firing the incinerator or opening the plant chamber for human access. The system successfully controlled gas transfer during test in which a human crew inhabited the Bioplex for ninety days. It was not expected, however, to maintain operation in the face of failures, though many would likely occur over a 4-year mission.

We are currently working to integrate the Livingstone mode identification and reconfiguration engine with JSC's 3T architecture, adding to it the ability to determine the current state of the testbed and respond to anomalous situations or failures by performing high level, system-wide reasoning. This will result in a single, reusable architecture which maintains the best possible operation of a regenerative life support system and other complex physical plants during both nominal operation and failures, somewhat analogous to the autonomic and immune functions of a living organism.

We intend to demonstrate the combined system by maintaining operation of the testbed over an extended test period and providing both fully autonomous and human-centered operation. To test the system, an outside examiner will be employed to introduce failures into the testbed as desired which the system will diagnose and attempt to mitigate.

The second goal is to demonstrate and extend the ability of model-based systems to reduce analysis, development and operations costs. The testbed application will be rapidly developed with tools that could be used to develop mission applications. Users will develop and operate the testbed by manipulating explicit models with visual tools. If previous experience is to be believed, far less effort will be required to develop, understand and revise the system than in an approach where system model is implicit but still must be maintained.

If successful, this demonstration will increase the likelihood that autonomy technologies being developed by NASA are appropriate and sufficiently mature when they are required for HEDS missions to Mars and other destinations. It will also ensure that the necessary technologies can be integrated and will identify needed extensions before such shortcomings could impact the critical path of a mission. In addition, JSC will have a prototype of a reusable, fault-tolerant, high-capability autonomous control system and the expertise to apply this system to a flight experiment or mission. This could be applied to any complex physical system that must be controlled and maintained over an extended period of time such as spacecraft, power plants, ISRU machinery, and autonomous or semiautonomous surface vehicles.

In-situ Resource Utilization

In-situ resource utilization, or "living off the land", is critical to making a piloted Mars mission robust and affordable [1]. More specifically, it is envisioned that in-situ propellant production (ISPP) plants will arrive on Mars years before humans and begin combining hydrogen brought from Earth with CO_2 from the Martian atmosphere to create methane. This fuel will power the ascent vehicle that will lift the crew off Mars to begin their trip home in addition to powering any methane-fueled surface vehicles the astronauts might possess.

Though the chemical reactions involved are conceptually quite simple, on Mars they are somewhat complicated by issues such as the low atmospheric pressure and slow contamination of the ISPP catalysts by trace elements in the Martian atmosphere. To ensure that adequate ISPP capability is available for future Mars missions, NASA has begun to explore ISPP designs and build prototype hardware for operation in Mars-like test chambers. Both JSC and NASA Kennedy Space Center (KSC) are involved in early ISPP development, and the KSC team is integrating Livingstone into their ISPP prototyping efforts.

The short-term focus of this collaboration is to integrate Livingstone's ability to diagnose and mitigate failures with existing KSC model-based technology to gain experience with a model-based monitoring, diagnosis and recovery system for ISPP. A secondary short-term goal is to determine if any other autonomy technology previously invested in by NASA, for example the Smart Executive, can be reused on the ISPP testbed, thus increasing capability without greatly increasing cost.

A longer term goal is to continue research into control of physical systems which must continuously adjust their operation to unforeseen degradation in capability (for example an ISPP unit where Martian dust covers solar panels or slowly clogs air filters) rather than taking a discrete recovery action as Livingstone does. Related issues include reasoning about hybrid discrete/continuous systems, predictive diagnosis and relearning models of the continuous dynamic behavior of the system. This research should contribute to development of ISPP and other systems that are robust and yet run at the ragged edge of optimality throughout their lifetimes, neither being overly conservative nor exceeding their remaining degraded capabilities.

Autonomous Rovers

The Remote Agent system, described above and consisting of a planner, a smart execution system and Livingstone, is being adapted for use on the NASA Ames Marsokhod rover as part of an effort to demonstrate increased rover autonomy. That effort is described in [17].

7. ACKNOWLEDGEMENTS

This paper touches on the work of a great many people too numerous to name here. The Autonomous Systems Group at NASA Ames Research Center consists of about twenty computer science researchers pursuing all manners of autonomy research, much of which was not mentioned here. Members of the JPL New Millennium Program and AI Group contributed to the Remote Agent architecture and to making it work on a flight platform. Advanced development groups at NASA JSC (3T and Bioplex PGT), NASA KSC (ISPP and KATE) and JPL (space based interferometry) have shared their expertise with us and are helping to push the model-based autonomy technologies described here forward.

REFERENCES

Many of the following papers may be found on the World Wide Web at http://ic-www.arc.nasa.gov/ic/projects/mba/

[1] R. Zubrin and R. Wagner. The case for Mars: The plan to settle the Red Planet and why we must. The Free Press, 1996.

[2] S. J. Hoffman and D. I. Kaplan, editors. Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team. NASA Special Publication 6107. July 1997.

[3] A. H. Mishkin, J. C. Morrison, T. T. Nguyen, H. W. Stone, B. K. Cooper and B. H. Wilcox. Experiences with operations and autonomy of the Mars Pathfinder microrover. In Proceedings of the IEEE Aerospace Conference, Snowmass, CO 1998.

[4] G. M. Brown, D. E. Bernard and R. D. Rasmussen. Attitude and articulation control for the Cassini Spacecraft: A fault tolerance overview. In 14th AIAA/IEEE Digital Avionics Systems Conference, Cambridge, MA, November 1995.

[5] B. C. Williams and P. Nayak, A Model-based Approach to Reactive Self-Configuring Systems, Proceedings of AAAI-96, 1996.

[6] B C. Williams and B. Millar. 1996. Automated Decomposition of Model-based Learning Problems. In Proceedings of QR-96.

[7] N. Muscettola, B. Smith, C. Fry, S. Chien, K. Rajan , G Rabideau and D. Yan, Onboard Planning For New Millenium Deep Space One Autonomy, Proceedings of IEEE Aerospace Conference, 1997

[8] B. Pell, E. Gat, R. Keesing, N. Muscettola, and B. Smith. Robust periodic planning and execution for autonomous spacecraft.

[9] B. Pell, D. E. Bernard, S. A. Chien, E. Gat, N. Muscettola, P. P. Nayak, M. D. Wagner, and B. C. Williams, An Autonomous Spacecraft Agent Prototype, Proceedings of the First International Conference on Autonomous Agents, 1997.

[10] D. E. Bernard et Al. Design of the Remote Agent Experiment for Spacecraft Autonomy. Proceedings of IEEE Aero-98.

[11] J. de Kleer and B. C. Williams, Diagnosing Multiple Faults, Artificial Intelligence, Vol 32, Number 1, 1987.

[12] J. de Kleer and B. C. Williams, Diagnosis With Behavioral Modes, Proceedings of IJCAI-89, 1989.

[13] J. de Kleer and B. C. Williams, *Artificial Intelligence*, Volume 51, Elsevier, 1991.

[14] . S. Weld and J. de Kleer, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[15] D.Schreckenghost, M. Edeen, R. P. Bonasso, and J. Erickson. Intelligent control of product gas transfer for air revitalization.. Abstract submitted for 28th International Conference on Environmental Systems (ICES), July 1998.

[16] R. P. Bonasso , R.J. Firby, E. Gat, D. Kortenkamp, D. Miller and M. Slack. Experiences with an architecture for intelligent, reactive agents. In Journal of Experimental and Theoretical AI, 1997.

[17] J. Bresina, G. A. Dorais, K. Golden, D. E. Smith, R. Washington, Autonomous Rovers for Human Exploration of Mars. Proceedings of the First Annual Mars Society Conference. Boulder, CO, August 1998. To Appear.

[18] B. C. Williams and P. P. Nayak. Immobile Robots: AI in the New Millennium. In AI Magazine, Fall 1996.

[19] B. C. Williams and P. P. Nayak. A Reactive Planner for a Model-based Executive. In Proceedings of IJCAI-97.

[20] N. Muscettola. HSTS: Integrating planning and scheduling. In Mark Fox and Monte Zweben, editors, Intelligent Scheduling. Morgan Kaufmann, 1994.

[21] V. Gupta, R. Jagadeesan, V. Saraswat. Computing with Continuous Change. Science of Computer Programming, 1997.