AUTONOMOUS MISSION PLANNING AND EXECUTION FOR TWO COLLABORATIVE MARS ROVERS

Juan M. Delfa Victoria[1,2,3], Brian Yeomans[1], Yang Gao[1], and Oskar von Stryk[2]

[1]*University of Surrey, GU2 7XH Guildford, United Kingdom*
[2]*Technische Universität, Darmstadt, 64289 Darmstadt, Germany*
[3]*European Space Agency, 64293 Darmstadt, Germany*

ABSTRACT

One of the most important lessons learnt from more than ten years of robotic exploration in Mars is the high value of autonomy in terms of operations and science return. So far, only a few full-scale experiments to foster autonomy for planetary exploration have been conducted in Europe. One of the most relevant is the recently completed FASTER FP7 framework project, with one of the main scientific contributions being the application of autonomous planning and execution under uncertainty for two collaborative rovers in a Mars scenario where it is not possible to have humans in the loop. This paper presents the results of implementing the QuijoteExpress (abbreviated QE) mission planner and SanchoExpress executive to tackle these problems. QuijoteExpress is a novel platform-independent planner which strengths are performance, plan robustness and expressiveness. QuijoteExpress was able to produce robust collaborative plans for both rovers with only minor tuning under undeterministic and unstructured scenarios. SanchoExpress is a ROS-based, platform-independent timeline executive that can command multiple subsystems in parallel. SanchoExpress (abbreviated SE) can encode platform-dependent knowledge including repair methods for some errors which allow the executive to fix the plan in certain fault scenarios without the need of the replanner, therefore contributing to a more robust execution of the plan. The combined system was found to operate successfully and reliably during the FASTER field trials, enhancing the systems autonomy and enabling complex behaviour to be executed in a robust and fault tolerant manner.

Key words: Timeline planning; Automated Execution; Planetary rover; QuijoteExpress; APSI; Cooperative robotics; FASTER.

1. INTRODUCTION

Increased autonomy is a priority because constrained communication links, long round trip communication times and the lack of models of remote planetary environments all exacerbate the problems of control for reliable conduct of science and engineering operations.

To date, very few full-scale experiments aimed at fostering increased autonomy for planetary exploration have been conducted in Europe. Two of the most relevant projects in this field are The Goal Oriented Autonomous Controller (GOAC) [5] and the recently completed FASTER FP7 project. GOAC was developed within ESA as an on-board controller to demonstrate key concepts required for fully autonomous operations. FASTER (Forward Acquisition of Soil and Terrain data by Exploration Rover) was a FP7 framework project aimed to enable higher travel velocities whilst reducing risks during the traverse. The project was composed of two exploration rovers working cooperatively, comprising the ExoMars prototype "Bridget" as the primary rover (PR from now on), and a high mobility, lightweight hybrid wheel / leg vehicle used under the control of the primary rover to scout the planned path (SR from now on), gathering data to assess its trafficability. Data on terrain conditions was shared between the rovers to build a global map. In case of elevated risk due to poor trafficability, re-planning was conducted to avoid the dangerous zones, thus enhancing overall mission safety.

This scenario is challenging to execute without humans in the loop.The environment is by definition uncertain, and tasks will take an unknown time to complete. However, for a realistic mission scenario, it is not possible to have humans in the loop. Rather, an approach where the two rovers operate autonomously is required; given the uncertainties inherent in the scenario, a robust application of autonomous planning and execution strategies was required. This paper presents the results from implementing the QuijoteExpress mission planner and SanchoExpress executive to tackle these problems.

2. BACKGROUND

Several temporal planners have been deployed to date in real missions: HSTS [15], used for the short term

scheduling of the Hubble Space Telescope; Europa2 [13], used for several NASA missions including MAPGEN, the ground-based daily activity planning system for the Mars Exploration Rover (MER); ASPEN [7], a hierarchical, timeline-based planner (TLP) used in missions such as EO-1 or DS-1; IxTET, a pioneer about time and resources reasoning; and AP^2[6], a planner developed for the GOAC architecture[5] which integrates planning, execution and replanning capabilities. MER and Curiosity rovers were endowed with multiple dedicated autonomous systems on-board such as auto-navigation[2], target tracking, autonomous instrument placement, dust devils&cloud detection and specially automated science targeting[12]. However, no mission planner has been used to date in a planetary rover mission, mainly due to the hard constrains imposed by a close interaction with such an undeterministic environment. QuijoteExpress addresses some of these challenges by means of a more expressive and faster planner able to handle uncertainty for future space robotic missions such as MSR[9].

Besides an automated mission planner, the spacecraft needs to be capable of timely executing and monitoring the activities contained in the plan. Automated execution has not been so deeply studied as planning from a research point of view. Some general executives such as Smach[1] or Teer[2] have been developed for ROS, but none satisfied the requirements posed by FASTER. SMACH requires as input a model of all automata defined with its own specific language forcing the users to duplicate the modelling for the solving an execution layers. Besides, SMACH does not support the temporal relations required to execute our flexible plans. Teer has a closer approach to timelines but it was not considered powerful enough to properly represent conditional execution. T-Rex has been used with APSI planners in the GOAC experiment[6] but requires a lot of tailoring and depend on proprietary libraries. Moreover, it is not implemented in ROS, making difficult its use in many robotic platforms including FASTER.

Planner and executive are integrated by means of an architecture. It is usually divided in several layers operating at different levels of abstraction and reaction-time. Traditionally, three tier architectures such as 3T[3], Atlantis or LAAS-CNRS clearly dominated. They are based on the following layers: (1) Deliberative: Long term planner; (2) Reactive: Short term planner; (3) Functional: Lowest level layer, tightly coupled with the specific platform. An alternative idea is based on two layers represented by IDEA[16] and CLARAty where the planner and executive work together with a common plan.

Following sections present the architecture designed for FASTER and describes how QuijoteExpress, the mission planner, and SanchoExpress, a reactive executive, worked together in a collaborative multi-rover scenario.

[1]SMACH official site, `http://wiki.ros.org/smach`.

[2]Teer official site, `http://wiki.ros.org/executive_teer`.

3. ARCHITECTURE

To coordinate users, planner and executive, a 3-layers architecture depicted in Figure 1 has been developed. The deliberative layer is in charge of generating and repairing the plan, the reactive performs action dispatching and monitoring while in the functional, the dedicated executives implement platform-dependent low level functions.

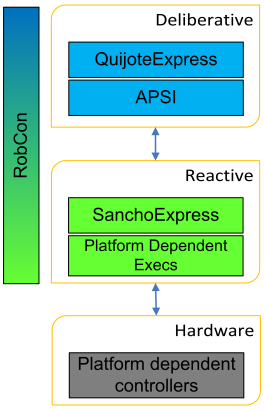

Figure 1. Three layers architecture.

For FASTER, the deliberative components were located in the Ground Segment represented by a laptop. The reactive component, a generic executive installed on the the on-board computer (OBC) of the PR, was in charge of dispatching actions to both rovers. The approach derives from the idea of a master robot in charge of deciding what to do according to the uploaded plan (taking advantage of higher computational capabilities) and a subordinated robot following instructions. This schema is very simple and allowed the mission to reach the targeted E3 level of autonomy, similar to current MER or Curiosity missions. Nevertheless, the architecture can operate in the whole range of autonomy levels, from E1 to E4[1] and can be easily adapted to other scenarios such as distributed robots sharing the same responsibilities.

3.1. Operations in FASTER

The operations cycle starts on-ground with the definition of a new problem, consisting on the specification of the current state of the system and the goals to be achieved. These inputs are passed to the planner on-ground in charge of generating either the initial plan or fixing a failed plan. For this duty, QE uses computational-intensive algorithms that can deeply evaluate the search space.

The outcome is a temporal, hierarchical, sufficient plan that is uplinked to the PR that will start execution at the origin time indicated by the first action. Once the plan is completed, the rovers get into $Idle$ mode until further plans are uploaded. In case of failure, two options are possible: (1) The executive manages to repair the plan

without altering the time constraints and resumes execution; (2) The status of the robots is sent back to Ground Segment while the robots wait for further instructions.

Next sections provide further details about the mission planner and executive.

4. QUIJOTEEXPRESS

QuijoteExpress is an APSI-based planner for autonomous robotic systems (on-ground and on-board) in partially observable, non-deterministic and dynamic scenarios such as Mars rovers or highly reconfigurable satellites.

The planner addresses the problems presented by this type of environments by means of three characteristics: performance, plan robustness and expressiveness.

Improving planning performance is critical in situations where a fast reaction determines the success of the mission. In space, this is critical in missions such as Earth observation satellites that need to rapidly react to capture images of a specific region [8] or Mars rovers to capture serendipitous events such as dust devils [4].

Second, to be able to produce robust plans even for highly complex scenarios where all assumptions previously described are applicable.

Third, to improve the interaction user/planner in two ways: Increase the language expressiveness to create more realistic models and second to generate more understandable plans, easier to be verified and validated by visual inspection.

4.1. Technology

QuijoteExpress is a first-in-class planner based on the combination of applied and classical planning techniques[10, 11]. From an applied point of view, it is based on a hierarchical timeline approach. However, it profoundly deviates from any other timeline planner developed so far to the best of our knowledge, as it uses heuristical, forward-chaining search in the state-space as most modern classical planners such as[14] do, which means that the planner starts searching from the initial state moving forward towards the last state in opposition to temporal planners, based on fixing flaws regardless of their temporal ordering.

The planner is sound and complete but the scheduler inherited from AP^2 is not. Nevertheless, the lack of completeness never resulted on a failure to generate a plan and might not be possible (nor desirable) to be guaranteed under some circumstances, like for an anytime replanner which main target is to provide a solution as fast as possible.

The main novelties that QE presents are:

Level	Description	Functions
E5	Execution of opportunistic-science mission operations on-board	During long missions involving smaller human teams, it allows the robot to autonomously discover and explore serendipitous science opportunities

Table 1. New Autonomy Level.

HTLN: The planner is based on the formal definition of HTLN planning intended to provide a more expressive language and more understandable solutions in the form of hierarchical plans. Contrary to most HTN planners, QE is based on the construction of hierarchical structures rather than goals replacement, adding new levels of detail as complex tasks are refined into subtasks. This strategy is derived from HTLN formalism[10].

Sufficient Planning: The planner can generate partially defined plans which helps to create more robust and versatile solutions as it postpones the need to take some decisions to the moment when the information is available. This could enable a new level of autonomy described in Table 1. E5 seems to be specially suitable for missions such as MSR where the rover is expected to cover big distances every Sol [10].

Parallel Planning: QE can take advantage of modern microprocessors endowed with multi-threading, multi-core capabilities by running in parallel several resolvers and heuristics to improve the performance[10].

Forward-Chaining in the state-space: QE is the first heuristic forward-chaining TLP planner to the best of our knowledge. That technique, adapted from classical planners to timelines, implies that search starts in the initial state towards the goal states (which have been previously ordered) while other temporal planners use a partially ordered approach named plan-space. The aim is twofold: (1) To improve the performance of the planner and (2) To benefit from the huge amount of research performed in this field, incorporating to TLP algorithms and research trends from classical planning[9].

Heuristic Planning: Derived from the previous point, QE can benefit from classical heuristics such as Landmarks or PDBs for satisficing and optimal planning in the state-space in opposition to Fewest Alternative First (FAF) heuristics typically used in TLP planning. The benefits are identical to those stated above[9].

Even though some of these aspects have been previously explored for classical planning, they are certainly innovative in the field of timeline planning. The combination of Sufficient Planning and HTLN might represent the first steps towards more advanced levels of autonomy such as E5 presented in Table 1.

4.2. Planning for FASTER

QuijoteExpress is domain-independent and therefore does not need to be specially tailored for any specific problem. However, we need to provide QE with the appropriate inputs, which are: Domain (D), Behaviours (B) and Problem (P).

4.2.1. Domain model

The domain model D contains a formal description of the components about which it is required to perform planning.

Taking advantage of QE hierarchical capabilities, FASTER domain has been organised in two layers (see Figure 7[3]): abstract and primitive.

The higher layer contains abstract components detached from any specific subsystem, used by operators to define mission goals. Four abstract components were defined for FASTER: Mission, Navigation, Primary Rover and Scout Rover. The Mission component contains generic FASTER mission goals. In nominal situations, the user will exclusively use this component and the planner will take care of adding the corresponding actions for the rest of components in order to achieve the goals. In non-nominal situations such as failures, the operator might want to specify lower level goals to have a better control of the plan generated. Due to the relevance of traverses for this scenario, a dedicated component called Navigation has been defined. With respect to the rovers, each has an abstract component that contains the high level goals each can perform.

The lower layer defines the subsystems of the primary rover, scout rover and external elements. The first contains five subsystem: mission planner, executive, path planner, locomotion and antenna, but only the last three are required to generate plans. The scout has three subsystems: executive, path planner and locomotion, but only the last two are relevant to generate plans. Besides the formal description of the robots, D must contain as well relevant information about those external elements that might pose constraints to the mission such as ground segment, satellites, other robots, etc.

4.2.2. Behaviours (Knowledge DataBase)

While the domain is used to formally describe the world, the behaviours are used to describe mission-dependent concepts, more specifically, how to achieve complex goals by means of networks of subgoals and constraints among them.

[3]Due to the complexity of the model, multiple sub-states and relations have been abstracted away.

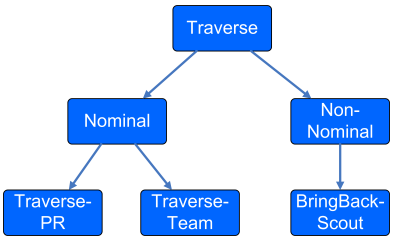

Figure 2. List of hierarchical behaviours included in the Knowledge Database of QuijoteExpress for FASTER.

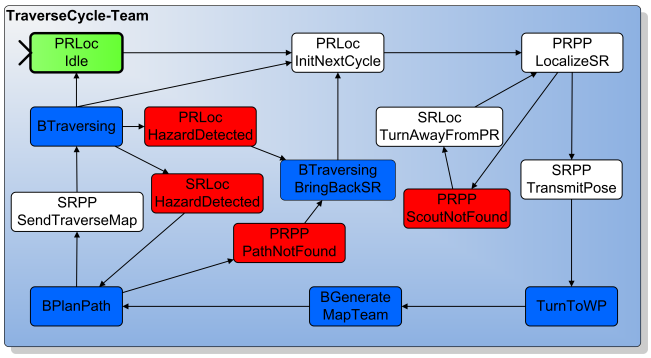

Figure 3. Behaviour for collaborative traverse involving a Primary and Scout rovers.

FASTER behaviours are also modelled in a hierarchical way containing two main groups: $Traverse$ and $Communication$. The first group, illustrated in Figure 2 is the most relevant and models three different types of autonomous navigation, each containing complex activities further subdivided in sub-behaviours:

- $Traverse - PR$: Describes the activities to conduct when the primary is traversing alone. This behaviour might be required in case the scout is lost.
- $BringBack - SR$: Describes the activities to bring back the scout nearby the primary. This behaviour is required in situations in which the scout did not find a path, the primary cannot traverse the planned path towards the scout or there is a change in the mission goals when the two robots are not close.
- $Traverse - Team$: Describes the activities to conduct when both primary and scout are navigating in nominal formation.

The most interesting one is $Traverse - Team$ which decomposition is illustrated in Figure 3.

The execution of this behaviour is only possible in case: (1) Both robots are operational and in formation (see Figure 4); (2) Ground Segment has generated a mission plan containing a $Traverse-Team$ activity and a global map containing a number of identified paths for the rovers[4];

[4]A path is represented as an undirectional graph divided in waypoints, each separated at most 4 meters away due to the limitation in the range of the different sensors.

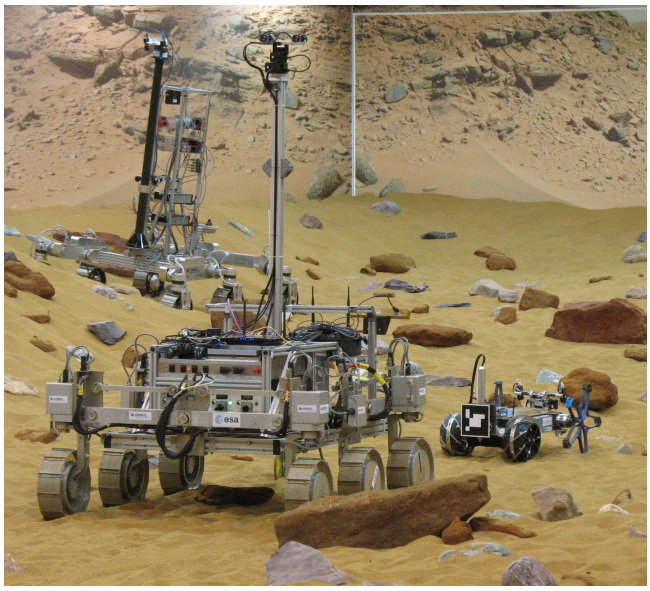

Figure 4. Primary and Scout rovers in formation before starting the next traverse to waypoint (Courtesy of Airbus).

(3) Both the plan and the global map have been uplinked to the PR.

First of all, the PR selects one of the routes and tries to localize the marker of the scout based on imagery provided by the Navcams. If the localization is successful, the PR calculates its position and send it back to the scout. Otherwise, the plan can be fixed by commanding the scout to position itself in front of the primary and the localization process is repeated.

Next, the rovers turn in place until they are facing the next waypoint. The PR generates a new map mixing the sensing information from the scout and primary in the global map and a new path is calculated based on the new information. In case a path cannot be found and the scout is far away from the primary, it is commanded to come back and the process is repeated based on the updated traversability graph. Otherwise, the path is sent to the scout, who starts first to move towards the next waypoint. During the traverse, trafficability information from the Soil Sensing System (SSS) is used to update the local map. If the scout arrives to the waypoint, the primary follows. In case the scout or later on the primary cannot avoid an obstacle in the path, the scout is brought back and a new cycle is started.

The end of the planned traverse can be reached in the following ways. Either the rovers can no longer find a global path to the final waypoint, in which case they should wait to the next communication window to report the problem and obtain further instructions from Ground Segment. Otherwise, the rovers have reached the goal waypoint, which indicates a successful execution of the traversal plan. In case this is not the last activity, execution continues.

5. SANCHOEXPRESS

SanchoExpress is a domain-independent executive to be used on-board with the mission of translating the plans generated by an APSI planner into actions to be executed by the robot or spacecraft. It can be used directly by the user in low-levels of autonomy (E1 or teleoperation) or in direct connection to the planner in higher levels.

An executive is responsible for two tasks:

- Dispatcher: In charge of selecting the next decision d_v to be executed and timely send it to the corresponding subsystem.
- Monitor: In charge of tracking the execution of the decisions and take appropriate measures in case of errors.

5.1. Technology

SE is based on ROS. It is endowed with the following capabilities:

Temporal Execution: Each time-tagged decision contains a starting and ending time, labelled as t_s and t_e respectively. The executive should be able to handle these tags in order to dispatch the decisions at due time.

Parallel execution: Capable of executing/monitoring concurrent systems, e.g multiple sub-systems in parallel.

Fault detection, isolation, and recovery (FDIR): Execution might fail for different reasons. The following FDIR capabilities are mandatory to guarantee the safety of the spacecraft: (1) The executive shall be able to detect and identify failure of an on-going plan; (2) On recognition of a failure, the executive shall attempt to validate and execute contingency action sequences; (3) If possible, autonomous recovery might take place after the execution of the contingency actions.

SE supports the execution of Flexible Domain Timelines. In [17] a flexible domain timeline is defined as a set of two or more totally ordered time points with associated tuples of values plus a set of minimal and maximal distances between each pair of consecutive time points (see figure 6). The first time point is always the origin of the temporal problem while the last is the horizon, i.e., if the temporal problem is defined in $[O, H]$, the first time point of the flexible timeline occurs in $[O, O]$ and the last occurs in $[H, H]$.

Internally, a plan is represented in SE as a matrix of values where each row corresponds to a timeline and each column to a transition (see Figure 6), being a transition represented by a time point in which at least one of the timelines changes its value. For example, $plan[i][j]$ corresponds to the value of the $i-th$ timeline at $j-th$ transition.

SE is divided in two layers: A generic executive and platform-dependent executives.

If SanchoExpress is not executing already a plan then it calls the generic executive. The execution starts with the first transition specified in the plan. For the current transition, SE extracts for all the timelines changing value, their next decision d_v. For each of these decisions, the algorithm sets as the current starting point the lower bound of the current transition and as upper bound (the last instant at which the decision can finish execution) the upper bound of the transition in which its timeline change value again. Finally it calls the dedicated executive in charge of executing d_v at the earliest start time of the lower bound and moves to the next value.

The algorithm iterates over all the transitions until either the last transition is successfully finished, it is interrupted by the user or an error arises during execution. Changing the strategy to an execution based on the latest starting time is trivial. Once the execution is completed, SanchoExpress sends back the plan containing updated information about the execution.

5.2. Execution for FASTER

A dedicated executive is an abstract subsystem that offers a list of services mapped to one or several real subsystems. In FASTER five dedicated executives were defined: Communications, Primary Rover Path Planning, Primary Rover Locomotion, Scout Rover Path Planning and Scout Rover Locomotion. Primary Rover Locomotion for example requires services from two real subsystems: Global Navigation and Primary Locomotion System.

Once a goal is received, the dedicated executive calls the corresponding method in charge of interpreting it. Each method might perform several low level operations involving services from different subsystems. For example, the method $TurnToWP(goal)$, implemented in the ExecutivePrimaryLocomotion to turn the rover to the given waypoint, calls internally the functions $GN\_NextWaypoint$ provided by the global navigation subsystem, $GetTransform$ to change the waypoint coordinates from the world reference to the rover and finally $BLS\_PointTurn$ provided by Bridget locomotion system to actually perform the turn.

Each low level operation is sent to the controller and the executive waits for its completion. At this level is where the monitoring actually takes place. In case the call fails or the returned values from the call are incorrect, the dedicated executive will label the result as $Execution\_Failed$. Finally, the executive returns a message with the updated values to SE.

In the FASTER scenario, some dedicated executives were endowed with short-scope repairing capabilities for very specific errors. In case one of the methods returns an error for which the dedicated executive has a repairing procedure, it will automatically call it. As the failing decision d_v has not been re-planned, the repairing procedure must be completed within the time allotted to d_v. Moreover, it must be taken into account possible issues related with the consumption of resources when implementing these repair procedures. The time t_e in which the repair procedure finishes will be the one indicated in the timestamp of d_v. In case of succeed, SE will not have even noticed that there were a problem and still it will receive in the result the appropriate values regarding resource and time consumption.

6. FIELD TRIALS

The final system was presented to EU, ESA and NASA officials during the full day Workshop and Final Demonstration event, held on October 23rd, 2014 in the New Mars Yard facility, Airbus DS, Stevenage, UK.

The demonstration was based on the following scenario:

- The FASTER System would target a location 15m away in straight line from the starting point.
- The path to the target should contain both visible, hard obstacles (e.g. rocks) of different sizes and hidden hazards consisting of soils considered unsafe for the traverse of the PR.
- The rocks will be placed to force the PR to go through the hidden hazard zone, comprising a subsurface hazard consisting of small pit(s) containing a Sand Trap Analogue covered with a layer of sand to make it visually similar to its surroundings.
- The planned path will be first traversed by the SR, whilst deploying its sensors. The scout should detect the hidden hazard, evaluate its severity using its onboard sensors, and convey data to the PR with a trafficability evaluation and the location of the hazard.
- Data on visible hazards will be derived both from the SR on-board sensors and from PR data

6.1. Preparing the test

Setup of the Mars Yard: It is a large indoor facility that represents a mock-up of a Martian-like environment for testing robotic prototypes. The configuration of the scenario is shown in Figure 5. A sand trap, not visible with cameras, was situated in the way between the rovers and the target. A big rock represented the second obstacle after the trap in the way of the rovers.

Setup of the rovers and computers: The two robots were in formation, pointing toward the target and all subsystems in their default states. The mission planner was

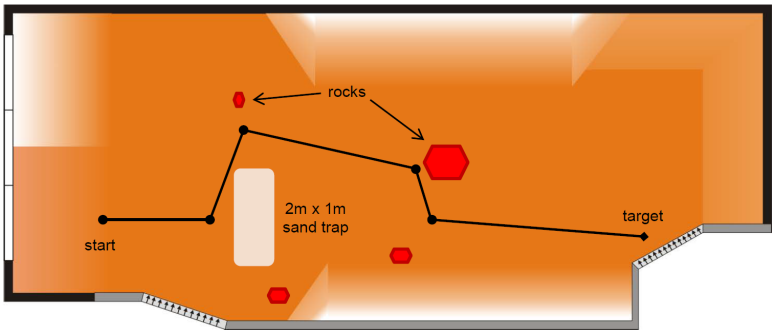

Figure 5. Graphical representation of the Mars Yard configuration for Scenario 1. Sloped sections of the yard are shown by gradient fill areas where the slope rises toward the lighter fill colour. Black lines represent the final trajectory of both robots. A sandtrap is marked as a white rectangle while rocks are red hexagons of different sizes.

deployed in an external laptop, while the rest of the flight software was running in the primary OBC. The three operators where connected to the OBC only to initialize the different subsystems and launch the plan.

Setup of Ground Segment: Field trials required the participation of at least one responsible for each subsystem: PR On-Board Computer (OBC), SR OBC, Planner/Executive, Path planner, Wheel Bevameter, Penetrometer, Primary rover, Scout rover, etc. Upon starting of the tests, only three operators were required: Primary OBC, Scout OBC and Planner/Executive. Once the two OBCs were ready and a plan generated, the Executive operator could launch the plan, starting the autonomous operations.

Mission definition: The mission consisted on just two goals: a long traverse and a final communication with ground segment. The formal specification of the problem for QuijoteExpress included the initial state for all the sub-systems plus the two goals.

6.2. Results

The domain shown if Figure 7, the behaviours partially illustrated in Figure 2 and the problem previously described represented the input for the planner. QuijoteExpress was configured in single-thread mode (no parallelism) and with blind heuristics. Still, a whole plan was generated in less than ten seconds. The planner was configured to generate a flexible plan in which starting and ending times have tolerance margins.

The resulting plan consisted of 12 traverse cycles, each containing 18 transitions organized in five timelines, one for each subsystem. Figure 6 partially illustrates how a traverse for the PR looks like.

In case of failure, two possible repairing activities were considered: (1) The executive could repair a number of specific errors without re-planning; (2) Stop execution, drop some activities of the plan and launch it again.

Once the plan was uploaded in the primary OBC, all subsystems including the generic and dedicated executives were started and the flight software was commanded to start autonomous execution of the plan. At that time the role of the operators got reduced to a mere supervision activity.

The execution proceeded as follows: **Arrival to sand trap**: The scout moved towards the target. Once it entered the sand trap, it produced a "No-Go" trafficability value and a circular area around the trap was marked as non-trafficable.

Executive repairs the plan: ExecutiveScoutLocomotion then called the failure state SRLoc-HazardDetected, which contains a specific behaviour to repair the plan. Internally, the dedicated executive called the path planner to recalculate the path of both rovers, the updated map was sent to the scout and traverse resumed. The repair activities were executed during the time boundaries of $SRLoc-TraverseWithSS$. If the time bounds were exceeded, the error would have been scaled for replanning.

Arrival to first intermediate waypoint The scout then performed the avoiding manoeuvre. Once it arrived to the first intermediate waypoint, it stops and the primary started to move towards it following the updated path.

Reach successive intermediate waypoints: At every intermediate waypoint, the rovers were pointed towards the next waypoint to allow the sensors to generate a map of the area they should traverse. Several perception cycles were repeated between intermediate waypoints.

Arrival to target: Due to the limited time for the demo, the rovers were stopped before reaching the target.

7. CONCLUSIONS

The FASTER team was able to successfully demonstrate the whole system. From the point of view of mission planning, we could demonstrate the validity in a real scenario of a complex plan and the feasibility of flexible

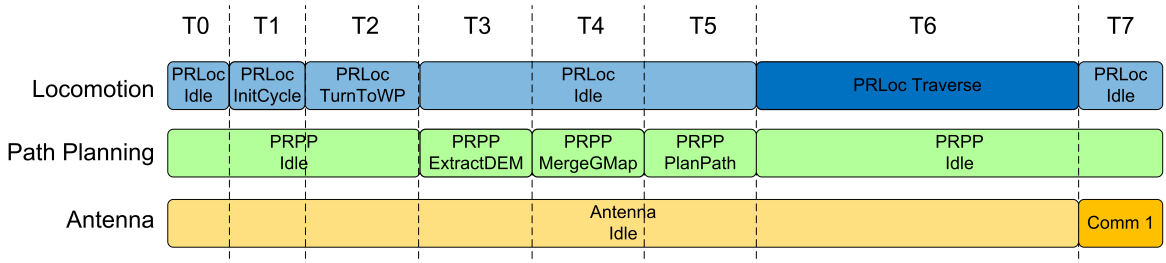

Figure 6. Flexible domain timelines for the FASTER Primary Rover. The highlighted boxes represent goals.

timelines in such scenarios. The executive worked as expected and was able to automatically recognize a failure condition, launching appropriated repairing behaviours that managed to work within time, without the need of any re-planning.

The solution proposed could be easily adapted to different levels of autonomy just modifying the role of QuijoteExpress. It allows the architecture to be used in different scenarios such as robotic teleoperations in near bodies such as the moon or deep space such as Mars.

An important update of QuijoteExpress is near completion and will be soon tested in a rescue scenario to demonstrate its re-planning capabilities. It is also planned to improve in the future the heuristics to obtain even better performance. At the same time, a close integration with a fast path planner has been proposed, allowing QE to make better estimations about time and resources required by rovers on navigation missions. Regarding the executive, the intention is to integrate it together with the deliberative layer in such a way that both share a common representation of the plan.

ACKNOWLEDGMENTS

This work was supported by the European Commission through the SPACE theme of the FP7 Programme, under Grant Agreement 284419 and by the Networking/Partnering Initiative (NPI) between ESA-ESOC and TU Darmstadt. The authors would like to thank the whole FASTER consortium, specially to Space Applications Services and Airbus DS, which contribution to the results shown here were crucial.

REFERENCES

[1] Ecss-e-70-11 space segment operability.

[2] Jeffrey J. Biesiadecki and Mark W Maimone. The mars exploration rover surface mobility flight software driving ambition. In *IEEE Aerospace Conference (IAC)*, 2006.

[3] R. Peter Bonasso, R. James Firby, Erann Gat, David Kortenkamp, David P. Miller, and Mark G. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2):237–256, April 1997.

[4] Andres Castano, Alex Fukunaga, Jeffrey Biesiadecki, Lynn Neakrase, Patrick Whelley, Ronald Greeley, Mark Lemmon, Rebecca Castano, and Steve Chien. Automatic detection of dust devils and clouds on mars. *Machine Vision and Applications*, 19(5-6):467–482, 2008.

[5] Antonio Ceballos, S Bensalem, Amedeo Cesta, L. S Silva, Simone Fratini, F. Ingrand, J. Ocon, A Orlandini, Frederic Py, Kanna Rajan, R. Rasconi, and Michel Van Winnendael. A goal-oriented autonomous controller for space exploration. *ASTRA*, 2011.

[6] Amedeo Cesta, Simone Fratini, Andrea Orlandini, and Riccardo Rasconi. Continuous planning and execution with timelines. In *International Symposium on Artificial Intelligence (i-SAIRAS)*, 2012.

[7] Steve Chien, Gregg R Rabideau, Russell L Knight, Robert Sherwood, Barbara Engelhardt, Darren Mutz, Tara A Estlin, Benjamin Smith, Forest W Fisher, Anthony C Barrett, G Stebbins, and Daniel Tran. Aspen - automated planning and scheduling for space mission operations. In *International Conference on Space Operations (SpaceOps)*, pages 1–10, 2000.

[8] Steve Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, and Rebecca Castano. The eo-1 autonomous science agent. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '04, pages 420–427, Washington, DC, USA, 2004. IEEE Computer Society.

[9] J. M. Delfa Victoria. *Hierarchical Temporal Planning for Autonomous Robots*. PhD thesis, Technical University of Darmstadt, 2015.

[10] Juan M. Delfa Victoria, Nicola Policella, Yang Gao, and Oskar Von Stryk. Quijoteexpress - a novel apsi planning system for future space robotic missions. In *ASTRA*, 2013.

[11] Juan M. Delfa Victoria, Fratini Simone, Policella Nicola, O. von Stryk, Y. Gao, and A. Donati. Planning mars rovers with hierarchical timeline networks. *Acta Futura*, 9(9):21–29, 2014.

[12] Tara A Estlin, Benjamin J Bornstein, Daniel Gaines, Robert C Anderson, David R Thompson, Michael Burl, Rebecca Castano, and Michele Judd. Aegis automated science targeting for the mer opportunity rover. *International Symposium on Artificial Intelligence Robotics and Automation in Space (i-SAIRAS)*, pages 1–25, 2010.

[13] Jeremy D Frank and Ari K Jonsson. Constraint-based attribute and interval planning. *Journal of Constraints Special Issue on Constraints and Planning*, 8(4):339—-364, 2003.

[14] Malte Helmert. The fast downward planning system. *J. Artif. Int. Res.*, 26(1):191–246, July 2006.

[15] N. Muscettola. Hsts: Integrating planning and scheduling. In M. Zweben and M.S. Fox, editors, *Intelligent Scheduling*. Morgan Kauffmann, 1994.

[16] Nicola Muscettola, Gregory A. Dorais, Chuck Fry, Richard Levinson, and Christian Plaunt. Idea: Planning at the core of autonomous reactive agents. In *in Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, 2002.

[17] Fratini Simone. Apsi timeline representation framework v. 3.0. Technical report, ESA, 2014.

APPENDIX – FASTER domain model

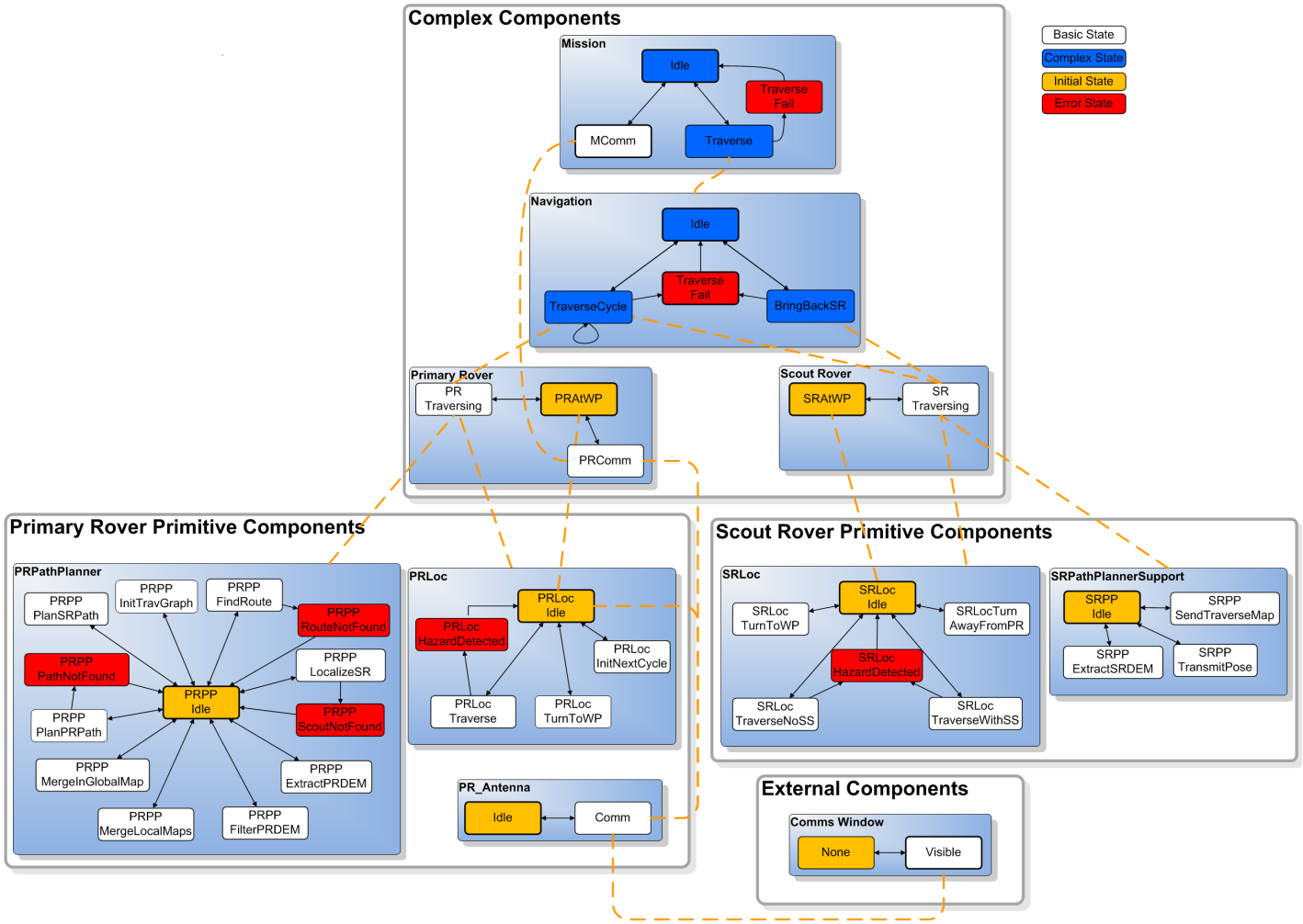

Figure 7. Subsystems of the Primary and Scout rover relevant for QuijoteExpress. Coloured boxes represent different state types: Blue - Complex; White - Simple; Red: Error; Orange: Default. Coloured lines represent different relations between states: Black - Transition; Blue - Decomposition; Dotted orange - Dependency