AIPlan4EU: Planning and Scheduling for Space Applications

K. Kapellos[1], **A. Micheli**[2], **A. Valentini**[2]

[1] Network Research Belgium - TRASYS International BU
[2] Fondazione Bruno Kessler – Planning, Scheduling and Optimization Unit

1 ABSTRACT

AIPlan4EU is a European Horizon 2020 project aiming at lowering the access barrier for practitioners interested in using automated planning and scheduling techniques. The project encompasses eight major use-cases, and many more have been elicited by a third-party funding schema.

Among the major use-cases, 'Planning for Space' targets the automation of the planning process of multi-asset human-robotic missions as prepared by the National Space Agencies, the European Commission, and the European Space Agency. Typical examples are the ExoMars mission for Mars exploration, as well as missions for moon exploration and exploitation. In this context, the ground operators prepare Activity Plans to be executed by the robotic system on the planet as a logical and temporal composition of robotic Activities. In addition, the system shall merge Activity Plans proposed by different remote operators to a final consolidated Activity Plan to be uploaded for execution.

This paper discusses the project genesis and the major results, with a specific focus on the space application use-cases and its integration into ESA activities.

2 INTRODUCTION

Automated planning is the problem of synthesizing a course of actions to achieve a desirable objective exploiting a formal model of the system to be controlled ([1]). Over the years, a plethora of techniques and tools have been developed for different kinds and flavors of planning problems. Notably, classical planning is the problem where the activities of the system are assumed to be instantaneous, and the model is finite-state; numeric planning extends classical planning by allowing numeric variables and arithmetic constraints in the model; temporal planning relaxes the instantaneous duration assumption and allows the modeling of schedules, deadlines and temporal constraints. In addition to these classes, others, such as Multi-Agent Planning, Task and Motion Planning, Contingent Planning, are possible ([2]).

Despite considerable success stories in the literature (e.g., [3]), these techniques are rarely employed in real world applications and their adoption is hindered by technical difficulties. In particular, each planning engine[1] differs from others in terms of supported features, sometimes input language and command line interface. While this is not too problematic for planning researchers, it makes it hard for practitioners to select the right tool for the task at hand and for comparing the performances and merits of different planning engines.

The major objective of the AIPlan4EU Horizon 2020 project is to overcome this situation, providing a unified modeling framework for planning problems and making it trivial to test different planning engines. Moreover, the project is bringing automated planning to the European AI On Demand platform ([4]), reporting a vast number of use-cases, demos and experiences to provide planning users with all the information needed to explore and use automated planning in practice.

Among the AIPlan4EU project use-cases, one of the major ones is planning for space applications, in which we demonstrate the use of the Unified Planning library ([5]) for the automation of the planning process of multi-asset human-robotic missions as prepared by the National Space Agencies, the European Commission, and the European Space Agency. Typical examples are the ExoMars mission for Mars exploration, as well as missions for moon exploration and exploitation. In this context, the ground operators prepare Activity Plans to be executed by the robotic system on the planet as a logical and temporal composition of robotic Activities. In addition, the system shall merge Activity Plans proposed by different remote operators into a final consolidated Activity Plan to be uploaded for execution.

The paper is structured as follows. We first provide an overview of the AIPlan4EU project, its architecture and the major results it has produced so far. Then, we present

[1] We use the term "planning engine" in a very broad sense in this paper to indicate not only plan generation tools, but also any other software asset providing a service in the context of automated planning.

the solution we developed for the space-domain use-case of the project, explaining the problems it can solve, its main characteristics and contextualizing it in the state of the art. Finally, we discuss the next steps and draw our conclusions.

3 AIPLAN4EU ARCHITECTURE

AIPlan4EU is a Horizon 2020 project funded by the European Commission under the ICT-49 call ([6]). The project started in January 2021 and will end in December 2023. The focus of the project is to simplify the accessibility of automated planning technologies bringing them in the European AI On-Demand Platform.

From a technical perspective, the overarching objective of the AIPlan4EU project is to provide a single, easy-to-use access point to planning technology. Concretely, the project developed a Python library, called Unified Planning, which provides a convenient API to model, manipulate, and solve different classes of planning problems. The library leverages a curated collection of planning engines which are integrated and seamlessly available to the users in such a way that it is possible to test and use different engines for a certain problem without caring about the engine details: the library takes care of the needed rewritings and interfacing transparently. The Unified Planning API can then be used to answer different planning queries in applications.

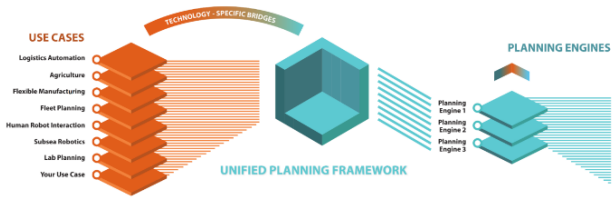

Figure 1: The AIPlan4EU Overall Architecture

In addition, the AIPlan4EU project provides several "Technology Specific Bridges" (TSB), which are interfaces of the Unified Planning services for certain domain-specific technologies. For example, we developed several reusable TSBs for robotics, allowing the use of the Robotic Operating System (ROS) infrastructure to interface with the Unified Planning services. **Error! Reference source not found.** summarizes in an infographic the project vision: the Unified Planning (UP) framework is the central component, providing services to the use-cases through "Technology Specific Bridges" (TSBs) and abstracting away the specificities of planning engines integrated in the library. The figure also shows by means of color the "domain specific" parts in orange and the "domain independent" parts in blue: the UP is designed to be domain independent in the way it offers its services and relies on a series of domain independent planning engines.

The second technical objective of the AIPlan4EU project consists in bringing the unified planning framework to users of the AI On-Demand Platform. This is done in two ways: we provide educational and case study material on the platform, and we provide users of the AI4Experiments tool a reusable component encapsulating the features of the unified planning framework and its engines.

In the following, we describe in more details the Unified Planning framework, the planning engines and the TSB concepts before focusing on the space applications and describe the part of the project concerned with the consolidation of activity plans for ESA missions.

4 UNIFIED PLANNING

In order to concretely realize this vision, we designed a Python library (called `unified_planning` in code and referred to as "UP library" in this document). The library is being developed publicly under a permissive open-source license (Apache 2.0) and the code is available at https://github.com/aiplan4eu/unified-planning.

The library offers a series of unique features to incarnate the vision above. First, it provides a clean and documented API for creating, manipulating and (de)serializing different kinds of planning problems. Second, it provides a set of services to solve planning problems standardizing the interaction with the underlying planning engines by means of "Operation Modes" and "Custom Resolution Strategies". Finally, it provides a plug-in system allowing external planning engine developers to provide new engine capabilities without touching the UP library (and thus benefiting from the UP services and interoperability without being involved in the UP development in any way).

The UP library allows the user to represent planning problems, scheduling problems, plans and hierarchical structures by means of a series of interconnected classes provided by the `unified_planning.model` package. This package provides the basic building blocks to represent planning entities and allows the user to create (either programmatically, or by means of the interoperability facilities) the planning elements to then pass to the engines to be solved/analyzed/transformed. Another core modeling feature we provide is the `unified_planning.shortcuts` package, which provides simplified access to the most common features of the library without importing them from the various sub-packages. The package allows the creation of the basic modeling elements as well as the invocation of

operation modes and the specification of custom resolution strategies. At the time of writing, the UP library supports the following classes of planning problems:

- **Classical planning**: a planning problem where all the fluents are Boolean and the actions are assumed to be instantaneous.
- **Numeric planning**: an extension to classical planning where fluents can be of numeric type and arithmetic can be used in conditions and effects.
- **Temporal planning:** the problem of finding a plan for a planning problem involving durative actions and/or temporal constraints.
- **Scheduling:** a restricted form of temporal planning where the set of actions (usually called activities) are known in advance and the problem consists in deciding the timing of the activities.
- **Multi-agent Planning:** a planning problem where many agents operate in a common environment, each with their own view of the domain. The objective is to produce a set of plans, one for each agent, which allows the agents to achieve their goals.
- **Contingent planning:** an action-based problem in which the exact initial state is not entirely known and some of the actions produce "observations" upon execution.
- **Hierarchical planning:** a planning problem augmented with high-level tasks that represent abstract operations (e.g., processing an order, going to some distant location) that may require a combination of actions to be achieved. Each high-level task is associated with one or several methods that describe how the task can be decomposed into a set of lower-level tasks and actions.
- **Task and Motion Planning** (TAMP) integrates Task Planning, i.e., the problem of finding a set of discrete high-level actions that let the object perform the assignment, and Motion Planning, i.e., the problem of finding a valid (i.e., collision-free) trajectory and subsequent low-level motor commands that allows the object to execute such high-level actions in the continuous-space environment in which it acts.

Also, the plans (i.e., the solutions to planning problems) are represented in a standardized way in the UP library in the package `unified_planning.plans`. Different kinds of problems admit different kinds of plans, but multiple plan kinds can be used on the same problem description (e.g., for a classical planning problem, the simplest kind of plan is a sequence of action instances, but also a partial order of actions instances can be seen as a more general solution). We currently support 5 kinds of plan:

- **Sequential**: an ordered sequence of action instances.
- **Partial-Orde**r: a set of action instances and precedence constraints among them.
- **Time-Triggered**: a sequence of triplets <t, a, d>, where t is the time at which durative action needs to be started and d is the prescribed duration.
- **Simple Temporal Network**: a set of time points indicating either the start or the end of a durative action and a set of temporal constraints of the form x - y <= k where x and y are time-points and k is a rational constant.
- **Contingent Plan**: a tree structure, where each node is an action instance to be executed and the edges are labeled with observation. An execution is a path in the tree where we select the edges that are observed at runtime.
- **Hierarchical Plan**: it combines a sequential or time-triggered plan (that specifies the primitives actions achieving the problems objectives) and decomposition metadata that associate the primitive actions with the high-level tasks they achieve.

Operation Modes (OMs) represent and standardize the possible interactions with a planning engine. Each OM defines an API that an engine willing to support the OM shall implement: in this way, engines declaring to support the same OM can be used interchangeably. All the operation modes can be invoked by using the `Factory` class, which implements the engine-selection mechanism based on our plug-in system. When calling the OM constructor (either from the factory or from the shortcuts) it is always possible to specify the name of the engine to be used (i.e. to force to select a specific engine, if available) or to let the UP to select an engine automatically, by specifying only the `ProblemKind` (which can be retrieved automatically from a problem p using the `p.kind` property). An operation mode defines and standardizes a possible way to interact with a planning engine. The available operation modes are:

- **OneshotPlanner**: single call to a planning engine that given a problem returns a plan (or a failure response).
- **PlanValidator**: given a planning problem and a plan, checks if the plan is valid.
- **SequentialSimulator**: given a problem provides an interactive way to explore the reachable states.
- **Compiler**: transforms a given problem into an equivalent one performing some kind of rewriting.
- **AnytimePlanner**: iteratively generates solutions to a planning problem (e.g., incrementally better plans).
- **Replanner**: interactively generate new plans given a problem and subsequent changes to it.
- **PlanRepairer**: given a planning problem and a (possibly invalid) plan, returns a valid plan.

- **PortfolioSelector**: given a planning problem selects the best engines to solve the problem.

Finally, Custom Resolution Strategies (CRS) are procedural specifications (i.e., python code) that can be used to guide an engine or to specify some action behaviors. The general intuition behind CRSs is that being the UP a python library it makes sense to cleanly expose advanced control capabilities of planning engines by allowing the use of python code. Moreover, it is sometimes hard to specify complex behaviors using standard explicit modeling, while it could be convenient to use procedural specifications. The UP library currently offers two kinds of CRS: namely Simulated Effects and Custom Heuristics.

5 PLANNING ENGINES

The integration of different planning engines in the UP serves the purpose of lowering the barrier for accessing planning technologies. We provide a homogenous Python interface that is exposed to the UP, which delivers the operation modes and we have identified and integrated a number of planning engines from the literature, and developed a number of facilities that enable the use of such planning engines with different operation modes. In particular, we have developed two Problem Specifications Interfaces to meet the different requirements of integration of the planning engines, and we have made the integrated planning engines usable across different operating systems (MacOS, Linux, and Windows). We concentrated on six kinds of planning engines, depending on the expressivity of the planning problem formulation, namely: classical planning, numeric planning, temporal planning, multi-agent planning, refinement planning, and combined task and motion planning. Note that the distinction between these classes does not prevent that some planning engine integrated for a task can also be suitable for one or more other tasks, in fact many engines are capable of handling multiple classes.

We support two types of integrations: one is based on a declarative representation of the task being solved; the other one is based on a procedural representation. The declarative interface is based on a well-known standard language that is used by the planning community: PDDL (Planning Domain Definition Language). This interface is based on transforming the data contained in the UP into PDDL and using the planning engine through a message-passing mechanism based on files. That is, once a problem is formulated in the UP, such a problem is translated into PDDL (using the facilities provided by the UP) and, therefore, any planner supporting PDDL can be connected to the UP. The second interface that we provide is based on a deeper integration of the planning engine in the UP. Instead of interfacing it through files, we populate the data structures of the planning engine directly from the UP data-structures. While the declarative interface makes the infrastructure more modular and requires no change to the implementation of the planning engine, the deeper integration interface potentially reduces the overhead of the integration of the planning engine. More importantly, planners using deep integration can provide advanced features such as custom heuristics and simulated effects, while this is not generally possible with a declarative kind of integration. As we will see, some planning engines support the former interface, while some other planning engines support the latter one. We now report some details about the engines that have been integrated.

- **Fast Downward** ([7]) is a state-of-the-art domain-independent classical planning system based on heuristic forward search. It implements a wide range of heuristics and search techniques that are now accessible through the UP.
- **Pyperplan** ([8]) is a lightweight STRIPS planner developed by the participants of a planning course at the Albert-Ludwigs-Universität Freiburg (Germany). It supports STRIPS actions with uniform costs and emphasizes the clarity of code and the underlying concepts.
- **ENHSP** ([9]) is a heuristic forward state space search planner that looks for a plan in the transition system induced by the numeric planning problem definition. ENHSP is equipped with different search strategies and heuristics leading us to use the system in multiple operation modes.
- **LPG** ([10]) is a planner based on stochastic local search and action graphs. The search space of LPG consists of “action graphs” representing partial plans. The search steps are certain graph modifications transforming an action graph into another one, and the search is guided by relaxed plan heuristics.
- **TAMER** ([11]) TAMER is an application-oriented temporal planner for the ANML (read as “animal”) planning specification language. The objective of TAMER is to provide functionalities to model, solve and analyze planning problems in practice.
- **FMAP** (Forward Multi-Agent Planning) ([12]) is a multi-agent planner aimed at solving multi-agent planning problems in a cooperative, fully distributed and privacy-preserving setting.
- **Aries** ([13]) is a constraint-based planner and scheduler whose primary focus is on hierarchical and temporal planning problems. It works by compiling planning problems to a CSP for which it leverages a specialized combinatorial solver.
- **SpiderPlan** ([14]) is a flexible constraint-based planner that can be extended with new types of

constraints SpiderPlan supports TAMP: motion planning is implemented as propagation and is based on the multi-robot coordination framework ([15]).

6 TECHNOLOGY-SPECIFIC BRIDGES

TSBs connect the client applications that are specific to the different use cases with the Unified Planning framework. A TSB contains components and interfaces that provide additional functionalities that are required to use the UPF in the applications. For example, TSBs collect, maintain and transform all the needed data and encode it in a planning problem by calling the respective UPF library functions, they invoke the UPF engines at the appropriate time to generate a plan, they process the solution to translate it back to the data structures usable by the client technology. Additional TSBs could execute the plan, monitor the execution, decide if re-planning is required, and visualize the plan to the users. In that way, the overall system can consist of multiple TSBs that bridge between the UPF and the application systems. The requirements and implementations of those TSBs heavily depend on the specific use cases.

Within the AIPlan4EU project we developed many TSBs to demonstrate both the internal use-cases and the use-cases elicited by means of open-calls. The internal use-cases of the project are listed below.

- **Space Domain**: Generation and Consolidation of Activity Plans for a remote exploration rover. More details on this are given in the next sections.
- **Agriculture Domain**: Organize agricultural activities for silage maize harvesting.
- **Flexible Manufacturing:** organization of the factory activities to achieve a desired production. The use-case scenario involves the operations to construct truck axles.
- **Logistics Automation**: realization of an offline design aid tool for the automatic debugging of Behavior Trees used to control an autonomous robot capable of intra-logistics tasks. Moreover, we tackled the problem of runtime reactive planning for Behavior Trees.
- **Shuttle Fleet Management**: mission assignment for a fleet of Automated Guided Vehicles to fulfill transportation demands and use the recharging stations optimally.
- **Automated Experiment Design**: automation of consumer goods testing by means of a robotic arm controlled by an automated planner.
- **Subsea Robotics**: realization of an on-line planner and a re-planner for underwater autonomous inspection missions.

In addition, we identified and implemented TSBs offering general functionalities. Those are not specific to a certain technology but more generic and relevant to multiple types of application. In this direction, we identified and implemented two approaches and functionalities that are relevant: the first one is the Embedded Systems Bridge ([16]) which serves two main purposes. Firstly, it helps the user to maintain a mapping between representations on the UP-planning side and the application side. Secondly, it provides functionalities for executing and monitoring a UP plan. Also, although it was initially aimed at robotic applications; it was designed in a way that it is independent from robot frameworks/middleware. The second approach goes the other way and provides nodes for ROS and ROS2 ([17]) that wrap the Unified Planning Framework ([18]).

7 SPACE TSB – INTEGRATION INTO THE EXOMARS ROCS

The Space Use Case aims to introduce automated activity planning into the tactical operations planning of robotic planetary exploration missions. We target the ExoMars ESA Mars exploration mission and we demonstrated the integration of the corresponding TSB into the ExoMars Rover Operations Control System (ROCS) ([20]).

This section is organized as follows:

– Section 7.1 presents the ExoMars rover autonomy concept and its organisation in Activities (Tasks and Actions),
– Section 7.2 presents the ExoMars ROCS operations planning workflow and discusses the benefits of introducing an automated planner, and finally,
– Section 7.3 presents the integration of the UP framework into the Rover Visualization and Planning (RVP) subsystem of the ExoMars ROCS.

7.1. EXOMARS ROVER AUTONOMY CONCEPT: ACTIONS, TASKS & STATE DIAGRAMS

The ExoMars Rover is designed to implement E3 level of autonomy allowing the “Execution of adaptive missions operations on-board”. It is capable to adapt the execution of the plan to the available resources and equipment’s state and automatically switch to Alternative Plans prepared by Ground (e.g., lack of energy with respect to the foreseen energy availability will induce the selection of an Alternative Plan).

At the basis of the design of the E3 Autonomy level are the concepts of Action, Task and the RAPD programming language.

- **An Action** represents elementary rover Activities such as MAST_Initialise and GNC_Initialise. The Actions implement functionalities of a single rover subsystem (e.g., Mast, Wisdom, etc).
- **The Tasks** are logical and temporal composition of Actions (e.g., the RV_Prepare4Travel Task embeds

the MAST_Initialise and GNC_Initialise Actions in parallel). The Tasks, as composition of Actions control several subsystems in parallel.

- Finally, the need of an **event driven Activity Plan** has led to the definition of a new on-board mechanism in addition to the standard PUS services of Mission Time Line, Event Action tables, TC and DTC files, based on an evolution of the DTCF executor concept: the Activity Plan is contained in one (or more) text file(s), following the semantics of a specific Rover Activity Plan Description language (RAPD) and are interpreted on-board.

The identification of the Actions associated to each subsystem is intimately linked with the model of this subsystem. Briefly speaking, an Action can be triggered when the subsystem is at a given state, during its execution sets the system at a new state and finally, at completion, leaves the system at a final state. For the ExoMars rover and mission, a significant work has been performed in close collaboration between the TASI (the prime of the mission), the scientists responsible of their instruments and the rover platform responsible to model all the rover subsystems as Finite State Machines driven by the Actions. In addition, at each state of the model is associated a set of information indicating the resources consumed by the given subsystem at this state. It includes, the Duration time (sec) (when applicable), the Power Consumption (Watt) and the Critical/Non-Critical Data generated.

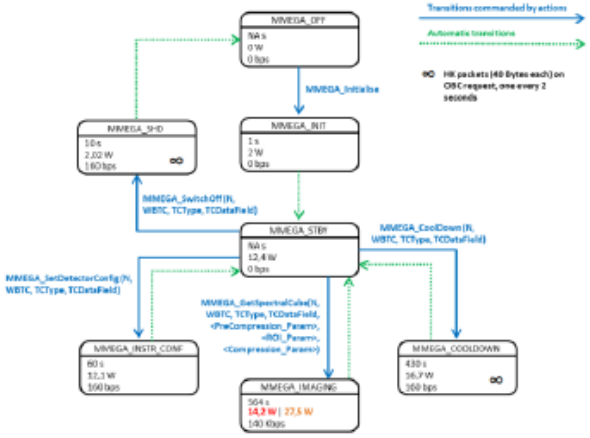

Figure 2: Example of the Micromega subsystem State Diagram & Actions

All the concepts, the models, the Actions and the Tasks described so far are instantiated for the ExoMars Rover implementation in the TASI Functional Architecture Document. In particular, 18 subsystems are identified with 154 associated Actions and 40 Tasks.

7.2. OPERATIONS PLANNING WORKFLOW

The typical rover Activity Planning workflow (also referred as tactical planning) in ExoMars involves a well-defined set of steps presented here below; at each step we detail only the elements that are relevant to the AIPlan4EU activity. There are:

- **Telemetry Acquisition and Processing**: the telemetry packets are received and processed to the appropriate level to generate products to be further analysed by the engineering and science teams.
- **Engineering and Science Data Assessment**: the rover engineers and the scientists, in parallel, analyse the rover system status, the followed path, the payload status, the imagery and the instruments products. Note that the analysis of the downloaded data allows to establish the *Initial State* to be considered as the starting point for the planning operations.
- **Engineering and Science Planning**: the engineers and the scientists, based on the assessment results, in parallel, define the Partial Engineering Activity Plans and the Partial Science Activity Plans they would like the rover executes in the next period. Partial Engineering and Science Activity Plans are manually created by composing Activities using dedicated MMIs. The prepared Partial Plans are submitted to a central Activity Planning tool for further consolidation. The concept of the *Partial Activity Plan* is central in our use case.
- **Activity Plan Consolidation**: during this step an Activity Planner integrates all the submitted Partial Science and Engineering Activity Plans and schedules them in a semi-automatic way in a *Consolidated Activity Plan* so that constraints and resources are respected.
- **Activity Plan Validation and Uplink**: this step covers the validation of the Consolidated Activity Plan by an operational simulator and its uplink. The validation process creates a simulated Execution Report and a simulated Final State with which the real data will be compared in the next Engineering and Science Data Assessment phase.

Note that this workflow is bounded by the time in between the reception of the Rover telemetry and the dispatching of the Activity Plan in the following contact window and each step shall be completed within strict deadlines.

The introduction of automated planning is expected to improve several aspects/steps of the Activity Planning workflow. Let us discuss three specific aspects:

- The **manual generation of Partial Engineering and the Science Activity Plans** requires from the operators (engineers and scientists) to be aware of several engineering low level constraints (e.g., set the robotic system to a state compliant to their objectives, avoid the use of subsystems that create conflicts during operations, be aware about the resource consumption of their objectives). As a result, the Partial Planning step requires a significant amount of time to be completed and very

often provides invalid Partial Plans that are later rejected.

- In the current Activity Planning workflow the **consolidation of the Partial Activity Plans** to a final valid Consolidated Activity Plan to be uploaded for execution is performed manually. Filling the gaps between the partial plans to construct a valid complete plan is time consuming and error-prone. The automatization of this task further decreases the required planning time.
- The proposed Partial Engineering and Science Activity Plans, generally, over subscribe the available resources (time, power and memory). Automated planning may **generate optimal plans** with respect to the available resources and therefore maximize the science return. As a particular case of interest is to automatically identify and propose to the operator the values of well-defined parameters that allow to fit the automatically generated plan in the range of the available resources.

7.3. THE PLANNING MODEL

The core of the Space TSB resides in the so-called "Problem Encoder" component, which is responsible for gathering the information on the robotic activities and to encode the possible evolutions of the system as a planning problem amenable to being solved with the UP library.

The Planning problem is constructed starting from existing engineering models which are currently used for the operations preparation, the simulation and the execution of the mission. In particular, the "Activities Template Library" (ATL) is a database containing all the Actions and the Tasks (see section 7.1) with a description of the parameters such activities expose and their domains. In addition, the ATL contains high-level description of the requirements (i.e., the conditions) for the activities and the post-conditions (i.e., the effects). During the AIPlan4EU project, one of the major advances has been the refinement of the ATL model to contain all the information needed for applying automated planning. We highlight that this approach is different from creating a planning model from scratch to be used for planning: we want to use the very same data source used by ROCS to avoid any problem concerning model misalignment.

The problem encoder reads the ATL and takes in input one or more objectives to be performed by the rover during the tactical planning period. Internally, it computes a planning model using the UP library containing all the possible activities that the asset can perform, it initializes the model and sets the planning objective. The planning model we construct is an automatic abstraction of the information contained in the ATL: we preserve all the details needed for planning and to guarantee the causal consistency of the plans, but we discard the irrelevant low-level information; this is pivotal to construct a model that can be efficiently solved by current planning engines. We then use the 'OneshotPlanner' operation mode to solve the problem and compute a feasible plan. The plan is first validated using a high-level simulator called 'Rehearsal-As-A-Service' to check that the plan is valid considering unmodeled resources. If the plan is valid, it is returned, otherwise we refine the set of goals and look for another plan.

7.4. SOFTWARE INTEGRATION

The AIPlan4EU framework has been integrated into the ExoMars ROCS – RVP component (see Fig. 3) that is at the center of the Tactical Planning process.

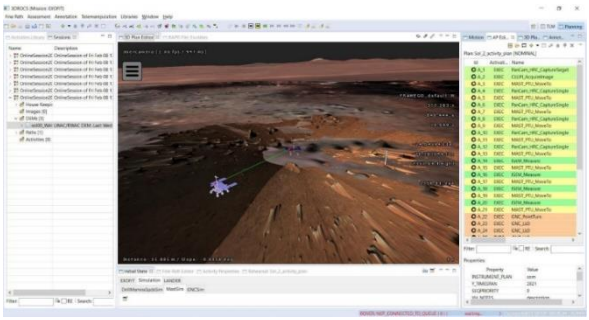

Figure 3: ExoMars ROCS – RVP component

It allows to:

- Render in a synthetic 3D scene the Rover and the environment in which it operates considering the rover model, the terrain model, the illumination sources, shadows and textures,
- Annotate the scene with paths, targets, areas and labels to support the rover path planning,
- Rehearse rover and rover mechanisms motions to support the operations planning,
- Present to the operator the available Activities for ground planning and create Partial Activity Plans as a composition of Activities,
- Rehearse the Activity Plans (Partial, Nominal or Alternative), visualize the consumed resources on dedicated views and charts and finally visualize the Activity Plans in a Gantt Chart form.

The integration with the UPF is performed as follows:

- The data model is enhanced to integrate the concept of the Goal; the available Goals are included in the ATL and visualized in the 'Activities Library' view.
- The 'Activity Plans Editor' view accepts as inputs user selected goals and allows the operator to request the automatic generation of an Activity Plan. In case a valid Plan cannot be generated the UPF provides the reason in terms of Goals/States that cannot be reached.
- The operator may also request by the UPF the validation of user defined Activity Plans; the provided feedback allows him to progressively construct a valid Plan.

- The ‘Consolidated Plans’ view is also connected with the UPF allowing to construct a complete plan (Consolidated Activity Plans) from the set of the Partial Plans submitted by the science and engineering teams.

From a software point of view the UPF is deployed in a container and provides its services using REST API.

The benefits of using automated planning in the Activities Planning Workflow has been evaluated and confirmed in the particular case of preparing the ExoMars nominal ‘sol 5’ operations for ‘subsurface sample collection’ involving rover preparation for travelling, travelling to the area of interest, preparing for drilling, drilling and acquiring a subsurface sample and finally transferring the sample into the sample container.

8 CONCLUSIONS

In the paper we presented the AIPlan4EU European Horizon 2020 project aiming at lowering the access barrier for practitioners interested in using automated planning and scheduling techniques.

It provides a single, easy-to-use access point to planning technology called Unified Planning. It allows to model, manipulate, and solve different classes of planning problems with the support of a collection of planning engines which are integrated and seamlessly available to the users The Unified Planning API can then be used to answer different planning queries in applications.

Many ‘Technology Specific Bridges’ demonstrated the effectiveness of the approach in various domains ranging from space to agriculture, flexible manufacturing, logistics and subsea robotics.

In particular, in the Space domain, the Unified Planning framework has been integrated into the ExoMars Rover Operations Control System (ROCS) – RVP component responsible for the operations tactical planning. The evaluation clearly shows the benefits on the minimisation of the duration of the planning cycle and on the generation of optimized plans.

9 REFERENCES

[1] Malik Ghallab, Dana S. Nau, Paolo Traverso: Automated planning - theory and practice. Elsevier 2004, ISBN 978-1-55860-856-6, pp. I-XXVIII, 1-635

[2] Malik Ghallab, Dana S. Nau, Paolo Traverso: Automated Planning and Acting. Cambridge University Press 2016, ISBN 978-1-107-03727-4

[3] Félix Ingrand, Malik Ghallab: Deliberation for autonomous robots: A survey. Artif. Intell. 247: 10-44 (2017)

[4] https://www.ai4europe.eu

[5] https://github.com/aiplan4eu/unified-planning

[6] https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-49-2020

[7] Malte Helmert: The Fast Downward Planning System. J. Artif. Intell. Res. 26: 191-246 (2006)

[8] Alkhazraji, Y., Frorath, M., Grützner, M., Helmert, M., Liebetraut, T., Mattmüller, R., Ortlieb, M., Seipp, J., Springenberg, T., Stahl, P., & Wülfing, J. (2020). Pyperplan. Zenodo. 10.5281/zenodo.3700819

[9] Scala, Enrico, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramirez. "Interval-based relaxation for general numeric planning." In ECAI 2016, pp. 655-663. IOS Press, 2016.

[10] Gerevini, A., Saetti, A., & Serina, I. (2003): Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. Journal of Artificial Intelligence Research. 20: 239-290.

[11] Valentini, A., Micheli, A., & Cimatti, A. Temporal Planning with Intermediate Conditions and Effects. AAAI 2020, 9975-9982

[12] Torreño, A., Sapena, O., & Onaindia, E. (2014). FMAP: Distributed cooperative multi-agent planning. Applied Intelligence, 41(2), 606-626

[13] Roland Godet, Arthur Bit-Monnot. Chronicles for Representing Hierarchical Planning Problems with Time. ICAPS Hierarchical Planning Workshop (HPlan), Jun 2022

[14] Köckemann, U. (2016). Constraint-based Methods for Human-aware Planning. (Doctoral dissertation). Örebro: Örebro university

[15] https://github.com/FedericoPecora/coordination_oru

[16] https://github.com/aiplan4eu/embedded-systems-bridge

[17] Stanford Artificial Intelligence Laboratory et al. (2018). Robotic Operating System. Retrieved from https://www.ros.org

[18] https://github.com/aiplan4eu/UP4ROS and https://github.com/aiplan4eu/UP4ROS2

[19] L. Joudrier e.a.: 3DROCS - 3D Based Rover Operations Control System, ASTRA 2013

[20] Trucco et al.: ExoMars Rover Operation Control Centre Design Concept and Simulations, ASTRA 2008