What is next in Autonomous Control Techniques?

María Dolores R-Moreno
European Space Research and Technology Centre (ESTEC)
European Space Agency
and
Departamento de Automática.
Universidad de Alcalá.
Ctra. Madrid-Barcelona, km 33,6.
28805 Alcalá de Henares (Madrid), Spain.
e-mail: mdolores@aut.uah.es

James Kurien
NASA Ames Research Center
MS 269-3, Moffett Field, CA 94035
e-mail: James.A.Kurien@nasa.gov

Abstract

Spacecraft domains have become a popular area of AI research due to the excitement created by the discoveries of recent missions, for example on our neighboring planet Mars, and the ever-increasing capability of spacecraft. With the current US, Japan and European focus on launching spacecraft for exploration, communication, broadcasting, or localization tasks, among others, the automatic control of these machines becomes an important problem. In this paper, we compare recent directions in AI research with practice in current missions, using planning and scheduling, intelligent execution and fault protection as examples. We attempt to synthesize trends in both spacecraft operations and the use of AI technology, and suggest how they might impact future research and its adoption by missions.

Introduction

For all of our history, humans have shown an interest in exploring, discovering and knowing what lies beyond Earth's boundaries. Many individuals, known and unknown, have labored often with very few resources to reveal some of the secrets of our solar system such as the distance to the sun or the to the moon, the orbits and size of the planets, and so on. The era of active space exploration started in the 1960's due to the aggressive competition between the USSR and USA to achieve a landing on the Moon. Although a return to the Moon is in the center of our sights, it is the red planet, Mars, which is now focusing our attention.

Future missions such as NASA's Mars Science Laboratory or ESA's ExoMars require control systems that can manage the severe resource constraints of these missions. Advances in AI technology have the potential to help spacecraft such as the use of resources efficiently by responding directly to their environments, to make spacecraft less expensive to operate, and to increase science return. It's therefore interesting to consider how and why some AI technologies have been applied to missions in a straightforward fashion, some have been applied in unexpected ways, and some are still struggling to gain traction.

Planning, scheduling, sequence execution and fault protection are at once operations phases that are routinely carried out for spacecraft and active topics of AI research. By comparing and contrasting standard AI techniques in these areas with the state of the practice on spacecraft missions, we hope to provide insight into what kinds of infusion of AI technology into missions has a high probability of success, and some help in predicting what future areas research might be adopted by missions.

This paper is structured as follows. First, we consider planning and scheduling of activities. Planning is the problem of selecting activities such that they achieve one or more goals and satisfy a set of domain constraints. Scheduling addressing the problem of organizing activities in time such that the resources during each activities execution are available. Typically, activities or commands carried out by a spacecraft are planned and scheduled in advance. We discuss one flight experiment and one operational system where AI Planning and Scheduling (P&S) was used on-board to allow spacecraft to autonomously select activities to respond local conditions or science opportunities. We also discuss the use of P&S technology in mission control centers and the direction the technology has taken in order to work with human planners rather than autonomously.

We next consider plan or sequence execution. All space missions require execution systems that execute commands and monitor the result. To enable the most efficient execution of plans and schedules, research has focused upon explicitly modeling scheduling assumptions during execution and rescheduling the primary activities while they are executed whenever a broken scheduling assumption is detected. We consider a flight experiment that took this approach, and discuss a number of operational systems that use time and resource margin to enable a simpler execution semantics. We discuss the recurring theme of trading theoretical peak execution efficiency for simpler operations software, some very reasonable motivations for taking this tradeoff, and how research might respond.

Fault protection is an engineering process that incorporates robustness to faults into spacecraft hardware, software, systems engineering and operations. We discuss the onboard fault protection software on a recent spacecraft as well as spacecraft flight experiments that incorporated model-based diagnosis technology. Model-based diagnosis systems are based on physical parametric constraints where a generic software engine or set of principles is developed in the hope of addressing a large class of diagnosis problems. We discuss some of the cost/benefit issues we see with model-based diagnosis as a possible explanation for its lack of adoption.

For each of these areas, we conclude by summarizing the trends we have observed in the missions and technologies we surveyed.

AI Planning and Scheduling Systems

AI Planning and Scheduling (P&S) are two related areas that initially evolved separately. **AI Planning** is a search problem that selects activities such that they achieve one or more goals and satisfy a set of domain constraints. **AI Scheduling** addresses the problem of organizing tasks in time. Given a set of activities, a scheduler must assign start times and required resources to the activities, obeying the temporal restrictions on the activities and the capacity limitations of the resources. In the remainder of this section, we briefly describe common approaches to P&S from the literature. Then, we survey systems that have been used in spacecraft operations. These systems typically perform both P&S and, as in most applications the choice of actions to achieve a goal is intimately tied to time constraints and what resources are available. Finally, we discuss what we believe to be trends in applications of P&S technology to spacecraft operations.

Among the many planning techniques developed are:

- Partial Order Causal Link planners (Weld 1994) search through the plan space. These planners must perform constraints satisfaction to ensure the consistency of the order constraints. In order to avoid interferences between actions, dependencies are recorded in a data structure called causal links. Causal links must be checked periodically during the planning process to make sure that no other action can threaten them. In case a plan contains a threat, some additional ordering constraints can be added to avoid it.
- GRAPHPLAN-based planners search through a plan graph. They alternate between graph expansion and solution extraction. The graph expansion extends the plan graphs forward until it has achieved a necessary condition for plan existence. The solution extraction phase performs a backward-chaining search on the graph, looking for a plan that solves the problem. If no solution can be found, the cycle repeats expanding the planning graph. The basic idea is to perform a kind of reachability analysis to rule out many of the combinations and set of actions that are not compatible.
- Heuristic Search Planners (HSP) transform planning problems into problems of heuristic search by automatically extracting heuristics functions *h* from STRIPS encoding instead of introducing them manually (Bonet & Geffner 1999). It uses a declarative language for stating problems and a general mechanism for extracting heuristics from these representations. The same code is then able to process problems from different domains.
- A SAT-based planner takes a planning problem as an input, guesses a plan length and generates a set of propositional clauses (CNF) that are checked for satisfiability. After the translation is performed, fast simplification algorithms as unit propagation and pure literal elimination are used to shrink the CNF formula. For satisfying assignment, a SAT solver can use Systematic (Liberatore 2000), Stochastic (Selman, Levesque, & Mitchell 1996) or Incremental (McAllester 1990) methods.
- HTN planners try to use the knowledge available of the domain to solve the planning problem. This knowledge can be obtained using additional task and goal structures, search control techniques, interaction with humans or different type of constraints. A method maps a task into a partially ordered network of tasks, together with a set of constraints. The basic algorithm is to expand tasks and resolve conflicts iteratively, until a conflict-free plan can be found that consists only of primitive tasks. The task network may contain conflicts caused by the interaction among tasks: after each reduction, a set of critics is checked so as to recognize and solve interactions between this and any other reductions. Thus critics provide a general mechanism for detecting interactions early, so as to reduce the amount of backtracking.
- Probabilistic planners use probabilities to represent and reason about uncertainty in the planning domain. That is, when an action is selected the probabilities of the possible outcomes are evaluated. Additional actions may be added to construct plans that are likely to succeed even individual actions are uncertain. (Blythe 1999).
- A Markov Decision Process (MDP) expands the notion of uncertainty. It is defined by an initial probability distribution over all possible states of the system being modeled, a distribution that represents the likelihood of the system transitioning from the current state to each other state given an action, and the reward for taking each action in each state. Solving an MDP requires determining the action in each state that maximizes the expected cumulative reward of the system over time. Techniques based on policy iteration or value iteration (Puterman 1994) have been traditionally used. This can be intractable for most applications, so most of the work in this area has focused on using an approximation or abstraction of the state space.

For solving scheduling problems, we can consider two main approaches:

- The Operational Research (OR) area. The most representative techniques are Linear Programming and Sensitivity Analysis. This two techniques allow to find the optimal solution of a function that satisfies a set of linear constraints, in the first case or the consequences in the solution if some changes are made in either the data or in some of the solution values, in the second case.
- Constraint Satisfaction Technology (CSP). A Constraint Satisfaction Problem (CSP) prescribes some requirements for a finite number of variables in the form of constraints. The set of possible values, that is, the domain for each variable is finite. Each constraint specifies a relationship between the values of the variables that most hold. Solving a (CSP) means finding an assignment to each variable such that all of the constraints are satisfied. Most of the

algorithms used for solving a CSP fall into these two categories:

- Refinement or Constructive Search Methods: a CSP that progresses *incrementally* assigning values to variables, checking the constraints and backtracking when violations appear. The evaluation of global criteria can only be approximated given that there is only a partial schedule.
- Iterative Repair or Local Search Methods: they start with a complete assignment of values to variables and then reassign new values to variables to resolve violated constraints. The evaluation of global criteria is evaluated with low cost. However, one of the main disadvantages is that they can suffer from local minima and they are often incomplete.

Applications of AI Planning and Scheduling

Table 1 lists a representative set of P&S technologies and the missions for which they have been in experimental or operational use.

DEVISER (Vere 1983) was used to develop plans for the two Voyager spacecraft which photographed Jupiter, Saturn and the outer planets. It is a general purpose planner that generates plans by backward chaining from unordered subgoals. Unlike simple STRIPS planners, goals may have time restrictions on when they are achieved and how long they should be maintained. For each activity, a duration and a start time *window* are presented. The output is a partially ordered network of activities. Partially ordered Deviser plans were hand editing to produced fully scheduled sequences that were executed on Voyager.

In a few cases, planning software has been run on-board the spacecraft, with intent that the spacecraft could respond to events by generating a new plan of action without waiting for communication with operators on the ground. HSTS was used in an experimental mode on-board the Deep Space 1 (DS-1) spacecraft, generating three plans that were autonomously executed on board the spacecraft as a part of the Remote Agent experiment (Muscettola *et al.* 1998; Muscettola & Smith 1997). This experiment consisting of a 20 hours scenario and a 6 hours scenario was the first time an AI planner generated a plan on-board a NASA spacecraft and also re-planned in order to accommodate a simulated failure. Rather than generating a plan on the ground that is converted to spacecraft commands, HSTS generated a partially ordered, flexible time network of activities which was dynamically converted into spacecraft commands by an execution system (Pell *et al.* 1997). The domain and goals were modeled in the DDL language. A Timeline based representation where the values of state variables evolve over time. Constraints between state variables are set with compatibilities. The state variable approach differs from the predicate based representation used in STRIPS or PDDL where in DDL time and constraints are represented and treated separately.

The Continuous Activity Scheduling Planning Execution and Replanning (CASPER) system has been applied to a number of domains, including highly successful, operational use on-board the Earth Observer 1 spacecraft, (EO-1) activated in January of 2004 (Rabideau *et al.* 2006; Chien *et al.* 2000). CASPER is similar to HSTS in that they both represent activities on timelines and use consistency with a domain model to determine if a plan is valid. How they are used is very different. HSTS generates a plan to meet a set of goals and then exits. If execution of the plan fails, HSTS starts a new planning problem. CASPER is used in an iterative plan repair fashion. As a plan is executed, CASPER is fed information about the state of the spacecraft and continuously checks it against what is expected from the plan. If the actual state and the remaining plan are not consistent, CASPER repairs the plan. Repairs may include adding, deleting or rescheduling actions, adding temporal constraints, or changing activity parameters. New goals are handled in the same way; CASPER attempts to make repairs to the plan so that it is consistent with achieving the modified set of goals for the plan. On EO-1, the ability to smoothly add additional goals is used to perform autonomous science re-targeting. On-board algorithms scan images of the Earth for rapid changes involving cloud cover, flooding, or thermal emissions from fires or volcanoes. When a promising candidate is detected, goals to image the area are added to the plan and CASPER attempts to repair the plan to accommodate them. CASPER has become the primary mission operations tool and continues to operate as of this writing. To date, it has processed 107758 goals, and planned 13528 images, including 1330 triggered by on-board science processing.

Smart-1 (Camino *et al.* 2005) belongs to an ESA Small Mission for Advance Research and Technology. One of the aspects they want to consider was to reduce the workload on ground, by mean of the Mission Planning System (MPS). The purpose of the MPS is to ensure the consistency of the operations request against a variety of operational and spacecraft resource constraints, such as power, storage, downlink capacity, prior to generating the required telecommand stack for uplink. The functionality of the MPS goes beyond payload scheduling by integrating operations defined by Flight Dynamics, the Station Scheduling Office and the Flight Control Team into a single, consistent mission plan.

The EUROPA planner (Frank & Jonsson 2003) is used in such a mixed-initiative mode on the Mars Exploration Rover (MER) mission (Bresina *et al.* 2005). EUROPA is a framework for representing and solving CSPs, with an emphasis on temporal constraint networks, which is descended from HSTS. For the MER mission, the EUROPA planning engine was used within an end-user planning application called MAPGEN (Ai-Chang *et al.* 2004). MAPGEN includes numerous additions and control strategies to the basic planner to adapt it to a mixed-initiative role. For example, a key to dealing with oversubscription and the negotiation process was enabling the user to incrementally plan. The user requests a plan for a set of goals, understands the resulting plan and potentially makes adjustments to it, then adds additional goals. The planner then returns a plan for all of the goals, or rejects the new goals as unsatisfiable. This in turn was made possible by the development of a minimum perturbation heuristic, which specifies that given an existing plan and an additional goal, the plan to achieve all of the goals of the existing plan plus the new goal should resem-

System	Mission	Year	Used	Comments
DEVISER	Voyager	1977	Ground	Users edit Deviser plan into final sequence
PLANIT-II	Galileo	1995	Ground	Plan careful data mgmt due to antenna fault
	Mars Pathfinder	1997	Ground	Mars surface ops
	Spitzer space telescope	2003	Ground	Integrated with Hubble-heritage schedulers
HSTS	Deep Space 1	1998	On Board	Short experiment. First on board planner.
ASPEN	AMM-2	2000	Ground	Plan campaigns for mapping satellite
APSEN & CASPER	Earth Orbiter 1	2003	On Board	On-board replan for science opportunities
MISSION PLANNING SYS	Smart-1	2003	Ground	
EUROPA/MAPGEN	Mars Exp. Rovers	2003	Ground	Mixed initiative, min-perturb heuristic
MEXAR-2	Mars-Express	2005	Ground	
EUROPA 2/ENSEMBLE	Phoenix Mars Lander	2007	Ground	Manipulate infeasible plans,
	Mars Science Lab	2009	Ground	domain model is generated
PROBA	Proba	2001	On Board	Energy and Memory management

Table 1: Selected missions with operational or experimental use of P&S technology

ble the existing plan as closely as possible. This prevented what users less-than-affectionately termed "the blender effect" where, after one had just spent a considerable amount of time understanding and adjusting a plan, adding a simple and seemingly unrelated goal would result in a completely different plan due to due to arbitrary choices in the search mechanism inside the planner. With these mixed-initiative enhancements, EUROPA and MAPGEN have generated thousands of plans for the highly successful MER rovers, and continue to do so as of this writing.

The Phoenix Mars Lander and Mars Science Laboratory missions use the EUROPA 2 planning engine within the ENSEMBLE tactical planning system (Bresina & Morris 2006). EUROPA 2 is a high performance planning system descended from EUROPA that uses the same style of constraint and interval-based plan generation. ENSEMBLE in turn builds on the progress and lessons learned via employing MAPGEN and other MER tools for tactical rover planning. Of the many enhancements in the evolution, one of the most significant is the ability to work with infeasible plans. That is to say, rather than rejecting goals that cannot be met, the planner simply notes the way in which the unsatisfied goals (rover activities in this case) are inconsistent with the domain model (Morris & Bresina 2008). Intuitively, the planner might report that two activities are attempting to move the rover's arm at once, or that communication with Earth, which requires the rover to be stationary, is constrained to be at the same time as a rover drive. This is a significant improvement over goal rejection aimed at the Phoenix mission requirement that the planning system "shall not fight with the user". First, it allows the user to understand why the desired set of goals cannot all be met at once, then either choose goals to remove or ask Europa 2 to repair the plan. Second, it allows the user to ignore changes the planner would like to make to the plan. EUROPA 2 and the ENSEMBLE system have been delivered to the PHOENIX mission and are awaiting the landing in May 2008. Deliveries for MSL will continue through launch in 2009.

The Mars-Express Scheduling Architecture (Mexar2) (Cesta *et al.* 2007), is an AI-based ground tool in daily use on the ESA Mars-Express mission since February 2005. It solves the memory-dumping problem that consists of planning sequences of dump activities which specify the packet store and the amount of data to download in any available time interval.

ESA's PROBA (Creasy & Teston 2001) is a micro-satellite built with a double goal: space environment investigation and Earth observation, and on-board operational autonomy. It performs autonomous guidance, navigation, control, on board scheduling and payload resources management. Its payload includes a compact multi-spectral imager and high-resolution camera. It predicts on board energy and memory resources using algorithms from the Operational Research (OR) area.

Possible Trends

One recurring theme is the need for planning software and users to generate plans in a mixed-initiative (cooperative) fashion. Traditionally, the planning problem is posed as accepting a set of goals and automatically generating a feasible plan In many cases, this breaks down for several reasons. First, the initial goals often oversubscribe the spacecraft's resources. Users need to explore and negotiate which goals to remove rather than having the planner decide. Second, it's important for the user to examine infeasible plans, view plan flaws, and determine which activities to remove or what other repair strategy to consider. Third, during operations domain modeling bugs are discovered and mission managers decide that in certain cases a constraint in the domain model should be ignored. Thus the planner must be able to work with infeasible plans and users must be able to ignore changes or advice the planner is providing, investigate plan flaws, and quickly add new operating rules or relax others for special circumstances.

We also believe scheduling is the bigger issue than planning in the traditional sense. To achieve a goal, there are not typically lengthy sequences of actions with many alternatives. Instead, there are many independent goals that are easy to achieve individually, but interact with each other in complex ways through the resources limitations and operating constraints.

Finally, the ability of spacecraft to accommodate branches in execution often exceeds the ability for users to evaluate their impact to resources and constraints. Helping operators to manage and evaluate contingencies, in the spirit of "If the plan completes early or fails at this step, do these useful low-risk activities" might be a popular evolution, and more in line with the careful way spacecraft are operated than the familiar formulation of automatically choosing contingencies to ensure the primary goal is achieved.

Intelligent Execution Systems

Traditional autonomous control architectures are often based on 3 tiers (sometimes generically referred to as 3T architectures). The lowest tier constitutes the functional layer, that is, the interface between software and the hardware functionality of the spacecraft or other system. The top tier is the AI P&S system that takes several mission goals, finds activities to meet them, and schedules them for execution over an extended period of time, having in mind the resources available. The middle tier is an executive that can run procedures that achieve mission goals as guided by the plan.

Then, an executive is a software component that realizes preplanned actions. Executives are particularly useful in the presence of uncertainty. An executive can be seen as an onboard system that takes the actions of a plan and their expected outcomes (assuming a certain level of certainty) as input and manages their execution in an uncertain and possibly dynamic environment. The technology varies from systems that execute simple linear sequences of commands (Verma *et al.* 2005a), to complex systems (Aschwanden *et al.* 2006) that can plan and schedule in reaction to unexpected changes that endanger completion of the plan.

These systems require a language for the plans, actions and commands they will manage. They may also represent interdependence between actions in terms of precedence or other constraints, in addition to the expected effects of executed commands in order to monitor progress. We can differentiate two types of languages: procedural or declarative. Procedural execution languages are used to develop real world software where loops, branching and parallel execution are needed. Conceptually they specify how the execution should proceed, and thus are often at too low a level to be generated by a planner. For that reason the declarative languages use more a expressive representation of actions that focus on what the actions are rather than how they are to be executed. The intent is for the executive's semantics of execution to interpreting the actions and yield a robust behavior not achieved with traditional execution scripts. This requires significantly more modeling by the user and many constructs such as looping and branching cannot be expressed in a natural way.

As we did for Planning and Scheduling system, we next survey executive systems that have been used in spacecraft operations and then discuss possible trends. Although the literature identifies three categories of executive systems (Verma *et al.* 2005b), in our survey we discuss only two:

- Execution-only systems: The system accepts actions or commands for execution, but does not explicitly exchange information with an automated planner
- Execution systems coupled with an external planner: the systems have explicit interfaces to planners through an execution language or a standard plan format

Execution Systems in Practice

Spacecraft perform actions initiated by commands. The flexibility of the way commands are specified, and the software that must be developed to execute them varies. The simplest approach is *direct commanding* where commands are sent via radio from mission operators on the ground and executed immediately. This is the simplest method, but is still used, for example in low-earth orbit missions or to perform housekeeping activities during a long cruise phase of an interplanetary mission. For deep space missions, light-time delays and contention over the deep space network keeps us from performing large amounts of commanding in this way. In this case, *time tagged* commands allow us to send a sequence of commands each with a time tag all at once. At the appropriate time, each command is executed. Greater flexibility is afforded by *event-based* sequences, where timing of commands may be relative to external events, calculated durations or completion of other commands. The majority of spacecraft have used some variation of direct, time-based or event-based execution of command sequences to great effect. This is the case of the Proba and Smart ESA missions.

Table 2 lists a representative set execution systems that go beyond these methods, and the missions for which they have been in experimental or operational use.

The Spacecraft Command Language (Gaasbeck, Posner, & Buckley 1999) has been used on numerous missions, beginning with the Clementine lunar orbiter. It allows for time-tagged and event-based commanding. It's unique in the execution systems discussed here in that it provides a rule system that allows a procedure to be fired when conditions match a rule. This may be used for responding to anomalies. The SCL was used on the FUSE spacecraft from 1999 until it was de-commissioned in 2007 (Sahnow *et al.* 2000), as well as the EO-1 spacecraft, where it was integrated with CASPER to perform on-board re-planning to train science instruments on potentially interesting phenomena detected by on-board image analysis algorithms (Chien *et al.* 2005).

The highly successful MER rovers use flight software descended from the 1996 Mars Pathfinder lander, and in turn serves as the basis for the upcoming MSL rover (Reeves 2006). The sequence execution module of the flight software accepts a master sequence from mission operators. The master may spawn and kill subordinate sequences running on parallel execution engines, and a newly-uploaded master sequence may make use of sequences from a library that is managed on-board the spacecraft. Synchronization between executing sequences is provided by constructs that wait for a given time or duration, or wait for a shared variable to take on a value, enabling both time-based and event-based coordination of sequences. One of the most interesting aspects of the execution module is that it manages sequences of behaviors provided by high-level spacecraft services, rather than traditional sequences of individual spacecraft commands. This system has been running the MER spacecraft/rovers since launch, approaching 5 years as of the date of this writing.

Executive	Mission	Year	Comments
SCL	Clementine	1994	Time and event-based commanding.
	FUSE	1999	Rule-based firing of procedures
	Earth Orbiter 1	2001	Integrated with on-board science planning
	others		
MPF/MER/MSL Family	Mars Pathfinder Lander	1996	Manages execution of behaviors rather than
	Mars Exploration Rovers	2003	individual spacecraft commands. Similar to 3T.
	Mars Science Lab Rover	2009	
Remote Agent Exec	Deep Space 1	1998	Experiment. Integrated with HSTS and Livingstone FDIR system
VML	Mars Odyssey	2001	
	Spitzer Space Telescope	2003	
	Phoenix Mars Lander	2007	
	Mars Recon, **others**		

Table 2: Operational and experimental execution systems for selected missions

The Remote Agent Exec (Pell *et al.* 1998) accepted a plan for execution in the form of activities and conditions (collectively referred to as *tokens*) on parallel timelines, each with flexible start and end times. Also included were constraints between tokens, for example requiring that a condition (e.g. the camera power is on) meets the start of an activity (e.g. use of the camera) or that one activity must be completed before another (e.g. engine thrusting must complete before science images are taken). For each activity, the Exec would attempt to achieve the conditions that must persist before it, then execute it. When attempting to achieve or maintain conditions, the Exec would make use of the Livingstone diagnosis and recovery system. If conditions needed for an activity failed to hold during its execution, the Exec would suspend the activity and attempt to re-achieve the condition with Livingstone. If the Exec could not re-achieve initial or maintained conditions, it would fail the plan and re-invoke the HSTS planner. The Remote Agent Exec was activated in May 1999 on DS-1 for a 20 hour test and a 6 hour test. Three plans were executed, including three simulated failures with recoveries coordinated with Livingstone and one simulated unrecoverable failure that required generation of a new plan in coordination with HSTS.

Another approach is the sequencing procedural approach represented by the Virtual Machine Language (VML) (Grasso 2002). Sequences can have parameters and the capacity to branch and loop. They are also called functions and they are executed in engines or virtual machines that have two purposes: storing and executing sequences. Just one sequence can be executed at a time in each engine, so in order to obtain parallelism, several (limited) engines are provided for each mission. VML allows looping, conditionals, and use of code blocks. This allows VML sequences to make some adaptation to execution conditions rather than always relying on worst-case assumptions, without making the step to a full-blown intelligent execution system. VML has flown on numerous NASA spacecraft such as the Spitzer space telescope, Mars Odyssey, Mars Reconnaissance Orbiter, Dawn, Genesis, and Stardust. It is slated for future New Frontier and Discovery class missions, such as the Phoenix Mars Lander.

Possible Trends

Concepts from execution research have been transitioning to practice, but fielded systems tend to favor simplicity, focusing on monitoring of execution rather than dynamic rescheduling or other responses to execution problems. Executives are usually script languages with simple iterative sentences as loops or branching. Estimating bounds on resource usage and duration, often through simulation, is important for spacecraft, and eased by simpler semantics. In practice, uncertainty about real-time execution is often handled with conservative resource and duration margins. If a plan executes more quickly than expected, the spacecraft may wait for further instructions. On the MER rovers “bonus activities” are appended to the end of a plan but the expected duration is not increased. If the plan executes quickly, bonus activities are executed until the expected duration expires. Future missions have expressed interest in solving this problem simply but somewhat more generally. Missions are also carrying increasingly complex instruments, often with their own processors, scripting languages and conditional execution. This trend only accelerates the need for simple synchronization constructs to coordinate the operation of multiple independent operations.

Fault Protection Systems

As Neilson points out in an excellent overview of the MER fault protection system, (Neilson 2005) fault protection in an engineering process that incorporates robustness to faults into spacecraft hardware, software, systems engineering and operations. The on-board system for active fault detection, isolation and recovery (FDIR) of possible faults is one output of this process. For most spacecraft, at least a portion of the FDIR system is in hardware. For example, the MER rovers have software-based FDIR systems, but rely on hardware controllers to disconnect the batteries should a hardware fault or software problem drain them to a dangerous level. For the remainder of this section, we will focus on on-board FDIR software, keeping in mind it is only a portion of the overall fault protection system for a spacecraft.

Fault Protection Systems in Practice

On the two MER rovers, subsystem behaviors incorporate subsystem level fault protection. If the rover's arm (IDD) draws more than the allowed current during use, it is marked failed. The IDD behavior ignores any subsequent requests to use the arm, and the rover driving behavior is disabled if the IDD is not stowed away. During the next communication cycle (typically the next day), ground operators can inspect telemetry and debug the IDD before re-enabling it. The MER system also includes a set of system-level fault responses for when disabling a behavior for a particular subsystem is not sufficient. These involve battery undervoltages, overheating, and the like that may indicate a serious problem with rover hardware needed for survival. In these cases, the system-level fault protection software and hardware cooperate to put the rover into a state where it preserves power and has periodic opportunities to communicate with Earth at known times. This approach has the advantage that for most subsystem failures, the nominal behaviors and fault protection are integrated into a localized module. This system has protected the MER rovers for approaching five years. A great summary of anomalies encountered by the rovers in the first 780 sols (Martian days) of operation is available (Matijevic & Dewell 2006).

The Livingstone and Livingstone 2 systems are model-based diagnosis and recovery systems, where a generic software engine or set of principles is developed in the hope of addressing a large class of diagnosis problems(de Kleer & Williams 1989). These systems are adapted to a specific diagnosis problem via an addition of a model of the nominal and failure behaviors of the components they will diagnose. Livingstone is fed the commands given to the spacecraft hardware, and uses the model to predict the expected values of on-board sensors (e.g. switch status bits, temperature sensors, pressure transducers, etc.). If there is a discrepancy between the expected and observed sensors, Livingstone uses the model to find the most likely combination of failures that predict the observed sensor values. Using the same model, it can then determine if there is a way of achieving a desired state (e.g. transmitting data) using a different configuration of the system (e.g. switch from UHF to X-Band communication). The desired advantage is a spacecraft can "fail operational", that is after a failure or anomaly, a goal given to the spacecraft can still be achieved by diagnosing the source of the anomaly, determining what resources remain available, and reconfiguring the spacecraft to execute with the available resources rather than moving to a standby mode. The disadvantage is a significant amount of modeling. Livingstone must be able to predict both the nominal and failure behavior of components that it will diagnose and recovery, as opposed to recognizing a failure signature. In addition, there is a loss of predictability as the system is generating diagnoses and recoveries and then continuing operations, instead of mapping known failure signatures to responses and safing. Livingstone was activated in May 1999 on DS-1 for a 20 hour test and a 6 hour test. During the test Livingstone was fed simulated sensor readings consistent with a set of four pre-determined failure scenarios: switch position indicator failed, camera power switch stuck on, science instrument not responding and thruster stuck closed. In the first case, Livingstone ignored the sensor, and in the remaining cases recommended recoveries of re-trying the command, power-cycling the instrument, and switching thruster control modes, respectively. Livingstone 2 was activated on the EO-1 spacecraft for a total of 143 days in 2004 and 2005 and diagnosed 13 simulated failures (Hayden 2004).

Possible Trends

As with planning technology, model-based diagnosis was demonstrated on DS-1 and EO-1, as well as on testbeds and simulators for the X-34 and X-37 vehicles and many other systems. Yet to our knowledge, no spacecraft has flown this type of general purpose diagnosis and recovery engine as part of a baseline fault protection system. We believe there are relatively simple explanations. One hope for systems like Livingstone was that they could increase science return during routine operations by automatically returning spacecraft to operations. The summary of MER anomalies suggests the rovers have lost less than 3% of operating time to anomalies, and for many of those (*e.g.* stuck in the sand) it's not clear a system like Livingstone would help. Thus it seems fair to question the cost and risk of adding "fail operational" capabilities and continuing to execute after an anomaly. A second hope was that during critical periods such as orbital insertion where one has to continue operating in the face of anomalies, the ability of a model-based diagnosis system to generate novel diagnostic combinations could save a mission. For periods where a misstep could result in mission loss, one has to weigh the potential benefit of generating novel responses to less likely failures against a set of carefully engineered and validated responses to a smaller set of more likely failure scenarios. A broader discussion of this issue is found in (Kurien & R-Moreno 2008).

Conclusions

This paper has presented an overview of techniques applied to Autonomy for Aerospace. Among the techniques we have described AI Planning and Scheduling, Intelligent Execution and Model-Based Diagnosis. We have also presented and described the NASA and ESA missions using these techniques. Finally, the authors envisioned what they thought possible trends in each area would be.

Acknowledgements

We would like to thank Steve Chien, William "Curt" Eggemeyer, Sven Grenander, Luc Joudrier, Glenn Reeves ,Gianfranco Visentin, and Michel van Winnendael for providing insight into some of the flight projects we described. We would like to acknowledge the support of ESA project A0 1-5415/07/F/VS, Junta de Castilla-La Mancha project PAI07-0054-4397, and NASA.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Chase, A.; jung Hsu, J. C.; Jonsson, A.; Kanefsky, B.; Morris, P.; Rajan, K.; Yglesias, J.; Chafin, B. G.; Dias, W. C.; and Maldague, P. F. 2004. MAPGEN Planner: Mixed-Initiative Activity Planning for the Mars Exploration Rover Mission. *IEEE Intelligent Systems* 19(1):8–12.

Aschwanden, P.; Baskaran, V.; Bernardini, S.; Fry, C.; R-Moreno, M.; Muscettola, N.; Plaunt, C.; Rijsman, D.; and Tompkins., P. 2006. Model-Unified Planning and Execution for Distributed Autonomous System Control. In *AAAI 2006 Fall Symposia. Washington DC (USA), October*.

Barba, S. J.; Garcia, L. J.; A.Levine, D.; McElroy, D. B.; Mittman, D. S.; OLinger, J. C.; and R.Tyler, S. 2005. Spitzer space telescope observatory planning and scheduling team. In *Proceedings of SPIE*, volume 6270. SPIE.

Blythe, J. 1999. Decision Theoretic Planning. *AI Magazine* 20(2):37–54.

Bonet, B., and Geffner, H. 1999. Planning as Heuristic Search: New results. In *Procs. of the ECP-99. Springer*.

Bresina, J., and Morris, P. 2006. Mission Operations Planning: Beyond MAPGEN. In *Second IEEE International Conference On Space Mission Challenges for Information Technology*. IEEE.

Bresina, J.; Jnsson, A.; Morris, P.; and Rajan, K. 2005. Activity Planning for the Mars Exploration Rovers. In *Fourteenth International Conference on Automated Planning and Scheduling*.

Camino, O.; Alonso, M.; Blake, R.; Milligan, D.; Bruin, J. d.; Gestal, D.; and Ricken, S. 2005. SMART-1: Europes Lunar Mission Paving The Way For New Cost Effective Ground Operations. In *Procs. of the 6th International Symposium in Reducing the Costs of Spacecraft Ground Systems and Operations (RCSGSO)*.

Cesta, A.; Cortellessa, G.; Fratini, S.; Denis, A. O. M.; Donati, A.; Policella, N.; Rabenau, E.; and Schulster, J. 2007. Mexar2: AI Solves Mission Planner Problems. *IEEE Intelligent Systems* 22(4):12–19.

Chien, S.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 2000. Using iterative repair to improve the responsiveness of planning and scheduling. In *The Fifth International Conference on Artificial Intelligence Planning and Scheduling Systems*.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; and Davis, A. 2005. Using autonomy flight software to improve science return on earth observing one. *Journal of Aerospace Computing, Information, and Communication* 2.

Creasy, R., and Teston, F. 2001. Project for on board Autonomy: PROBA. In *Procs. of the ESA Workshop on On-Board Autonomy.*, 57–68.

de Kleer, J., and Williams, B. C. 1989. Diagnosis with behavioral modes. In *Proceedings of IJCAI-89*, 1324–1330. Reprinted in (**?**).

Eggemeyer, W. C.; Grenander, S. U.; Peters, S. F.; and Amador, A. V. 1997. Long term evolution of a planning and scheduling capability for real planetary applications. In *International Workshop on Planning and Scheduling for Space Exploration and Science*.

Frank, J., and Jonsson, A. 2003. Constraint based attribute and interval planning. *Journal of Constraints*.

Gaasbeck, J. V.; Posner, A.; and Buckley, B. 1999. Distributed space-segment control using scl. In *The Florida AI Research Society Conference (FLAIRS)*.

Grasso, C. 2002. The fully programmable spacecraft: Procedural sequencing for jpl deep space missions using vml (virtual machine language). In *IEEE Aerospace Applications Conference*.

Hayden, S. 2004. Advanced Diagnostic System on EO-1. In *Space 2004 Conference and Exhibit*. AIAA.

Kurien, J. A., and R-Moreno, M. D. 2008. Costs and benefits of model-based diagnosis. In *IEEE Aerospace Conference*. IEEE.

Liberatore, P. 2000. On the Complexity of Choosing the Branching Literal in DPLL. *Artificial Intelligence* 116(1–2):315–326.

Matijevic, J., and Dewell, E. 2006. Anomaly recovery and the mars exploration rovers. In *SpaceOps 2006*. AIAA.

McAllester, D. 1990. Truth Maintenance. *Procs. of the 8th Nat. Conf. AI* 1109–1116.

Morris, P., and Bresina, J. 2008. Active and passive constraint enforcement for activity planning. In *Ninth International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Muscettola, N., and Smith, B. 1997. On-Board Planning for New Millennium Deep Space One Autonomy. In *Procs. of IEEE Aerospace Conference, Aspen, CO, USA.*, 303–318.

Muscettola, N.; Nayak, P. P.; Pell, B.; and Williams, B. 1998. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence* 103(1,2):5–48.

Neilson, T. C. 2005. MER on-board surface fault protection. In *Proceedings of IEEE SMC 2005*. IEEE.

Pell, B.; Gat, E.; Keesing, R.; Muscettola, N.; and Smith, B. 1997. Robust periodic planning and execution for autonomous spacecraft. In *Procs. IJCAI-97*.

Pell, B.; Gamble, E. B.; Gat, E.; Keesing, R.; Kurien, J.; Millar, B.; Nayak, P. P.; Plaunt, C.; and Williams, B. C. 1998. A hybrid procedural/deductive executive for autonomous spacecraft. In *Second International Conference on Autonomous Agents*.

Puterman, M. 1994. *Markov Decision Process: Discrete Stochastic Dynamic Programming*. John Wiley and Sons.

Rabideau, G.; Chien, S.; Willis, J.; and Mann, T. 1999. Using iterative repair to automate planning and scheduling of shuttle payload operations. In *Fifth International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 813–820.

Rabideau, G.; Tran, D.; Chien, S.; Cichy, B.; Sherwood, R.; Mandl, D.; Frye, S.; Shulman, S.; Szwaczkowski, J.; Boyer, D.; and Gaasbeck, J. V. 2006. Mission operations of earth observing-1 with onboard autonomy. In *Second IEEE International Conference on Space Mission Challenges for Information Technology*.

Reeves, G. E. 2006. An overview of the Mars Exploration Rovers flight software. In *IEEE Systems, Man and Cybernetics International Conference*. IEEE.

Sahnow, D.; Moos, H.; Friedman, S.; Blair, W.; Conard, S.; Kruk, J.; Murphy, E.; Oegerle, W.; and Ake, T. 2000. The far ultraviolet spectroscopic explorer: one year in orbit. In *Proc. SPIE*, volume 4139, 131–136.

Selman, B.; Levesque, H.; and Mitchell, D. 1996. A New Method for Solving Hard Satisfiability Problems. *Procs. of the 10th Nat. Conf. AI* 440–446.

Vere, S. A. 1983. Planning in Time: Windows and Durations for Activities and Goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence* pami-5(3):246267.

Verma, V.; Estlin, T.; Jnsson, A.; Pasareanu, C.; Simmons, R.; and Tso, K. 2005a. Plan Execution Interchange Language(PLEXIL) for Executable Plans and Command Sequences. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS), Munich, Germany, 5-8 September*.

Verma, V.; Jnsson, A.; Simmons, R.; Estlin, T.; and Levinson, R. 2005b. Survey of command execution systems for nasa robots and spacecraft. In *Workshop at The International Conference on Automated Planning and Scheduling (ICAPS05)*.

Weld, D. 1994. An introduction to least commitment planning. *AI Magazine*.