On-board Timeline Validation and Repair: A Feasibility Study

M. Fox and **D. Long**
Department of Computer and Information Sciences
University of Strathclyde, Glasgow, UK

L. Baldwin and **G. Wilson** and **M. Woods**
SciSys Ltd, Clothier Road, Bristol, UK

D. Jameux
European Space Agency (ESA)
European Space Research and Technology Centre
Noordwijk, The Netherlands

R. Aylett
Herriot-Watt University, Edinburgh, UK

Abstract

We report on the progress and outcome of a recent ESA-funded project (MMOPS) designed to explore the feasibility of on-board reasoning about payload timelines. The project sought to examine the role of on-board timeline reasoning and the operational context into which it would fit. We framed a specification for an on-board service that fits with existing practices and represents a plausible advance within sensible constraints on the progress of operations planning. We have implemented a prototype to demonstrate the feasibility of such a system and have used it to show how science gathering operations might be improved by its deployment.

Introduction

Communication to distant landers is restricted, both by availability of a communication window and by the time it takes for a signal to pass from transmitter to receiver. This makes it essential to construct plans for the activities of a distant spacecraft, often spanning several hours or days of otherwise unsupervised activity. As has frequently been observed, plans rarely survive contact with reality unscathed. Plans must be constructed using predictions about the outcome of activities of the spacecraft and also predictions of the behaviours and reactions of the surrounding environment. These predictions can diverge from the actual behaviours when a plan is executed. In typical currently deployed systems, plan failure will (depending on the severity of the failure) lead to the spacecraft entering a safe mode and awaiting further instructions, having aborted execution of the remainder of the plan. This response shares an important characteristic with the models on which the plans are based from the outset: they are conservative. That is, both the predictions about the activities of a spacecraft and the response to failures in those predictions act to limit and constrain the science gathering operations of the spacecraft. To illustrate this conservatism, consider that it is now estimated that Sojourner spent at least 50% of its time idle, awaiting further instructions, either because it had completed its planned activities and had nothing left to execute, or because it had entered safe mode following a failure in some activity. Even the immensely successful Mars Exploratory Rovers (MER) mission has been extremely cautious: the original planned mission lifetime for the rovers was only 90 sols, yet they have now been active for more than 850 sols. Even so, they have travelled no more than 7 kilometers in the nearly three years of mission activity. Despite great improvements in the support technology for the planning of MER operations (Ai-Change *et al.* 2003), plans remain conservative and plan execution failures have caused many days of lost science gathering over the lifetime of the mission.

In this paper we describe the Mars Mission On-board Planning and Scheduling (MMOPS) project, in which we explored the construction of a prototype system that would help to address the loss of science caused by conservative mission planning and plan failure. Our prototype has been constructed to work with the Beagle 2 (Blake *et al.* 2004) hardware (see Figure 1), since the on-board software (OBS) and simulator were already available to the team. As part of the project, we have also considered the use of our approach for a mobile lander, such as the planned ExoMars rover.

Our approach has been to design a system that could be deployed on-board a remote spacecraft, granting the craft some measure of autonomy. Other space missions have also explored this possibility, with some success (Chien *et al.* 2004; Jönsson *et al.* 2000). In our work we have not attempted to construct a system in which planning is devolved to the on-board system. Instead, it remains under the control and supervision of the ground operations personnel. The on-board system is designed to manipulate plans (timelines) constructed on the ground, handling three problems that we consider to be central to execution issues on spacecraft:

- Plan failure isolation. When a plan contains an activity that fails, or that it is predicted will fail, the first concern is to isolate the consequences of that failure and to protect execution of the remainder of the plan.
- Over-subscription. When a plan contains more activities than it is predicted that there are resources available to support, activities must be removed from the timeline in order to bring it back within safe bounds. This situation includes both consumable resources such as power and fixed resources such as instruments.
- Under-subscription. Either as a consequence of failure isolation, or conservative estimates for plan execution, if it is predicted that more resource will be available than was expected before plan execution began, additional activities may be performed. We call these activities *opportunities* and they are described in more detail below.

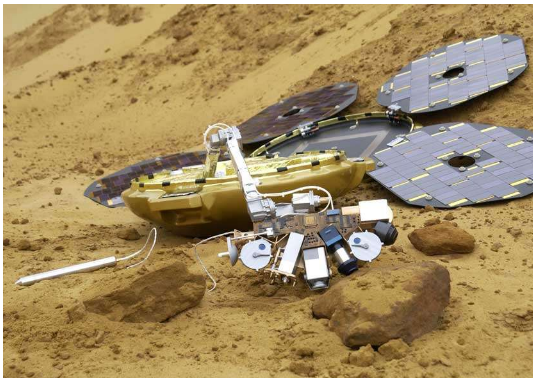

Figure 1: Beagle 2

The system we have developed performs Timeline Validation, Control and Repair (TVCR). These three services provide the foundation of the management of the problems identified above. The design of the system allows these services to be invoked incrementally, so that additional functionality is called on as ground personnel gain confidence in the behaviour of TVCR.

Background

On December 25th, 2003, a small lander, travelling with the Mars Express orbiter, was expected to land on Mars surface. Unfortunately, mirroring the fate of many other attempts to land on Mars, Beagle 2 was unsuccessful. Various explanations of its failure have been proposed, including the possibility that the density of the Martian atmosphere is not as high as had been thought and, as a result, the parachute-brake failed to slow the lander sufficiently before impact. Considerable expertise was built up around the Beagle 2 systems, including a partial domain model for human planning operations, and this formed the core of an initial project to exploit planning technology to support mixed-initiative and partially automated planning for the lander operations (Woods *et al.* 2003) and, from that, the project described in this paper. Beagle 2 was a static lander, equipped with a jointed arm carrying an array of scientific instruments in a "paw" at its end. Included in the paw was a mole capable of drilling into soil around the lander to a distance of more than 2 meters, to retrieve soil samples for gas analysis on board. Beagle 2 was, essentially, a geological survey system, capable of performing an array of geological and environmental measurements in its immediate surroundings.

All lander operations are constrained by power availability, provided by solar energy with a battery for storage, and by temperatures. Planning lander operations involves managing constraints on continuously changing quantities (the generated power levels and temperatures), scheduling the use of resources, planning the movement sequences of the arm and use of the instruments. Human operations planners were to have carried out the planning for Beagle 2 in a complex process involving scientists, providing mission goals and the operations to achieve them, and lander operations personnel, concerned with lander security and, therefore, the power resources and internal lander monitoring systems. When we became involved in the project we discovered that the existing partial domain description was in a form that closely resembled PDDL2.1 durative action descriptions (Fox & Long 2003). It was possible to translate the description into PDDL automatically using a simple automatic translator. The domain encoding began with over 50 actions and this has increased to nearly 70 actions following further domain analysis.

Power is the most important continuous factor in the operations of the lander. The lander operates close to margins and the model of the solar generation and the battery charging profiles are vital in determining when operations can be planned. The management of battery and solar power is sufficiently close to the margins of operational envelopes that plans must interact with the continuous changes involved in the physical system rather than with abstractions into coarse-grained simple step-function changes. Of course, with sufficiently small time-steps a step-function model can approximate the continuous change adequately, but it is infeasible to attempt to model this level of granularity explicitly in the planning domain description. Therefore, this problem demands that the planner (human or otherwise) has access to a sufficiently detailed model of the continuous changes that affect the power systems.

Ground-based Planning, On-board Repair

Our preliminary investigations (Woods *et al.* 2003) convinced us that, while on-board planning technology has an important role to play, there are important reasons why, in the short-term, fully automated plan construction is not the most important objective. The first reason is that on-board resources are extremely constrained, so both CPU and memory availability is likely to prevent realistic planning technology from being deployed on deep space probes in the near future (notwithstanding the important success in the Remote Agent Experiment (Jönsson *et al.* 2000)). The second reason is that neither scientists not operations personnel (those currently responsible for planning of spacecraft operations) are willing to relinquish their tight control over operations until significantly more experience and trust in automated technologies has been built up. Therefore, our goal has been to provide an on-board "planning assistant", providing support to the operations personnel responsible for constructing plans on the ground. The role of the assistant is to adjust and repair plans on-board when circumstances make it impossible execute the plans constructed on the ground. An essential constraint on this behaviour is that the plans that are manipulated on-board are all built on the ground by operations personnel.

Domain Models and Plan Fragments

In order to perform any on-board manipulation of plans it is necessary to provide a foundation for reasoning about the structure of those plans. Our starting point is the action-centred models of AI planning, describing the actions that form the building blocks of plans in terms of their preconditions and effects. Preconditions and effects are evaluated with respect to a model of the state of a system and its environment. The state is represented by a collection of propositions asserting both logical status and also numeric values of metric properties. In our work we have used PDDL (Fox & Long 2003) as our modelling language, since it offers us access to a range of research tools already constructed for the construction and analysis of plans and domains.

Although PDDL offers an expressive language for modelling the behaviour of individual actions and their interactions with other actions, it does not currently offer a way to express several other important constraints on the structure of plans. For example, Beagle 2 was equipped with a rock grinder and various imaging tools: it is standard scientific methodology to perform experiments so that the least invasive investigations are performed first, following them with those that might change the target of investigation physically or, finally, chemically. Thus, the grinder would not be deployed until after images of a target had been captured. This constraint is not a logical constraint on the performance of these actions — clearly, it is perfectly possible to grind a rock before taking any images of it. Instead, this is a methodological constraint on the actions in a plan. Although there are techniques that would allow such constraints to be modelled in PDDL, the fact that these constraints govern not the way in which actions can be performed but the circumstances under which it would be appropriate to perform them is very significant. In particular, methodological constraints may be relaxed under exceptional circumstances, while logical constraints cannot be. Recent developments in PDDL support the inclusion of soft constraints (indeed, this project was a source of significant inspiration to that work) (Gerevini & Long 2006). Methodological constraints are distinguished as one particular kind of soft constraints only because their source is scientists who wish to see them imposed to maximise the value of the scientific return. Other soft constraints might reflect the preferences of operations personnel who might wish, for example, to see the craft maximise its stored energy levels or minimise wear on sensitive components.

AI planning research has been primarily focussed on the construction of plans for goals that specify the conditions that should be achieved in the final state. In the context of space probe plans we found that this was not always the most convenient way to express the purpose of plans. Often, a plan is intended to perform a series of experiments and the simplest way to express their goals is to say which actions they are designed to perform rather than to express the goals in terms of the states that these actions achieve. One reason for this is that the effects of a science gathering activity seen from the perspective of the on-board state are typically to add data to some internal data buffer. The on-board state is not really any different if the data is acquired from a spectrometer or from a camera, but to support the expression of goals in terms of state conditions would require that a distinction be made between the various sources of data, including not only the instrument but also the target. This complicates the model and is counter intuitive for operations personnel when describing plans.

To address these problems we introduced a separate way to capture information about the structure of a plan and its purpose. Our objective was to provide a tool that would allow operations personnel to record information about the structure of a plan — information that was already known to them but has previously not been formally captured or recorded. Essentially, we wanted to capture the information that might be exchanged informally between (human) planners during the initial construction of a plan. This information includes:

- Plan structure. In order to identify the organisation of a plan into coherent blocks of activity we allowed plans to be divided into *plan fragments*. A plan fragment is a group of actions that are together in a plan to coordinate and achieve a single objective. The group forms a coherent unit of related activities, such as preparing equipment, deploying it and gathering data from it before stowing it again. The point in identifying these structures is that these actions are in a plan because of their mutual integrity. If, for example, an instrument is known to be non-operational then there is no benefit in deploying it and stowing it, even if those actions are logically independent of the status of the instrument. These components are very similar to hierarchical task network structures (HTNs), but they are generated on an *ad hoc* basis for each part of the mission, rather than once for the domain as a whole.
- Ordering constraints. When it is important that one activity should precede or succeed another, but this condition is not a logical constraint on the interactions of these activities then we record it as a plan constraint. Such constraints might hold between individual activities or between entire fragments. We distinguish between *ordering dependencies*, where the two constrained elements should either both appear in a valid execution of a plan, or else neither appear, and *conditional orderings*, where the two constrained elements must appear in a particular order if they are both executed.
- Timing constraints. Activities or fragments can be identified as requiring to be executed during darkness or during daylight. They can also be marked as *fixed*, meaning that the time at which they are to be executed cannot be changed. This is typical for communication activities that must synchronise with communication windows shared by orbiters or ground-based transmitters.
- Mutual exclusion constraints. These constraints prevent two activities, or fragments, from coexisting in the same timeline. This constraint can be useful in triggering the use of diagnostic activities where they are inappropriate when the corresponding instrument is functioning, but they should be included in the plan exactly when the activities using the instrument have failed (and therefore have not been executed in the current timeline).

Figure 2 shows the interface to our prototype tool, CONTOOL, developed for this project. It presents a simple view of the timeline and allows selection of elements to be split into fragments. The entry of constraints is managed in the smaller window showing the small coloured and number-

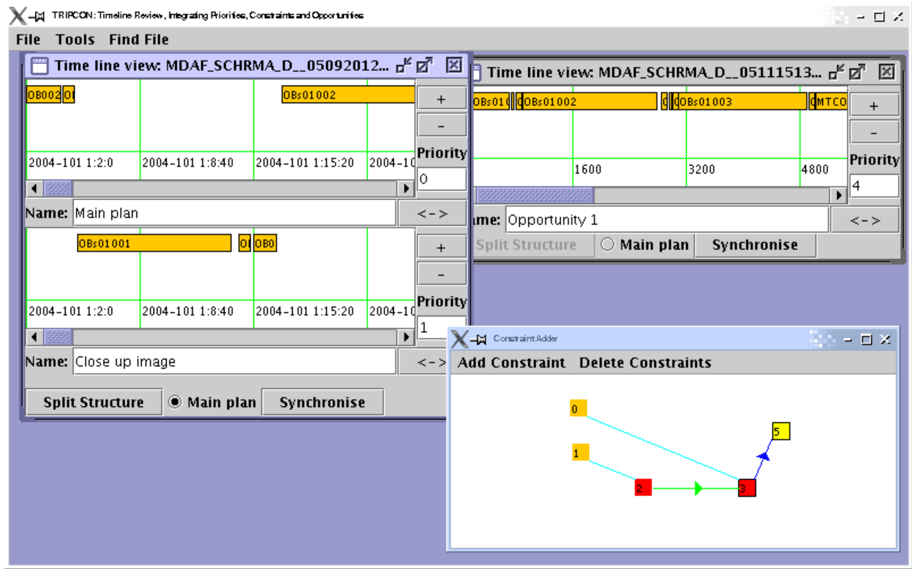

Figure 2: The plan fragment and constraint editing tool.

coded blocks representing activities and fragments. An important aspect of the use of CONTOOL is the identification of plan fragments. The fragments that form the main timeline are the building blocks that form the primary scientific experiments for that period. However, additional fragments can be created and included in the data sent to the lander, identifying optional additional experiments that might be performed. These are called *opportunities* (an example is shown in the upper right window of the screenshot). Each *opportunity* is a self-contained scientific experiment, forming a coherent planned structure of activities that could be executed by the lander. The identification of opportunities allows ground staff to propose useful additional activities that scientists would like to see executed but which have not been selected for the primary timeline for that period. These opportunities are used to reduce the problem of undersubscription: when the lander completes the primary timeline with a significant margin of unused resource — a situation that can often occur because of conservative assumptions made during planning — the excess can be deployed to achieve bonus experiments in the form of opportunities. Similarly, if failure of parts of the primary timeline leaves the lander with freed resource, opportunities can be used to achieve some scientific return from the resource that would otherwise be wasted.

System Architecture

Our system architecture is illustrated in figure 3. The lower part of this diagram shows the ground-based operations planning tools. These include standard timeline planning tools used in earlier ESA missions. The interface between these tools and the lander is achieved via the generated telecommands. The lander is represented using the original Beagle 2 on-board software (OBS), running in an ERC32 emulator on a Sun Solaris machine. Hardware elements of the lander are simulated in software. The Beagle 2 OBS was modified to allow communication with the additional module, TVCR. This module was constructed using several existing software components and was not, therefore, built with a view to on-board deployment. As a consequence, the most efficient connection with the OBS was determined to be through a page of external memory which could be written-to and read-from by the emulated ERC32. TVCR ran as an external library on the Solaris machine, invoked when the external memory page was written-to by the OBS.

TVCR: An on-board planning assistant

TVCR is built around our plan validation system, VAL (Howey, Long, & Fox 2004a; 2004b; Fox, Howey, & Long 2005), coupled with a plan-execution architecture, originally developed for robot control (Coddington *et al.* 2005). The execution architecture is not critical to the behaviour of TVCR, but was a convenient framework in which to work. TVCR receives three types of requests from the OBS: *validate*, *control* and *repair*. The *validate* request is issued when a new timeline has been transmitted to the lander. The OBS issues the request before the timeline begins execution. The timeline is then validated using the on-board state and model. The model is a PDDL (McDermott & the AIPS'98 Planning Competition Committee 1998) description of the domain, using continuous functions that very closely approximate the solar generation and battery use curves. We discuss the model below. The *control* request is issued periodically by the OBS, during execution of a timeline, in order to monitor the continuing evolution of the trajectory. In our prototype the control request frequency was set at 0.1 Hz, but this could be dynamically adjusted to suit the demands on CPU and memory, as well as the granularity of current activities. In response to this request, TVCR can revalidate the plan, but this depends on how close to the

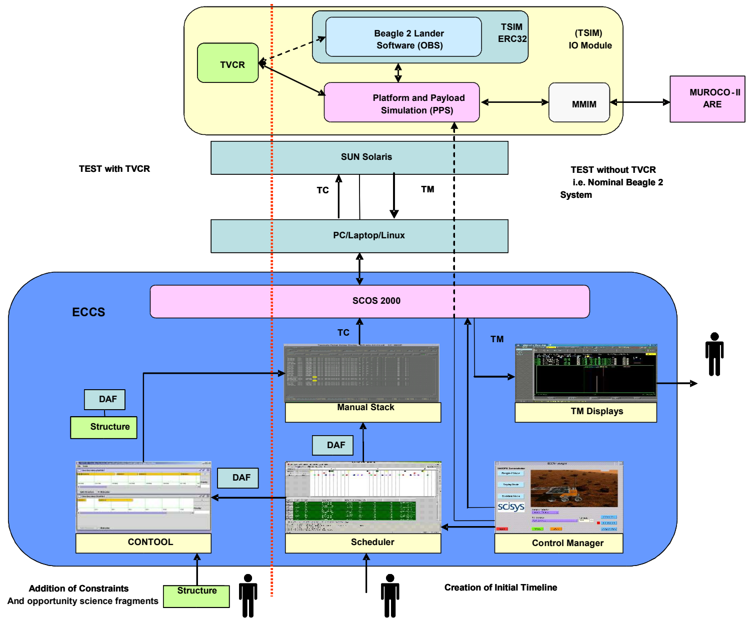

Figure 3: The system architecture, showing both ground and lander segments.

next critical transition in the activities of the system TVCR judges the system to be. In particular, TVCR validates the timeline following initiation of a new activity and approaching the end of an activity. It will also validate the timeline in between activities if there is sufficient gap before the next activity will begin. Essentially, once this gap is small, it is not possible for TVCR to respond to an anticipated activity failure before the failure would be triggered anyway on attempted dispatch to the lander executive.

The final request, *repair*, is only issued by the OBS after TVCR has signalled that an earlier request (either *validate* or *control*) has identified a potential failure in the timeline. In fact, when TVCR identifies such a potential flaw it does not immediately signal the fact to the OBS (although it is logged). Instead, the problem is signalled when TVCR judges that there is a window during which it would be appropriate to respond. One reason for this is that anticipated failures far into the future might be countered by activities in the nearer term completing well within the (usually conservative) estimated resource requirements. Reacting too early to anticipated plan failure can be as damaging as failure to act. TVCR continues to monitor the timeline execution and the anticipated failure point, based on the type of failure that is expected, until the point it considers appropriate to alert the OBS. Once OBS issues a *repair* request, TVCR is sanctioned to act to repair the timeline.

TVCR Acting to Repair the Timeline

TVCR is intended to operate within tightly constrained resource limits: processor cycles and memory availability are both severely restricted. For this reason, we have adopted a strategy that is based on a hierarchy of responses, starting with a simplest response which is to repair a damaged timeline by removing the activities from the current timeline that are affected by the observed failures. This process involves first removing the fragments in which the affected activities lie, using the structure identified by the operations personnel through CONTOOL. The impact of these removals is propagated through the constraints which were also entered by operations personnel. The resulting structure is revalidated and, if there remains a violation of resource demands, further components are removed, using the priority levels set on the ground to determine the order in which fragments are taken out of the timeline. Where there are dependencies between fragments, each fragment is assigned a temporary priority based on the highest priority fragment that would be affected by its removal. The objective, in this first stage, is to arrive, as quickly as possible, at a baseline core timeline that is executable and represents a maximal executable set of fragments from the original timeline. The only available choices in this process are in situations in which more than one fragment has been assigned the same priority. In this case, TVCR uses an estimated resource requirement for each fragment and eliminates the one that demands greatest resource. The priority value is assumed to represent an estimated scientific utility and, if two fragments are of equal utility, the one that returns its rewards at lowest resource cost is considered the better choice. There is no backtracking across these choices, since the alternative can never improve the opportunity to find an executable timeline. The process will always terminate, although it is possible that it might converge on an empty timeline. In the situation where the timeline is empty and the validator still assesses the timeline to be invalid (because resource thresholds are not reached at critical times), the lander is in a precarious state and the best chance for survival is to enter a safe mode in a state of alarm.

Once a valid base timeline has been identified, TVCR attempts to enhance the timeline. TVCR always maintains the current best timeline, so that if the OBS issues a command for an immediate response, or TVCR itself identifies that there is no more time to wait before releasing a timeline (in order to ensure that the next activity is released on time), this current best will be the one that is used. This creates a variant "anytime" behaviour, in that the repair process attempts to use whatever time is afforded to it to improve the existing version of the timeline until it is either interrupted or reaches the end of the process.

The enhancements of the timeline are performed by identifying a subset of possible opportunity fragments that might be added to the timeline. This set is pruned to include only those for which dependencies are satisfied (or mutual exclusions violated) and for which there is sufficient available resource (within the limits of the original primary timeline). The opportunities are then ranked in priority order. In general, there are relatively few opportunities to consider, but there can be choice between alternatives. In particular, when multiple opportunities are awarded the same priority, the choice between them must consider alternative resource demands and the interactions between the alternatives and other outstanding opportunities. In order to maintain a tight bound on memory and CPU demands the search is resolved heuristically, using a greedy selection. This could obviously be improved if resources were to be less constrained, but our experience suggests that greedy choice works well. It is worth recalling that the repair strategy is only applied in situations where resources would otherwise go unused, so even a sub-optimal enhancement of the timeline offers benefits over the timeline without intervention.

The hierarchy of extensions to the timeline begins by considering the addition of opportunity fragments to the timeline. It should be noted that when a timeline is modified, either by the removal or by the addition of fragments, then it is generally the case that some additional linking activities are required between the ends of the newly adjacent parts of the timeline. TVCR has a special-purpose planner designed for this job, which only considers a very small subset of activities (those relevant to this linking behaviour). This problem is very highly constrained and involves no search, but the linking activities must be known in order to determine the costs of execution of new fragments. Following the simple addition of fragments, the next possibility that is considered is the rescheduling of activities in order to open up wider windows of opportunity for execution of longer activities. This is particularly useful when small overlaps prevent an opportunity from fitting into a gap between other activities in the timeline. The rescheduling considers only the possibility of sliding activities along the timeline and this is restricted by any constraints stipulated on activities by operations personnel, including activities that are locked (such as communications activities) or that must occur in certain lighting conditions.

In principle, the addition of opportunities can open up new opportunities, through satisfaction of dependencies, so the set of opportunities can be modified after each enhancement of the timeline. The process of generation of further enhancements is restricted by the time, CPU and memory resources available to the repair process, but TVCR will continue to explore the hierarchy of extended timelines until notified that there is no further resource, or until no further improvement can be found. In our test cases there were relatively few opportunities to consider and reasoning resources were never a limiting factor.

The Domain Model and Timeline Validation

When we began work on the construction of a PDDL domain model, we started with a pre-existing model built by the scientists working on the Beagle 2 project. We found that this model was so similar in form to PDDL that a simple script provided us with an initial translation into PDDL. We concluded that the level of abstraction we require to support plan validation using our PDDL model is an appropriate and natural one for operations personnel and that it corresponds well to the level at which timeline activities are planned in actual missions. The model is based on a pre- and post-condition description of activities, most of which are durative actions. A few actions, such as turning on the torch attached to the PAW, are not best captured as durative actions. The model also required an appropriate power model and temperature model. In these cases we found that the extensions forming PDDL+ (Fox & Long 2006), comprising the addition of *processes* and *events* were an appropriate basis for modelling the domain behaviours. Torch activities are modelled as instantaneous actions (turning on and turning off the torch), with a process consuming power between the two. The power model is constructed using continuous processes that capture an approximate, but realistic, model of solar power generation and of battery state of charge. Our model of temperature is not continuous: we were supplied with a discretised model of the temperature of various nodes on the lander structure at hourly intervals. This is used to construct a series of timed assertions in the initial state of each planning problem instance. We chose to adopt the discretised model in this case in order to demonstrate that our approach can handle both a continuous and a discretised model. In both cases the model is an approximation, although with different levels of discrepancies, and one of the key issues in managing the plan validation is to ensure that the activity models are conservative with respect to these approximations.

Validation of a timeline involves projection of the state through a series of models of activities representing the timeline. This process involves confirming that the interactions between processes (power generation and consumption) are correctly monitored across intervals of activity. Since the functions describing these processes are non-linear, monitoring them involves checking whether non-linear functions have roots within certain intervals. We use a mixture of analytic techniques and numerical techniques to achieve this. Since we do not expect our systems to run too close to operational envelopes, the accuracy of numerical techniques is not an issue in these tests. Indeed, the accuracy of numerical techniques for solving these functions is far greater than the accuracy of the predictive models themselves. The environment introduces uncertainty in the form of degradation of solar panels through accumulation of dust, through atmospheric effects and through temperature effects. As a consequence, the prediction of the state of charge will always be subject to a degree of inaccuracy during actual execution. The objective is to achieve a sufficiently accurate model to enable robust prediction of the outcome of planned activity — if no such model can be constructed because of excessive uncertainty, then the whole purpose of planning is called into question.

Evaluation

Evaluation of TVCR is complicated by the fact that its role is most important when the execution trajectory has performed unexpectedly. We were further constrained by the hardware simulation we were using: this did not simulate the detailed function of some instruments, making it very difficult to model failures in those subsystems. In order to understand the impact of an operational failure during timeline execution, consider the schematics in figures 4, 5 and 6. Figure 4 shows the typical progression of operations, in which timelines are planned during the day (sol) preceding their expected execution (assuming an operational cycle of roughly one sol). At the end of each day the data generated by that day's operations is downlinked for analysis and a new timeline is uplinked for execution in the following day. If there is an operational failure in this cycle (shown figure 5 immediately following activity 21) the ground staff will not become aware of it until the end of that day's activities. The planned timeline for the next day will be downlinked before the ground staff become aware of the problem in the current day's activities. In the worst case, this timeline will no longer be executable, probably because the lander will have entered safe mode in response to the failure. Assuming that the ground staff now wish to execute diagnostics, they will have a first opportunity to perform new activities by the second day following the failure. It is likely that other activities might be included at this time, depending on the severity of the failure. Instrument repairs will then take place on the third day following the failure and normal activity will be resumed only on the fourth day after the failure. With TVCR (figure 6) the failure will (assuming the failure is not too severe) lead to the insertion of opportunities into the timeline on the first day. These can include a diagnostic activity which would be constrained to be added to the plan only if the instrument operation had not been performed (that is, the diagnostics would be mutually exclusive with a successful operation of the instrument for which they were relevant). The planned timeline for the following day would almost certainly not execute directly following the disrupted cycle of operations, but parts of it might execute successfully and TVCR could insert additional opportunities to absorb the otherwise undersubscribed resources available on that day. Meanwhile, the ground staff, armed with the diagnostics, would be in a position to plan instrument repairs and continuation of the remaining science mission, so that by the second day following failure a full sequence of activities could be completed. Normal operations would resume fully on the third day, but the intervening period would have included a fully populated timeline, so that the mission would have continued to gather scientific data and complete diagnosis and repair of instrument failures.

Of course, this schematic view is both simplified and, to some extent, optimistic. In practice, it is quite plausible that TVCR could not completely populate the timeline. In the case of severe system failures, TVCR would be unlikely to be able to anything. However, mission logs demonstrate that minor failures occur regularly and with frustrating frequency, leading to periods of significant science blackout as missions personnel put the lander back into an operational state and resynchronise their view of its status with reality. It is in these cases where TVCR can help to recover what would otherwise be lost time and resources.

To explore the performance we generated a collection of scenarios, based around a common timeline, featuring two rock geology experiments. The primary timeline contained two Mössbauer spectrometer experiments, supported by imaging and rock grinding activities (see figure 7). We then considered a variety of possible failures. These included:

- A scenario in which the Mössbauer was assumed to have failed prior to the timeline being uplinked. This might happen if the instrument had failed in an activity in the timeline immediately preceding this one, so that the failure would be unknown to the ground staff at the time of construction of the new timeline.
- A scenario in which the Mössbauer failed during the first activity in which it was used.
- A scenario in which there was insufficient battery charge at the start of the plan to allow it to complete execution.

In each case, TVCR performed as expected and repaired the timeline. Where a suitable opportunity was available, TVCR inserted it into the timeline, together with the necessary linking activities to create a complete, coherent timeline, which was then successfully simulated to conclusion. As can be seen in figure 8, the timeline executed when TVCR performed repair was significantly enhanced compared with the one executed without TVCR. In the upper case almost all of the timeline is abandoned without TVCR support. In both cases the timeline is repaired by removal of the damaged activity or activities and an opportunity is inserted into the timeline in the place of the first. This is an environmental sensing activity and relies on the PAW being placed into an appropriate attitude. This is achieved by the addition of some simple "glue" activities that reposition the PAW prior to execution of the fragment and then rejoin the execution of the original timeline at its conclusion.

A further dimension for evaluation is that of performance. In our tests, the validation of a timeline of approximately two sols of activity could be performed in less than 3 seconds using an estimated ERC32 processor speed, with repair in less than 6 seconds. Our prototype was not optimised for either memory or processor performance, but we consider this figure to be representative of the performance we can expect from TVCR.

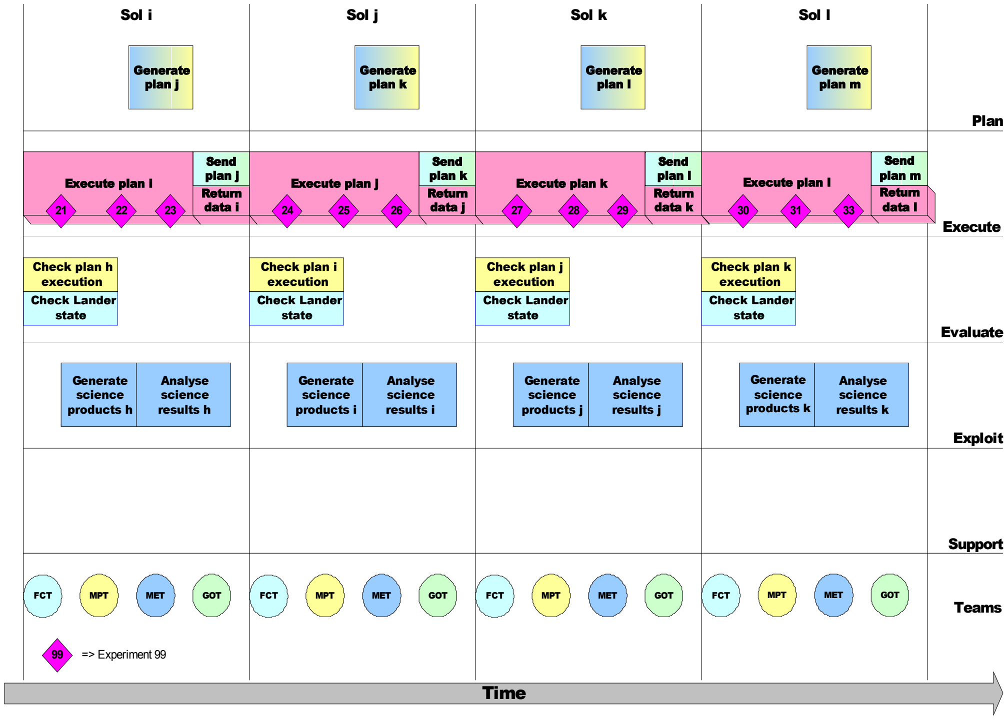

Figure 4: Normal operations sequence.

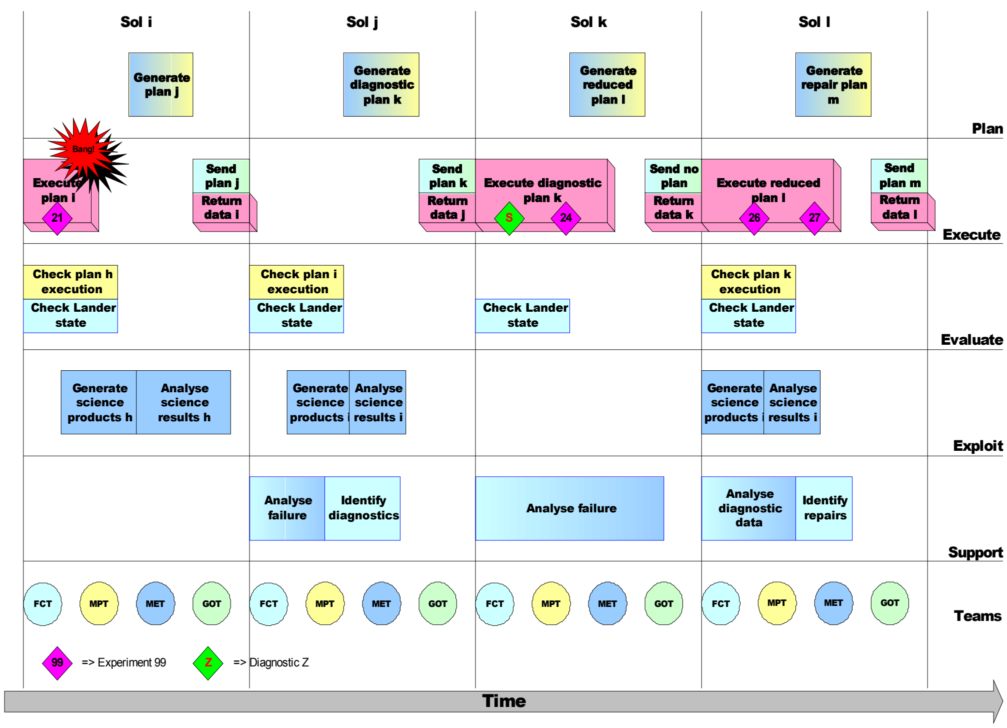

Figure 5: Operations sequence with failed activity.

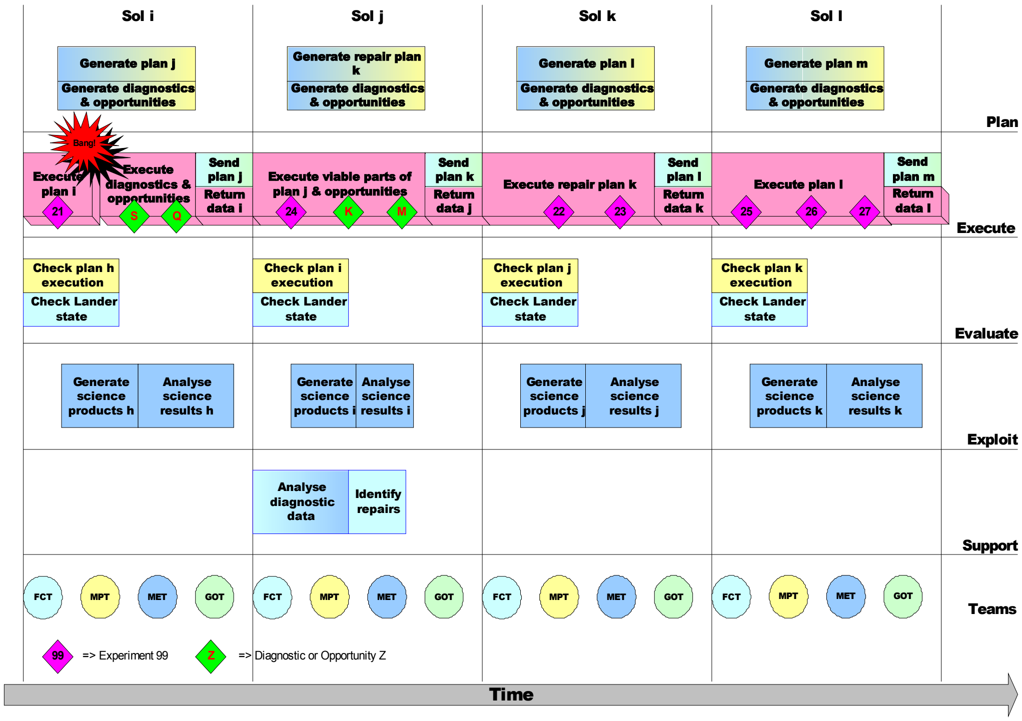

Figure 6: Operations sequence with on-board plan repair.

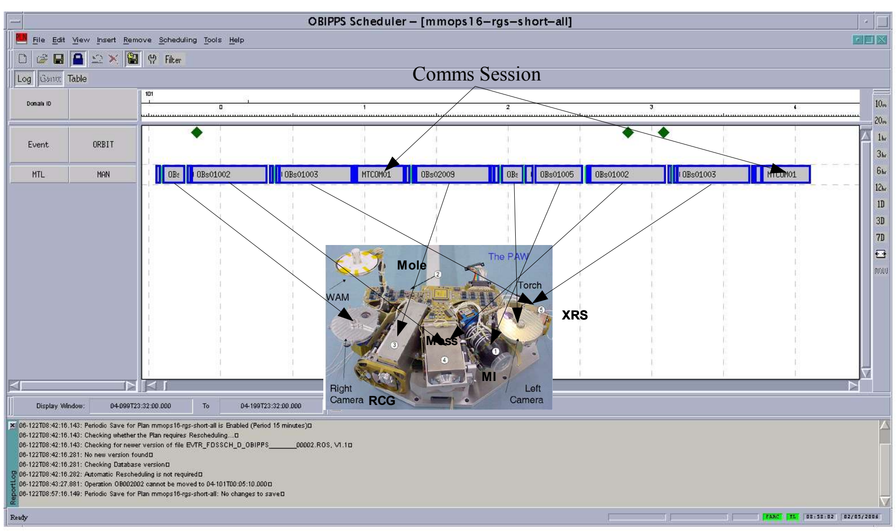

Figure 7: Base timeline showing instruments it commands.

Conclusion

We have successfully constructed a prototype system that acts as an on-board plan validation and repair system. We have used the prototype to demonstrate that on-board reasoning about plans is a practical objective which is less ambitious than full-scale planning, but that can grant huge benefits in terms of science return. Several features contribute to this. Firstly, an on-board system is best placed to monitor execution of a plan and react to failures in a timely and effective way. Secondly, the very fact that plans can be brittle in execution means that operations staff tend to be conservative in their construction of plans, leading to under-subscription of lander resources. The combination of these factors can lead to significant periods of downtime during deep space missions. An on-board planning assistant could exploit the slack during execution of a plan and adapt to plan failures in order to improve the scientific return.

References

Ai-Change, M.; Bresina, J.; Charest, L.; Hsu, J.; Jónsson, A.; Kanefsy, R.; Maldegue, P.; Morris, P.; Rajan, K.; and Yglesias, J. 2003. MAPGEN: Mixed intitive activity planning for the Mars Exploratory Rover mission. In *Proceedings of Demonstration Systems Track, ICAPS'03*.

Blake, O.; Bridges, J.; Chester, E.; Clemmet, J.; Hall, S.; Hannington, M.; Hurst, S.; Johnson, G.; Lewis, S.; Malin, M.; Morison, I.; Northey, D.; Pullan, D.; Rennie, G.; Richter, L.; Rothery, D.; Shaughnessy, B.; Sims, M.; Smith, A.; Townend, M.; and Waugh, L. 2004. *Beagle2 Mars: Mission Report*. Lander Operations Control Centre, National Space Centre, University of Leicester.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Lee, R.; Mandl, D.; Frye, S.; Trout, B.; Hengemihle, H.; D'Agostino, J.; Shulman, S.; Ungar, S.; Brakke, T.; Boyer, D.; Gaasbeck, J. V.; Greeley, R.; Doggett, T.; Baker, V.; Dohm, J.; and Ip, F. 2004. The EO-1 autonomous science agent. In *Proceedings of AAMAS-04*.

Coddington, A.; Fox, M.; Gough, J.; Long, D.; and Serina, I. 2005. MADbot: A motivated and goal directed robot. In *Proceedings of AAAI'05 (Intelligent Systems Demo)*.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of AI Research* 20:61–124.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *Journal of AI Research* forthcoming.

Fox, M.; Howey, R.; and Long, D. 2005. Validating plans in the context of processes and exogenous events. In *Proceedings of The 20th National Conference on Artificial Intelligence (AAAI-05)*.

Gerevini, A., and Long, D. 2006. Plan constraints and preferences on PDDL. In *ICAPS Workshop on Soft Constraints and Preferences in Planning*.

Howey, R.; Long, D.; and Fox, M. 2004a. Plan validation and mixed-initiative planning in space operations. In *Proceedings of the Workshop on 'Planning and Scheduling: Bridging Theory to Practice' (ECAI'04)*.

Howey, R.; Long, D.; and Fox, M. 2004b. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *Proceedings of 16th IEEE International Conference on Tools with Artificial Intelligence*.

Jönsson, A.; Morris, P.; Muscettola, N.; and Rajan, K. 2000. Planning in Interplanetary Space: Theory and Practice. In *Proceedings of International Conference on Artificial Intelligent Planning and Scheduling (AIPS)*.

McDermott, D., and the AIPS'98 Planning Competition Committee. 1998. PDDL–the planning domain definition language. Technical report, Available at: `www.cs.yale.edu/homes/dvm`.

Woods, M.; Aylett, R.; Long, D.; Fox, M.; and Ward, R. 2003. Assessing planning and scheduling technologies for deep space exploration. In *Proceedings of 4th British Conference on (Mobile) Robotics: Towards Intelligent Mobile Robots*.

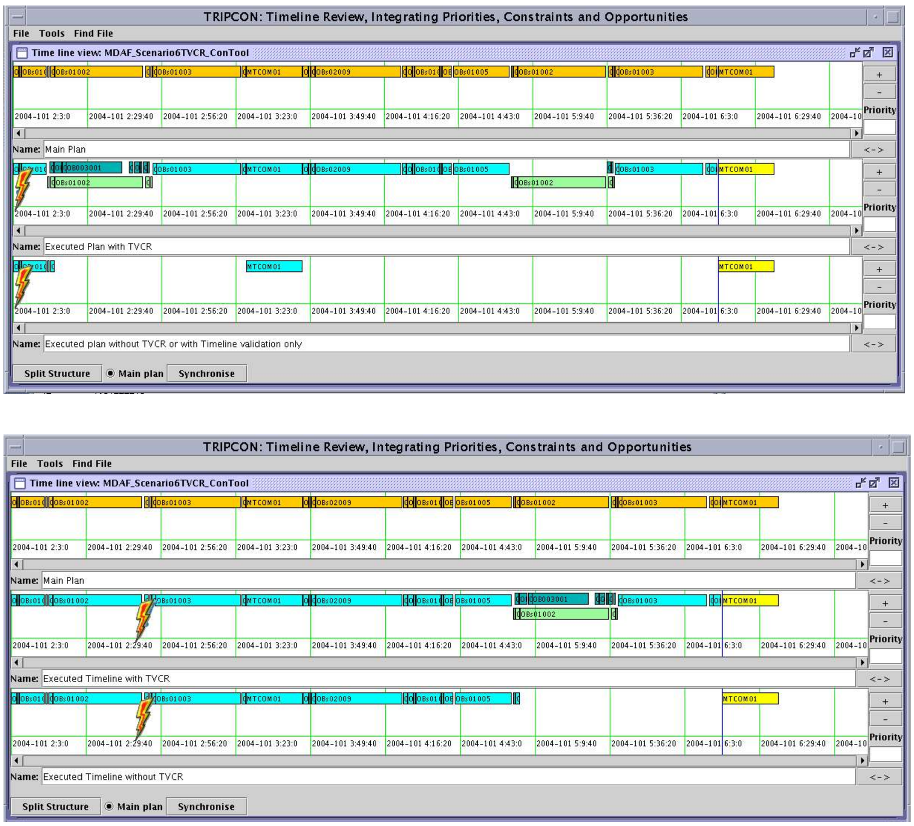

Figure 8: Timeline execution paths. The top screen shot shows the original timeline (top third) and the timeline actually executed with (middle) and without (bottom) TVCR, for the case in which the timeline is recognised at the outset as damaged. Actions shown in the pale colour below the executed timeline are activities that were removed from the timeline during the repair, while dark activities slightly offset above the timeline are new activities added into the timeline by TVCR. The lower screenshot shows the timeline repaired when the Mössbauer fails after its first operation in the timeline.