PMOPS: THE PLANETARY MISSION ON-BOARD PLANNER AND SCHEDULER

Phillip Rendell [(1)], Iain Wallace [(1)], Mark Woods [(1)], Derek Long [(2)]

[(1)] SCISYS, 23 Clothier Road, Bristol, BS4 5SS, UK, {firstname}.{lastname}@scisys.co.uk
[(2)] King's College London, Strand Campus, Strand, London, WC2R 2LS, UK, derek.long@kcl.ac.uk

ABSTRACT

The need for greater on-board autonomy at a tactical mission level has been well established in a series of studies by various space agencies. This study called PMOPS sought to advance prior AI Planning and Scheduling (AIPS) work by refactoring, integrating and prototyping software in a higher TRL space robotic environment. Symbolic plan validation technology from a prior study was integrated into the SCISYS OVERSEER robotic architecture, allowing on-board validation and re-planning, ultimately leading to test and demonstration on a robotic platform in the field.

1. INTRODUCTION

This need for greater on-board autonomy is fuelled by communications delays for missions such as ExoMars to Mars (up to 48 minutes), JUICE to 3 of Jupiter's moons (up to 1 hour 46), and the quantity of data to be returned to Earth. Typically plans are conservative and therefore sub-optimal.

Since 2006, the authors have researched and developed AI Planning and Scheduling Software (AIPS) for various ESA applications based on technologies such as PDDL[1], Val[2], and MADbot[3].

This series of studies showed a benefit of using AIPS technology for certain space applications. The most recent ESA study, MMOPS [4], [5], integrated and evolved a number of academic tools with the Beagle 2 Lander software and operations infrastructure.

In prior work, a dedicated framework called Timeline Validation, Control and Repair (TVCR) realised the primary AIPS functionality and a conglomeration of academically sourced architectures. For PMOPS, only KCL's Val [2] element of TVCR was considered viable and necessary for reuse. Val uses a PDDL plan representation that enables automatic checks for correctness with respect to the plan actions, preconditions, and resource consumption.

This was integrated into the SCISYS OVERSEER robotic architecture running on-board the SCISYS Q8 mobile robotic platform (Figure 1). Plans are created by an operator using the INTERACT mission planning tool and then dispatched to the on-board Executive that executes plans by commanding other OVERSEER components. The PMOPS software validates received plans and has authority to re-plan should validation fail before returning the updated plan to the Executive. As tasks are executed, the MissionPlanning component, developed for PMOPS, receives notifications from the Executive so as to maintain the plan state. Changes in plan state cause revalidation according to the robot state and ensuring that any logical constraints are not violated. Failed validation may mean that the MissionPlanning component invokes re-planning to remove non-mandatory tasks or to add opportunistic sub-plans. As before, any changes to the plan are then sent to the Executive execution.

Events that occur during execution may lead to new planning goals being added to MissionPlanning. Goals are used internally by our re-planner implementation to group sets of plan fragments for opportunistic insertion into the timeline if resources allow and their preconditions are met. These "opportunities" are designed in INTERACT and dispatched along with the plan.

The PMOPS software was tested on the SCISYS Q8 robot in both simulation and on two physical outdoor test sites including a beach. This was deemed a suitable Mars-like environment from a planning and symbolic reasoning point of view. Tests included a valid plan to ensure nominal operation, the persistence of mandatory tasks, replanning when traverses consume more resources than predicted, and the insertion of opportunistic tasks due to early completion of planned activities.

The testing of the PMOPS software raises the TRL of the existing MMOPS software to level 5. The PMOPS software allows operators to create plans knowing that the robot will not be put in danger of using all available resources and opportunities will be taken to maximise science returns.

Figure 1 - The PMOPS software under field test on the SCISYS rover.

1.1. Related Work

Over the past 15 years SciSys has been developing its AIPS software, working alongside academia on a number of projects.

The O-BIPPS and O-BIPPS CCN studies looked at the potential benefits of using AIPS technology across the Aurora and Beagle2 Lander missions.

MMOPS [4] followed on from the O-BIPPS work and integrated and evolved the academic prototype developed in O-BIPPS CCN with the Beagle2 Lander software and operations infrastructure. This showed the benefits of the planning technology and demonstrated feasibility of flight software integration. It combined technologies such as Val [2] (a PDDL-based[1] validator) and MADbot[3] (a project concerned with creating a motivated autonomous system) with the Beagle2 on-board software.

TVCR, the core software from MMOPS was adapted for use on the Crest[6] and Proviscout[7] projects, where it was used on-board a robot with early versions of the OVERSEER system. Crest focussed on how tactical planning-based autonomy can help with surface operations, involving traverses, image acquisition, and science assessment, with science being performed opportunistically if these images were deemed scientifically interesting. While Crest was performed in a lab-based Mars yard, Proviscout was based in Tenerife and advanced this scenario adding autonomous navigation.

Subsequent projects took this and developed the OVERSEER architecture – a collection of components for use with an autonomous robot. Plans of actions were created at a mission control centre and executed on-board a robot. These projects included IRPS[8], Seeker[9] and Safer[10], but the OVERSEER architecture is still in active development. The IRPS, Seeker and Safer projects used a simple test harness for TVCR that simply accepted any plan that it was given, and it was hoped that further work would add TVCR-like capabilities, providing plan validation and a replanning capability – PMOPS represents such work.

2. REQUIREMENTS ANALYSIS

To date, ExoMars Rover (EMR) presents the most challenging requirement for on-board autonomy. In order to increase the TRL of the existing MMOPS software, these were adopted as a baseline for PMOPS and provided a natural follow on from previous work. The mission management system (MMS) requirements were analysed and a number of these were selected to be mandatory, some not applicable, and others 'extended' – requirements that would enhance the developed software but would not be taken into consideration.

A secondary objective of PMOPS was to look at other classes of mission that may benefit from tactical mission planning capabilities, such as Planetary Flybys and Earth Observation (EO), and derive any additional requirements from these.

Currently, ESA is developing the JUICE spacecraft which is an ambitious and complex mission targeted at Jupiter and three of its largest moons. Success will depend on the successful sequencing of the instrument activities with respect to available power and time. Planning a timeline which accounts for such uncertainty given the payload operations complexity is difficult as reliable state estimation is hard. On-board planning would help with this and has been demonstrated in the Deep Space 1 missions.

EO spacecraft are also potential candidates for the use of autonomy. This would provide a closed-loop and fast response to serendipitous events detected in imagery, and allows a fixed team of human operators to control an ever increasing number of satellites.

After looking at these missions, no requirements were added to the ExoMars requirements as mandatory as field trials were to be rover based, but the exercise highlighted this general applicability as above.

3. SYSTEM DESIGN

As mentioned above, the primary objective of the PMOPS activity was to increase the TRL of the MMOPS software. The precursor MMOPS study contained a mix of fused architectures supporting multiple experiment types, namely testing with the Beagle2 flight software and stand-alone use. As such, only the main AIPS component, TVCR, was selected for reuse. All other components presented a means of controlling the simulation/experiments or provided

interfaces with standard on-board software functions and mission control elements which were not required.

TVCR held the primary AIPS functionality and was a conglomeration of academically sourced architectures. Aside from replanning and interfaces to allow it to integrate with other MMOPS components, the other major functionality provided by TVCR was the plan validation element Val.

The MissionPlanning component was developed in order to integrate Val with the SCISYS OVERSEER architecture and provide a level of plan repair functionality (see Figure 2). This was then deployed and tested on-board a SCISYS robotic platform, thus increasing the TRL of the software. Off-board mission control was realised via OVERSEER INTERACT which allowed mission plans to be prepared, dispatched and reviewed during or post execution.

Plans (consisting of sequences of actions – tasks, and their logical containers – sub-plans and jobs) are dispatched from INTERACT to the Executive, which are then passed to MissionPlanning in order to ensure validity against the current state of the robot from the RobotStateEstimator. The Executive then executes each task, commanding other on-board components to perform the required functions. As tasks are executed, the Executive notifies MissionPlanning, prompting validation checks to be performed against the present state of the robot, and the plan adjusted if necessary. For instance, a traverse task consuming more energy than expected might mean that future tasks get removed to stop the platform running out of energy. Conversely, if there are surplus resources then pre-prepared opportunistic sub-plans might be added to maximise science return.

MissionPlanning will only validate the plan and take necessary replanning action on initial submission and when a task has executed. This is contrary to the ExoMars approach where replanning only occurs if the next task to execute is a failure – MissionPlanning treats any invalid plan as a trigger.

4. ARCHITECTURE

Figure 3 shows the architecture of MissionPlanning. Its overall purpose is to ensure the validity of the current onboard plan executing in the Executive, checking it against the current state of the robot.

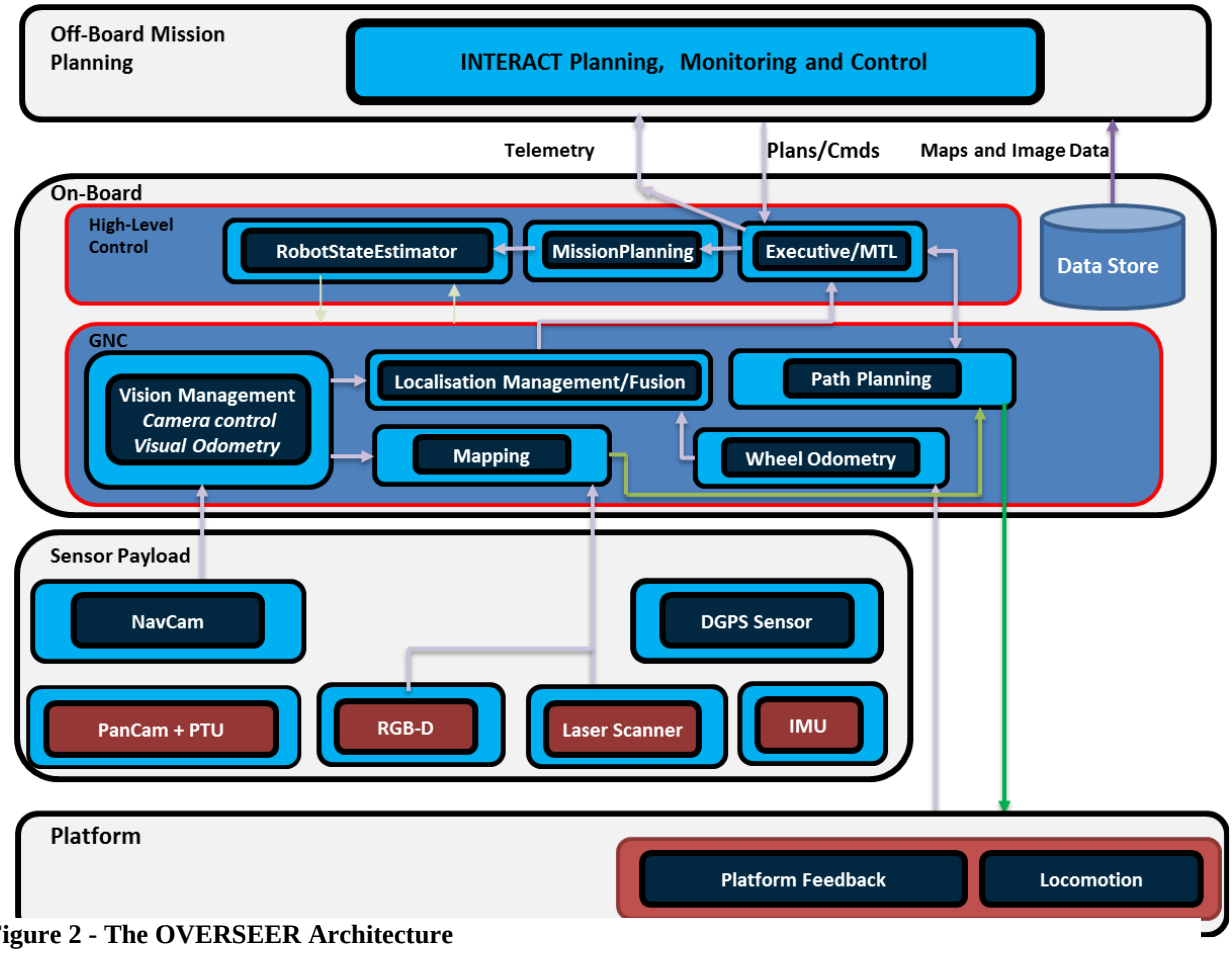

Figure 2 - The OVERSEER Architecture

Five MissionPlanning components are shown in Figure 3:

- **MissionPlanningManager** – acts as an interface to the Executive and to the RobotStateEstimator to get the current state of the robot.
- **Replanner** – submits plans received by MissionPlanning to the Validator and makes changes to this based on their validation results. The Replanner also checks the goals of MissionPlanning and adds any opportunistic sub plans when appropriate to do so.
- **KnowledgeManager** - translates from OVERSEER's native representation of the plan and robot state to a symbolic version that Val can understand.
- **Validator** – uses the Val library to validate plans against the robot state using their symbolic representations.
- **Val** – the library used on MMOPS to symbolically validate plans.

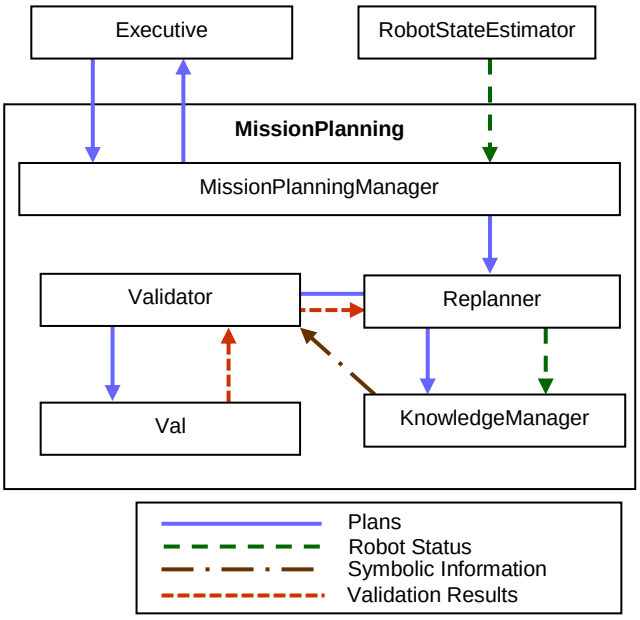

Figure 3 - MissionPlanning architecture

When a new plan or task execution state update is received from the Executive, the MissionPlanningManager obtains the state of the robot from the RobotStateEstimator and sends this with the plan to the Replanner. The Replanner provides these as input to the KnowledgeManager so that the plan can be reasoned about and symbolic representations of tasks can be created. This symbolic representation is used by Val to validate the plan. The Replanner then updates the plan accordingly and returns the plan to the Executive via the MissionPlanningManager.

When translating plans and the robot state to a symbolic representation, the KnowledgeManager has to deal with four types of information: positions, robot facings, and task resource usage. Positions are simply a mapping from approximate 3-dof locations to a symbolic counterpart (e.g. [0, 5, 3] might be translated to location1). Robot facings describe the heading (or yaw) of the vehicle. A high resolution image acquisition task (HRC) might require the robot face a given target, and the robot state reports the current yaw of the robot in radians. This yaw is translated symbolically as the robot at a position, facing another e.g. facingFromTo(robotLocation, facingLocation). The resource usage for tasks is also handled by the KnowledgeManager, as these will be passed to the Validator. It is assumed that one HRC acquisition will use the same amount of energy as another. Traverses are treated in a similar way, but vary in distance, and so are parameterised with their start and end locations. Their resource requirements are calculated on-board the robot based on the previous traverses on similar terrain.

A PDDL-based symbolic plan and associated state file is then generated for Val, which formally and automatically checks this for correctness with respect to tasks, pre- and post-conditions, and resource consumption. If the plan is deemed invalid, Val will return the first task that failed.

MissionPlanning's Replanner implements a simple, deterministic, and time-bounded algorithm, initially removing the first task that Val identified as failing. If this is now a valid plan, it exists. If this is still an invalid plan, tasks are removed from the original plan iterativelylowest priority first until a valid plan is created (the empty plan is always valid). The Replanner will then add these back to the plan, one at a time, highest priority first. Any task that causes the plan to become invalid is discarded. It should be noted that mandatory tasks are always present in the final plan. MissionPlanning's goals are then assessed, potentially adding opportunistic sub-plans when certain events arise, such as the surplus of resources, or an object of scientific interest being detected. This plan is then returned to the Executive.

5. FIELD TESTING

Validation of the MissionPlanning component was achieved using both offline simulation and online physical field trials at SCISYS and on Weston beach. Five plan scenarios were devised to ensure compliance with the ExoMars Rover requirements marked as mandatory, together with three more complex real-world scenarios:

- A basic, valid plan to ensure nominal operation.
- A plan containing a mandatory traverse, an optional task, and a further mandatory traverse,

where there is not enough power to execute the whole plan. MissionPlanning will remove the optional task, ensuring mandatory tasks remain.

- A plan containing two traverses. Energy requirements are recalculated after the first traverse and the terrain is found to be more difficult to traverse over than expected, resulting in the second traverse being removed from the plan.
- A plan containing a traverse that finishes early, resulting in a further traverse and image acquisition, maximising science return.

5.1. Unexpected Terrain Complexity

It is anticipated that traverses over the Martian surface will sometimes consume more resources than anticipated at planning time, possibly due to wheel slippage or the terrain being rougher than was foreseen from remote sensing. In order to cater for this, the RobotStateEstimator records the resources consumed over the previous traverses. Unlike other tasks, where resource requirements are taken from tasks, the KnowledgeManager uses this to estimate required resources for traverses, allowing this type of problem to be detected early. This is an example of when the plan needs to change to account for some environmental condition that the robot can sense but is not known at planning time. Without this adaptive planning, the plan could be a failure and require another communication cycle with replanning from earth.

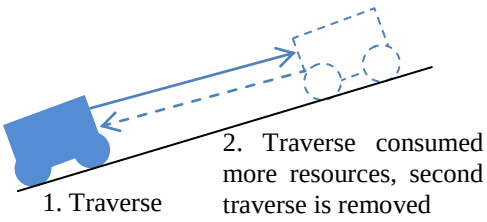

Figure 4 - Unexpected Terrain Complexity Test

In this test the robot was positioned at the bottom of a slope given a plan to traverse up and down assuming flat ground. The amount of energy available to the robot was set such that the original plan would validate successfully but, when the plan is revalidated after the first traverse, MissionPlanning determines that the first traverse consumed so much energy that if the second traverse consumed the same amount the robot would run out of energy, and so removes the second traverse. In a real world scenario, at this point ground would be contacted and required in intervene.

5.2. Early Traverse & Opportunity

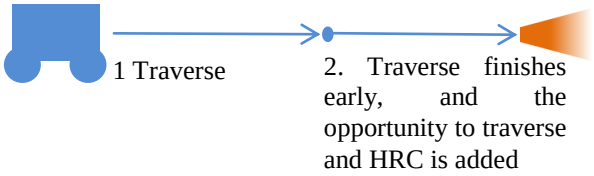

Figure 5 - Early Traverse & Opportunity Test

Occasionally tasks might complete early. When this is anticipated opportunistic sub plans can be inserted when resources allow. These are specified at planning time by the operator, and are sub plans, very similar to sub plans found in plans, except they are inserted opportunistically by the Replanner to satisfy a goal.

In this test a traverse is planned with an opportunity to perform a further traverse and acquire a HRC image. The plan is dispatched, the first traverse completes early and MissionPlanning determines that time and resources allow for the opportunity, which is added, resulting in the further traverse and HRC image acquired.

6. CONCLUSIONS

The MissionPlanning component was implemented to integrate Val, the validator used on MMOPS, into the OVERSEER architecture. It was run on-board a robotic platform in a representative environment, raising the TRL of the core of the MMOPS software to 5. The re-planning algorithms were simple and transparent, making them deterministic in a way that would increase confidence in the system. These algorithms are only invoked on initial plan dispatch and when a task completes in order to reduce computational cost while maximising operational and scientific gains.

6.1. Future Work

Previous analysis of the Mars Sample Return (MSR) mission has highlighted the considerable benefit from autonomy. Future work could take the form of demonstrating the effectiveness of this approach to mitigate against its tight schedule and potential anomalies.

The MSR mission is comprised of three sub-missions. One of these is to collect caches of rock samples from the surface and place these in a container for return the Earth. Several of these collection-return cycles will occur over the course of this mission, meaning that the robot will have to travel over 15km in 180 sols, during which a significant solar conjunction is due to occur, resulting in 30 Sols of the nominal mission where communications are not possible. Dust storms, poor illumination conditions, and normal mission anomalies will further restrict operations causing, with traditional

non-autonomous operations, a very tight mission schedule.

Further work would mean making the following improvements to MissionPlanning:

- Fixed windows of time for tasks such as communications sessions
- Requiring a minimum light quality for pancam image acquisitions and recharge
- Specify a minimum number of samples to be collected per trip (robot might only have enough energy for a 4 out of 5)
- Ability to detect sandstorms so that operations can continue as soon as possible afterwards
- Science assessment for detecting and collecting soil samples in the backup reference mission, also requiring parameterised opportunities

As well as these functional improvements, another goal would be raising the TRL to 6, bringing MissionPlanning closer to being used on-board a mission. This would involve writing MissionPlanning in C, including implementing a simplified Val component, and testing this on flight-like hardware.

REFERENCES

[1] M. Fox and D. Long, 'PDDL2. 1: An extension to PDDL for expressing temporal planning domains.', *J Artif Intell ResJAIR*, vol. 20, pp. 61–124, 2003.

[2] R. Howey, D. Long, and M. Fox, 'VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL', in *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, 2004, pp. 294–301.

[3] A. Coddington, M. Fox, J. Gough, D. Long, and I. Serina, 'MADbot: A motivated and goal directed robot', in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 2005, vol. 20, p. 1680.

[4] M. Woods *et al.*, 'MMOPS: Assessing the Impact of On-Board Autonomy for Deep Space Robotic Missions', in *SpaceOps 2006 Conference*, 0 vols, American Institute of Aeronautics and Astronautics, 2006.

[5] M. Woods *et al.*, 'MMOPS: Developing and Autonomous Timeline Management Capability for a Robotic Mars Mission', in *9 th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, 2006.

[6] M. Woods *et al.*, 'Crest autonomous robotic scientist: Developing a closed-loop science exploration capability for European mars missions', in *i-SAIRAS: International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2008.

[7] G. Paar *et al.*, 'PRoViScout: a planetary scouting rover demonstrator', *Proc SPIE 8301 Intell. Robots Comput. Vis. XXIX Algorithms Tech.*, p. 83010A–83010A–14, 2012.

[8] M. Woods and P. Rendell, 'OVERSEER Visual Planning, Monitoring, Control and Simulation Environment for Robotic Exploration Missions', in *SpaceOps 2010 Conference Delivering on the Dream Hosted by NASA Marshall Space Flight Center and Organized by AIAA*, 2010, p. 2000.

[9] M. Woods *et al.*, 'Seeker—Autonomous Long-range Rover Navigation for Remote Exploration', *J. Field Robot.*, vol. 31, no. 6, Oct. 2014.

[10] M. Woods, A. Shaw, I. Wallace, Malinowski, Mateusz, and Rendell, Phillip, 'Simulating Remote Mars Rover Operations in the Atacama Desert for Future ESA Missions', presented at the 13th International Conference on Space Operations, 2014.