Hierarchical Temporal Planning in an Earth Observation Satellite Software Architecture

Alexandre Albore
DTIS, ONERA, Université de Toulouse
31000 Toulouse, France
alexandre.albore@onera.fr

Rafael Bailon-Ruiz
DTIS, ONERA, Université de Toulouse
31000 Toulouse, France
rafael.bailon_ruiz@onera.fr

ABSTRACT

We present the use case of a commercial optical Earth Observing Satellite (EOS), with the mission of observing areas on the surface of the planet, and embedding in the on-board architecture the capacity of producing and executing autonomous plans.

The need to move on-board certain satellite functions derives from the limitations of the mission plans generated by ground control stations, and then uploaded to the EOS. Modern EOS applications include multiple acquisition requests with different degrees of priority, and need to reason taking into account information present on-board only (e.g. environment variables, the exact volume of observation data).

We propose a modular architecture for an autonomous EOS, where planning and execution deals with the arrival of urgent acquisition requests, and other relevant information, while meeting several operational requirements from the end-users. The architecture is based on OARA actors and skillsets. We then describe the model of the planning problem we are facing, and the extension of the PDDL language it uses.

KEYWORDS

Planning, Satellite, Artificial Intelligence, Robotic Architecture

ACM Reference Format:
Alexandre Albore and Rafael Bailon-Ruiz. 2024. Hierarchical Temporal Planning in an Earth Observation Satellite Software Architecture. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 10 pages.

1 CONTEXT

Earth Observing Satellites (EOSs) are usually directed from ground control in order to perform a sequence of observations of some areas of the Earth surface. However, mission plans built on the ground by human operators, and then uploaded to the space segment on a regular basis so that they can be executed while orbiting around the Earth, yield several undesired outcomes. This is in part because some information is available on-board only, and because the models used during planning can not fully project the evolution of the satellite state and the environment. For instance, the expected duration of an attitude transition may be under- or overestimated, resulting in missed data acquisition opportunities that need to be rescheduled. New generations of commercial EOSs guarantee better performances by embedding a certain degree of autonomy in the on-board architecture.

The presence of clouds impairs target visibility and cannot be fully predicted well in advance of the mission specification. As a result, images obtained using valuable satellite resources may be unusable. In addition, plans can change at any time with urgent acquisition requests: those requests must be incorporated into the mission plan with high priority, if necessary cancelling other requests. This requires a rapid response, forcing a revision of the original acquisition plan without exceeding the limits of storage and power consumption.

Prediction of cloud cover is critical to the success of satellite imagery and is at the heart of Dynamic Targeting, a method of intelligent observation scheduling that uses information from a look-ahead sensor to identify targets for a satellite's primary sensor to observe in advance [4]. The approach has been evaluated for observations scheduling algorithms, and actually can incorporate many operational constraints including slewing capability, and updates utility in repeated observations [14].

Further approaches to cloud detection could leverage recent advances in on-board cloud detection from satellite imagery, showing how a convolutional neural network has been integrated into the OPS-SAT mission [10] and in the Φ−sat mission [12] for classification of cloudy images.

Our aim is to meet the needs of end users by providing more usable data through the use of cloud coverage information and actual file sizes. We also want to be reactive and integrate the implementation of urgent requests for orbit slots autonomously into the acquisition plan. It is not feasible nowadays to completely transfer these operations to the space segment, due to limited computing resources, but it is worth taking some decisions on board to reduce uncertainty by taking advantage of temporal proximity.

The inclusion of computational capabilities on-board satellites to execute autonomous functions has been explored in numerous projects and demonstrators. This addresses the obvious environmental and resource challenges of any EOS satellite, which limit its efficiency in delivering scientific results, and increases its autonomy [3, 6, 7, 10, 12].

The Earth Observing Autonomy project considers to use processing algorithms on-board to process image data (mainly spectral analysis) to maximise science return. This project embeds the CASPER and Eagle Eye Mission Planning software to generate mission plans from observation requests. These systems are part of a wider prototype that demonstrates the feasibility of performing several functions autonomously on-board [6].

An on-board planning and execution architecture deployed at the European Space Agency (ESA) targets this need of autonomy for the OPS-SAT mission [10]. This architecture, which is similar in

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.

spirit to the one proposed in this paper, integrates a model-based, domain independent planner within an executive architecture.

Using the information available only on board the EOSs, an embedded automated planner is expected to generate observation mission plans in a closed plan/replan loop as the environment changes or new requests are received from ground control. New requests for the allocation of orbital slots to fly over desired regions on Earth require an optimisation approach, which has been shown to be well suited to a Constraint Programming formulation for exploring different slot allocation alternatives [18]. Also, the planning process of generating new observation plans to fit orbit slot exclusivity requests and the state of the environment —which is not known in advance— requires to be integrated within a comprehensive software architecture in charge of orchestrating planning and execution.

Contemporary on-board software architectures follow a three-layer design to facilitate design, robustness and extensibility [26]:

(1) A decisional layer in charge of planning and monitoring spacecraft activity according to the objectives set on ground.
(2) An operational layer in charge of the executing the activity plan defined in the upper layer with high-level commands.
(3) A functional layer for low-level supervision and control of the spacecraft subsystems.

The work we present here gets inspiration from [22], but we go a step forward. First, we propose exploiting on-board cloud cover observations instead of forecasts made on the ground to anticipate target visibility with a wide angle camera looking ahead along the satellite path. Second, the reactor-based architecture with timelines, derived from T-REX [19], has been replaced by the OARA actors [15] and skillsets [16] framework. This framework has proven useful for robotic applications [1, 2], but it can also be applied to autonomous space systems too.

In addition to the EOS architecture, we propose a way to tackle the planning problem for this satellite use case. The type of planning performed includes hierarchical aspects that depend on the architecture of the sensors and actuators in the EOS, but also contains temporal elements due to the constraints of observing certain areas of the planet.

Outline: The next section describes the proposed satellite platform and its on-board architecture. Then, we propose an extension of the PDDL language to formulate the planning problem for this satellite use case. We conclude on a discussion on future extensions of this application.

2 EOS ON-BOARD PLANNING AND EXECUTION ARCHITECTURE

The proposed satellite architecture has three layers, as depicted in Figure 1. Namely, a *decision layer*, a *skillset layer* and a *functional layer*:

(1) The *decision layer* hosts the mission controller. It is implemented upon OARA actors [15], a framework to develop decision-making architectures for autonomous agents. The OARA framework model revolves around the idea of a hierarchy of actors managing goals by decomposition into Partial Order Plans. The sub-goals in the selected plan are dispatched to the appropriate child actors and monitored for execution reports. Plans can be set up to be repaired, replaced or dropped if a subgoal execution fails, or if a parent actor signals that the current goal is obsolete, i.e. urgent requests arrive.
(2) The *skillset layer* defines an abstract interface of the satellite capabilities via *skills* [2, 16]: the basic platform behaviours for autonomous mission execution with planning at the level of the task. Skills formally define their behaviour, the resources they employ and the inputs they require to run, so that the skillset layer can continue to operate in the face of interruptions and abnormal situations at the logic level.
(3) The *functional layer* implements the on-board control procedures that actually operate the spacecraft, either directly or trough a skill.

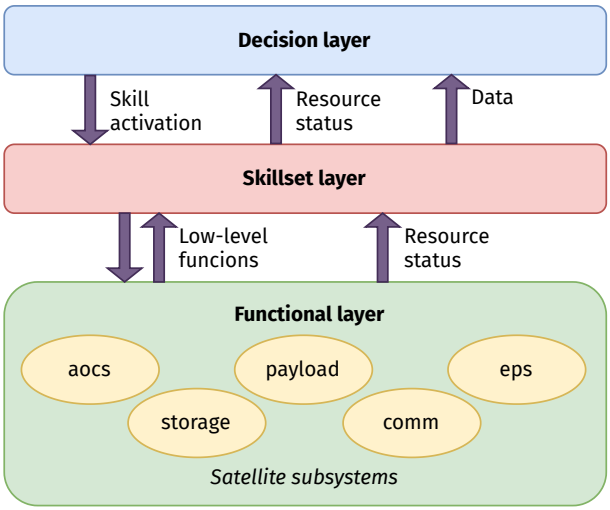

Figure 1: A three layer skills-based architecture for autonomy

OARA actors and skillsets are built upon the ROS 2 middleware, as well as the simulated functional layer interface. Note that it is possible to combine real components with simulated ones, for faster development and verification, as the physical design doesn't have to be finalised before developing the mission control software.

2.1 Requests formulation

The EOS mission goal is to fulfil a set of *Programming Requests* by taking images of portions of the Earth surface.

Definition 1 (Programming Request (PR)). *A Programming Request (AR) pr_i is a region of the Earth surface which is required to be fully covered by observations. It is characterised by:*

- *a date of deposit* t_i^{dep},
- *a validity period* $[t_i^{start}, t_i^{end}]$
- *the area of interest* AOI_i
- *an urgency flag* u_i
- *a cloud cover threshold* c_{max},
- *a maximum viewing angle* ψ_{max}

Programming requests can require to cover areas much larger than the satellite can observe at one time. Therefore, as shown

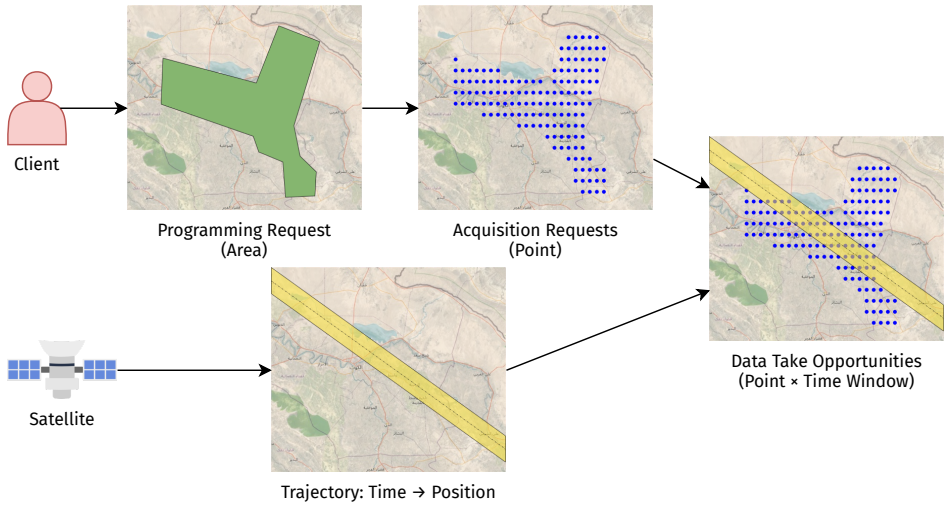

Figure 2: Illustration of the requests formulation model. Programming requests (PRs) are divided into acquisition requests (ARs) that can be observed by the satellite during data take opportunities (DTOs).

in Figure 2, they are decomposed into smaller chunks which can be photographed individually. The set of all the PRs describing a mission is denoted $\mathcal{P}$.

Definition 2 (Acquisition Request (AR)). *An Acquisition Request (AR) $ar_{i,j}$ is the portion of pr_i, such that an area of interest $AOI_{i,j} \subseteq AOI_i$ can be observed in a single satellite image, and for which the validity period is $[t_{i,j}^{start}, t_{i,j}^{end}] \in [t_i^{start}, t_i^{end}]$. It denotes the acquisition of a target position inside the PR area of interest.*

A programming request is fulfilled by completing all its associated acquisition requests, covering the complete PR area of interest. The satellite may require multiple passes to complete a Programming Request.

Definition 3 (Data take opportunity (DTO)). *The time window $[t^{lower}, t^{upper}]$ during which an acquisition request is valid for the satellite is called a data acquisition opportunity. It is denoted by $dto_{i,j}$ for a given acquisition request $ar_{i,j}$.*

Definition 4 (Mission DTOs). *The Mission DTOs is the set of the DTOs corresponding to the PRs the satellite has to complete. It is then described as the collection of DTOs s.t. $\{dto_{i,j} \mid [t_{i,j}^{lower}, t_{i,j}^{upper}] \subseteq [t_{i,j}^{start}, t_{i,j}^{end}]$, for all $ar_{i,j} \in pr_i$ and all $pr_i \in \mathcal{P}\}$.*

Note that there may be multiple opportunities k for any AR as the satellite passes by the same area while orbiting the Earth.

Finding Mission DTOs is computationally expensive due to the complexity of Earth's and satellite orbital dynamics. The calculation is not feasible on-board, and ground control must specify them.

2.2 Satellite description and functional layer

The satellite we consider, devoted to the observation of the Earth's surface, is moving in a low-earth orbit ($\approx$ 500 km) with low inclination (latitude $< 50°$) to increase revisits. The satellite is agile, that is it can rotate forward and backwards in addition to left and right with respect to its track. We assume a constant mean rotation speed to compute the duration of pointing actions, but a more accurate model could be used without invalidating the current design.

The satellite is equipped with two optical systems located nadir: a narrow field of view camera for observing ground targets and a wide field of view sensor that serves as a look-ahead camera. The latter can be positioned to observe the area ahead of the orbit to detect the presence of clouds or upcoming interesting targets.

Our satellite model simulates energy and memory usage. Batteries power the satellite while solar panels recharge them during sunlight. Images of the requested targets can be stored and retrieved from the storage module as required. They will be deleted once they have been successfully downloaded to the ground station.

The communications module is responsible for managing data transfer operations between the spacecraft and ground stations. Payload data can be downloaded when the satellite is in visibility of dedicated stations. The set of targets to be observed are uploaded in the form of the mission specification including PRs and Mission DTOs.

The functional layer has been implemented in the form of a ROS 2 package that emulates the behaviour of the satellite. Each subsystem of the spacecraft runs as a ROS 2 node written in the Python programming language. The `aocs` emulates attitude and orbit control, the `storage` node manages picture and requests storage, the `comm` node simulates data transfer, the `eps` node keeps track of energy consumption and generation and the `camera` node emulates mimics the behaviour of embedded cameras. Nodes exchange information by subscribing and publishing data to named buses called topics. As the purpose of the simulation is to validate the soundness of a novel EOS planning paradigm, the current satellite simulator implementation uses simple models that do not emulate precisely all the functions of an EOS the but capture the overall behaviour of the spacecraft. The simulation architecture comes with a supervision graphical user interface, shown in Figure 3, that displays the state of the satellite.

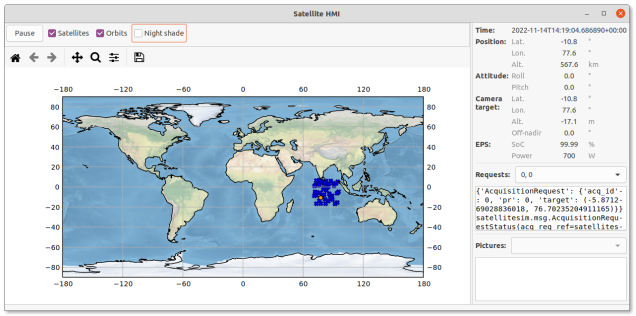

Figure 3: Screenshot of the satellite simulator supervision graphical user interface.

2.3 The OARA goal lifecycle model and actors

Goals received by an OARA actor go through a lifecycle state-machine following the goal reasoning process introduced in [24].

A goal is always in one of the following six decision steps: *Formulated*, *Selected*, *Expanded*, *Committed*, *Dispatched* and *Evaluated*. As illustrated in Figure 5, a goal starts in the *Formulated* state when the actor receives it. Then it moves to *Selected* if it can be attempted, that is the goal formulation is within the scope of the agent planning and execution capabilities. Next, the goal goes through a planning phase, where one or more decompositions in the form of Partial Order Plans are proposed but only one is picked (*Expanded* and *Committed* states respectively). Then, sub-goals are dispatched to corresponding actors according to ordering and time constraints. Once the *Dispatched* state is reached by the goal, a periodic monitor function supervises the execution. Upon evaluation, several actions can be triggered: *Continue* with the next sub-goal, *Repair* (commit to a different plan), *Replan* (expand the goal again), *Defer*, (go back to selection), or *Reform* if the goal cannot be executed anymore. Actors can also receive and send a *Drop* request that cancels the current goal and drops active sub-goals.

In most cases, a full goal lifecycle is too broad for the application. As such, OARA offers two lifecycle patterns that are particularly useful for an autonomous satellite. That is controller actors, and plan/replan behaviours based on Temporal Partial-Order Schedules (Timed POS).

Controllers are the actors at the lowest level of the deliberative architecture. They cannot create sub-goals and delegate them to other actors, but instead manage one goal execution with a direct interface to the robot functional layer. The *Expand* and *Commit* transitions do nothing specific, as no decomposition must be computed. Also, the controller cannot process external events but only reacts to execution reports from lower architectural layers, handled in the *Monitor* transition. The sole possible terminal statuses are then to *Finish* the goal in case of success, or to *Reform* it in case of failure.

Timed POS actors manage plans whose goals are partial-order scheduled and consider that goals have an optional dispatch time. They behave as POS actors regarding the precedence checks of sub-goals, but instead of dispatching immediately, the actors set up a timer that delays the execution of sub-goals. These actors are useful for plans that contain time-constrained actions. They can also handle hybrid plans with timed and not timed sub-goals.

Observers are particular OARA modules, different from actors, whose purpose is to monitor data exposed by the functional layer and information produced by actors. Actors can get the current value of pieces of data as they please and update values for other actors to use.

2.4 Decision layer architecture for mission control

The decision layer implements a mission controller using the OARA framework, generating plans to complete acquisition goals and supervises the execution.

The on-board decision architecture managing the acquisition problem is composed of 11 actors and 3 observers. Figure 4 depicts the hierarchy of actors and their interaction with the skillset layer and the ground station. More in the details:

The *Ground Segment Mission* actor encompasses and simulates the ground segment of the satellite system. It registers the programming requests and computes their decomposition into acquisition requests and data take opportunities. The PRs and ARs are loaded into the satellite order book, via the storage observer, alongside the DTOs. The Ground Segment Mission actor then formulates a goal to the mission actor that includes the set of DTOs that should be observed.

The *Mission* actor manages incoming observation requests. It also orchestrates the cloud detection, acquisition, and download actors deciding when to execute them and in which conditions. It produces an acquisition subgoal based on the requests that can be safely carried out according to the required maximum cloud presence and the actual cloud cover in the area as reported by the AR detector actor. When an urgent request for an orbit slot is received, the Ground Segment Mission actor drops the current goal of the Mission actor and revises the mission specification to incorporate the new request.

The *AR detector* actor orchestrates cloud detection and determines which targets are visible or obscured by clouds. This check is actually done by the *Cloud Detector* controller, which handles the cloud camera after being positioned by the *Rotate* controller. The visibility status of targets can be accessed through the *Mission* observer. The focus of this study is on the architecture and interfaces for integrating a cloud detection process into the acquisition mission, therefore the process is modelled at a coarse level. Currently, the *Cloud Detector* controller randomly filters which DTOs are classified as being covered by clouds.

The *Acquisition Planner* actor takes as input a set of data take opportunities and finds a suitable observation plan. To ensure that urgent requests are included in the plan, non-urgent requests with conflicting DTOs are ignored. This verification step is carried out by a Simple Temporal Network (STN) [8], which incorporates all temporal constraints of the requests and can determine several key properties in polynomial time, such as the global consistency of the network [5, 21, 23]. The plan is a temporal sequence of data takes, expected realisations of data take opportunities. Each data take in the plan is executed by the *Acquisition* Timed POS Actor which formulates rotation and image capturing goals at precise times. The

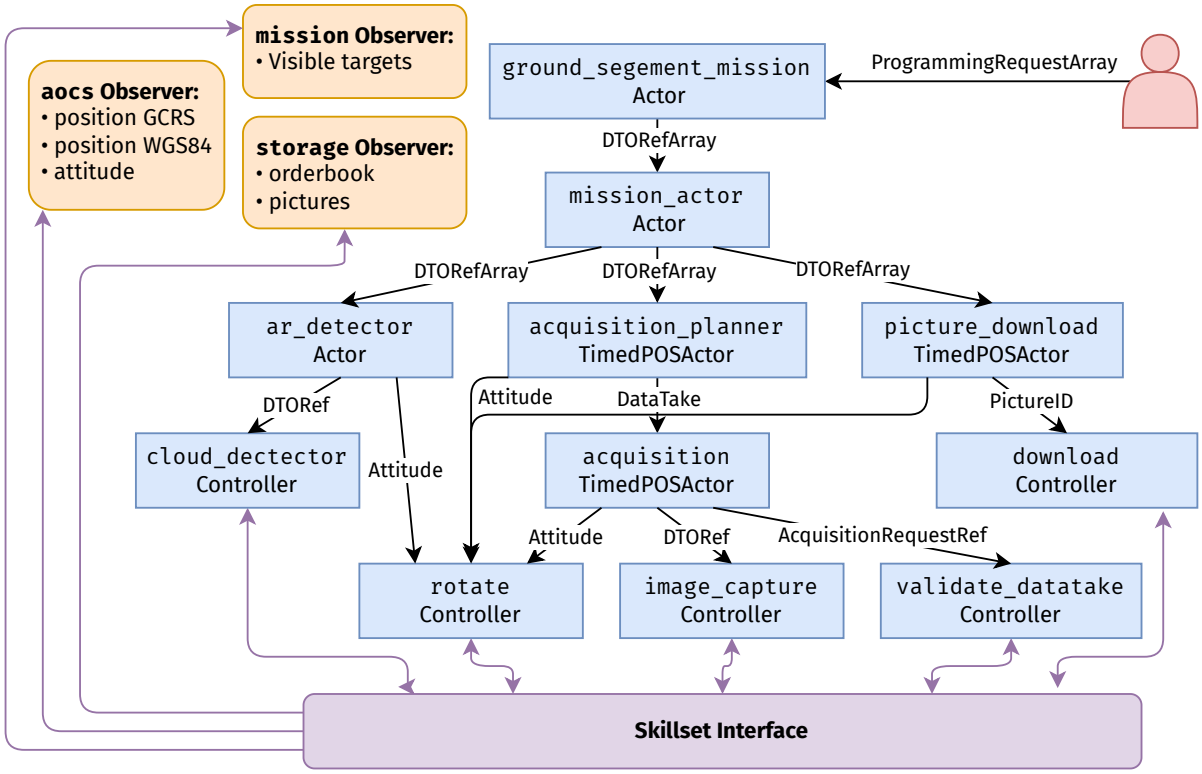

Figure 4: The decision architecture components and their hierarchical relationship. Orange boxes are observers, blue boxes are actors and black arrows denote possible goal decomposition with their type.

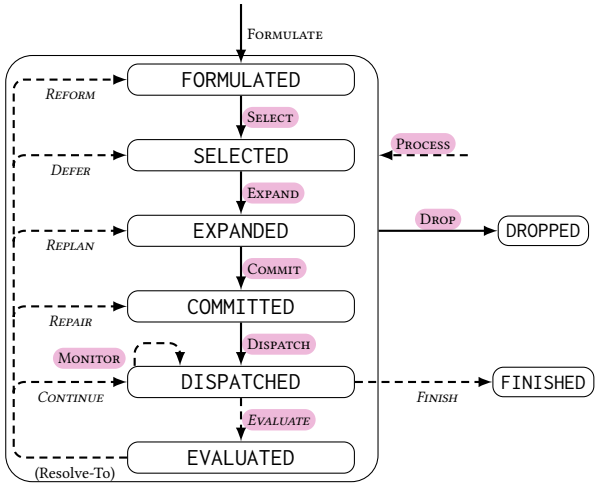

Figure 5: The OARA goal lifecycle. Plain transitions are requests coming from a parent actor. Dashed transitions are internal to the actor. Transitions highlighted in pink must be implemented by the actor developer, whereas transitions with italic labels are only triggered in the developer's code.

Rotate and *Image Capture* controllers manipulate the satellite AOCS and the camera payload by activating the appropriate skills. The *Validate* controller checks the quality of the acquired image and updates in the order book the completion state of the associated AR and PR.

Finally, the *Picture Download* actor orchestrates payload data download and formulates goals to the Rotate controller to point to the ground station and *Download* controller to perform the simulated communication. As the download problem has not been considered yet, the downloading procedure trivially sends all images back to the ground stations.

3 HIERARCHICAL TEMPORAL PLANNING FOR EOS

In order to reflect the temporal constraints present in the Acquisition Request schema, we adopt the HDDL 2.1 problem formulation [20], that includes elements of temporal planning borrowed from PDDL 2.2 [9] besides the usual HDDL formulation for expressing hierarchical planning problems [13]. Note that the action definition of this sort of "temporal HDDL" has then been modified to deal with durative-action definition as in PDDL 2.2. The definition of the temporal effects of the actions remains unchanged in HDDL 2.1. Only the numerical aspects have been removed as our in-house planner doesn't manage them yet. These choices are reflected by the following requirement flags, which are also compatible with HDDL:

- `:durative-actions` requires the applied system needs to support durative actions and temporal ordering constraints in method definition.
- `:duration-inequalities` requires the applied system needs to support duration inequalities in durative actions declaration. Implies `:durative-actions`.
- `:timed-initial-literal` requires the applied system needs to support initial state with literal that becomes true at a specified time point. Implies `:durative-actions`.

Besides, we add the following requirement flags:

- `:method-constraints` requires the need to support method decomposition constraints.

3.1 The satellite HDDL 2.1 model

We model our planning problem on the hierarchical satellite domain [25], tailored to a planning model that takes in account the specificities of our observation problem. Besides the requirements discussed in the previous section, the domain specifies the methods and actions needed to perform the data takes and the image capture tasks based on a set of available predicates and functions. A sample planning problem with several image targets is provided in Appendix A as well as the corresponding HDDL 2.1 domain.

In this representation, we reuse part of the *durative* action specifications from the original non-hierarchical PDDL description [17], namely: `turn_to`, `switch_on`, `switch_off`, and `take_image`. Some hierarchical aspects are present in the intended procedure for taking an image, where an instrument has to be selected, switched on, and switched off after the image capture[1]. This is reflected in the collection of methods available to execute the `do_observation`, and `activate_instrument` tasks.

The camera instrument can be in ON or OFF states and they must be ON to be used. This can be achieved in two ways: either by powering on the instrument after powering off an already activated one (`method4`) or, if any instrument is powered on, just activating the desired sensor (`method5`). Observations can be done with four alternative methods: `method0` activates the sensor, then points to the target and finally takes the image. `method1` and `method2` rely on being already pointing to the right direction and the instrument being on respectively. `method3` assumes everything is set up and simply taking an image is sufficient to execute the task.

The action specifications have a duration to reflect the warming up of certain instruments and the time to modify the attitude of the satellite, and a validity period $[t^{start}, t^{end}]$ corresponding to the data take opportunity. Timed initial literals are used to designate the validity period of each area of interest by making the `observable` predicate true or false. Also, the `turn-time` function returns the time require to rotate the satellite between image directions but as the actual computation of this duration is complex, pre-computed values are given in the problem description.

3.2 Preliminary experiments

A custom-made planner is used inside the acquisition planner actor. This planning solution uses depth-first search with a bounded temporal horizon. Given a set of visible DTOs within this horizon,

[1]Please note that, unlike the original satellite domain, we do not consider instrument calibration in this model.

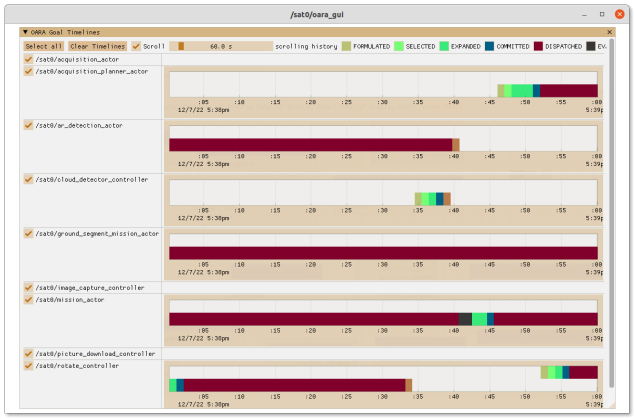

Figure 6: The OARA supervision interface displaying the activation of the satellite decision layer actors.

the objective is to find the optimal sequence of observations the maximise the number of targets acquired with priority granted to urgent tasks and abiding by the satellite physical constraints. Incidentally, the resulting plan minimises energy consumption as reducing the time passed changing orientations generally allows for more observations to be done in the same time span. Currently, a more developed version of a hierarchical temporal planner has been implemented and parses HDDL 2.1 domains and problems; efforts are currently being made to develop and improve such planner.

The architecture has been tested with the satellite simulator depicted in subsection 2.2 and the complete implementation of the decision layer from subsection 2.4. The Ground Segment Mission actor has been provided with a list of programming requests as shown in the Figure 3 as blue markers. The OARA actors in the decision layer decomposed and executed the tasks as illustrated in Figure 6. Each segment in the OARA supervision interface denotes the activation of a particular actor with a colour code corresponding to the state of the goal e.g. FORMULATED, SELECTED, etc.

4 ONGOING AND FUTURE WORK

We have presented our implementation of the use case of an autonomous optical Earth Observing Satellite using information present on-board only to plan and execute observation plans. The embedded mission control software architecture, implemented using the OARA actors and skillsets, integrates, in its observation plan, the cloud cover maps obtained from a forward-facing camera, and the arrival of urgent requests. We have proposed the usage of HDDL 2.1, a temporal extension to HDDL to express temporal constraints, timed initial literals, and action durations, so to model the planning problem relative to this use case.

The decision layer based on the OARA framework and supported by OARA skillsets adds a modular dimension to the architecture by dissociating the abstract capabilities of the satellite platform from the actual hardware. This permits to combine real components with simulated ones, for faster development and verification. This approach is even more interesting in the development phase, as the physical design doesn't have to be finalised before developing

the mission control software. In the future, we expect to be able to rely on the modularity of the actor-based architecture paradigm to solve the download planning problem. This problem shares analogies with the acquisition, instead of trivially transmitting all the images. To do so, the *Mission* actor would need to be tailored to set a memory allowance for the *Acquisition Planner* actor. Now, in order to produce effective acquisition plans, the Mission controller filters the planner's input with the information about cloud coverage. Better plans could be produced by exploiting more resource information that can only be known on-board, like the real volumes of data embedded in the pictures and the battery level. The usage of these variables in the planning phase require the numeric fluents introduced in HDDL 2.1, which were excluded in the first version of HDDL due to efforts to develop the language and related tools

The introduction of PDDL3 preferences [11] would allow to distinguish acquisitions qualities at various viewing angles ψ in a DTO time window by counting the number of times a constraint expressing an acquisition quality has been violated. However, a precise cost values according to the customer-specific requirements is complex to formulate. Preferences could be used as well to distinguish potential AR priority levels, solving DTOs conflict and urgent acquisitions placement. These enhancement of the model should allow for more fine grained insertions of urgent requests too.

A particular aspect of EOS mission goals is that missed targets can not be revisited immediately as it takes multiple orbital periods before flying past the same area of interest. This is critical with targets located in areas suffering from frequent adverse weather where valid acquisitions can be elusive. Thus, a constellation of autonomous satellites, in specific orbit configurations to increase the frequency of revisits, is required to improve the probability of acquiring a target. In a changing environment, achieving a common goal necessitates cooperation between satellites, besides the individual autonomy. Designing such a system is not trivial as communication is low-bandwidth and intermittent. Future work will consider the decision architecture for a satellite within such a constellation.

ACKNOWLEDGMENTS

This work has been performed with the support of BPI through PSPC project "LiChIE" of the "Programme d'Investissements d'Avenir" and ONERA PRF ACCEOS.

REFERENCES

[1] Alexandre Albore, David Doose, Christophe Grand, Jérémie Guiochet, Charles Lesire, and Augustin Manecy. 2022. Skill-based design of dependable robotic architectures. *Robotics and Autonomous Systems* 160 (Nov. 2022). Issue 104318. https://doi.org/10.1016/j.robot.2022.104318

[2] Alexandre Albore, David Doose, Christophe Grand, Charles Lesire, and Augustin Manecy. 2021. Skill-Based Architecture Development for Online Mission Reconfiguration and Failure Management. In *2021 IEEE/ACM 3rd International Workshop on Robotics Software Engineering (RoSE)*. IEEE, Madrid, Spain, 47–54. https://doi.org/10.1109/RoSE52553.2021.00015

[3] Carles Araguz, Elisenda Bou-Balust, and Eduard Alarcón. 2018. Applying autonomy to distributed satellite systems: Trends, challenges, and future prospects. *Systems Engineering* 21, 5 (2018), 401–416.

[4] Alberto Candela, Jason Swope, and Steve A Chien. 2023. Dynamic targeting to improve earth science missions. *Journal of Aerospace Information Systems* 20, 11 (2023), 679–689.

[5] Roberto Cervoni, Amedeo Cesta, and Angelo Oddi. 1994. Managing Dynamic Temporal Constraint Networks.. In *AIPS*. 13–18.

[6] Steve Chien, Russell Knight, Steve Schaffer, David Ray Thompson, Brian Bue, and Martina Troesch. 2015. An Onboard Autonomous Response Prototype for an Earth Observing Spacecraft. In *International Joint Conference on Artificial Intelligence Workshop on Artificial Intelligence in Space (AI Space, IJCAI 2015)*. Buenos Aires, Argentina. https://ai.jpl.nasa.gov/public/papers/chien-ijcai2015-autonomous.pdf

[7] Bharadwaj Chintalapati, Arthur Precht, Sougata Hanra, Rene Laufer, Marcus Liwicki, and Jens Eickhoff. 2024. Opportunities and challenges of on-board AI-based image recognition for small satellite Earth observation missions. *Advances in Space Research* (2024). https://doi.org/10.1016/j.asr.2024.03.053

[8] Rina Dechter, Itay Meiri, and Judea Pearl. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49, 1-3 (1991), 61–95.

[9] Stefan Edelkamp and Jörg Hoffmann. 2004. *PDDL2.2: The language for the classical part of the 4th international planning competition.* Technical Report. Technical Report 195, University of Freiburg.

[10] Simone Fratini, Nicola Policella, Ricardo Silva, and Joao Guerreiro. [n.d.]. On-Board Autonomy Operations for OPS-SAT Experiment. 52, 6 ([n. d.]), 6970–6987. https://doi.org/10.1007/s10489-020-02158-5

[11] Alfonso Gerevini and Derek Long. 2006. Preferences and Soft Constraints in PDDL3. In *ICAPS Workshop on Planning with Preferences and Soft Constraints*. 46–53.

[12] Gianluca Giuffrida, Luca Fanucci, Gabriele Meoni, Matej Batič, Léonie Buckley, Aubrey Dunne, Chris Van Dijk, Marco Esposito, John Hefele, Nathan Vercruyssen, et al. 2021. The Φ-Sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation. *IEEE Transactions on Geoscience and Remote Sensing* 60 (2021), 1–14.

[13] Daniel Höller, Gregor Behnke, Pascal Bercher, Susanne Biundo, Humbert Fiorino, Damien Pellier, and Ron Alford. 2020. HDDL: An extension to PDDL for expressing hierarchical planning problems. 34, 6 (2020), 9883–9891.

[14] Akseli Kangaslahti, Alberto Candela, Jason Swope, Qing Yue, and Steve Chien. 2024. Dynamic Targeting of Satellite Observations Incorporating Slewing Costs and Complex Observation Utility. In *IEEE International Conference on Robotics and Automation (ICRA 2024)*. Yokohama, Japan. https://ai.jpl.nasa.gov/public/documents/papers/Kangaslahti_DT_ICRA_2024.pdf

[15] Charles Lesire, Rafael Bailon-Ruiz, Magali Barbier, and Christophe Grand. 2022. A Hierarchical Deliberative Architecture Framework based on Goal Decomposition. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*. Kyoto, Japan.

[16] Charles Lesire, David Doose, and Christophe Grand. 2020. Formalization of Robot Skills with Descriptive and Operational Models. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA. https://doi.org/10.1109/IROS45743.2020.9340698

[17] Derek Long and Maria Fox. 2003. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research* 20 (2003), 1–59.

[18] Sara Maqrot, Stéphanie Roussel, Gauthier Picard, and Cédric Pralet. 2022. Orbit Slot Allocation in Earth Observation Constellations.. In *PAIS*. 3–16.

[19] Conor McGann, Frederic Py, Kanna Rajan, Hans Thomas, Richard Henthorn, and Rob Mcewen. 2008. A Deliberative Architecture for AUV Control. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, 1049–1054.

[20] Damien Pellier, Alexandre Albore, Humbert Fiorino, and Rafael Bailon-Ruiz. 2023. HDDL 2.1: Towards Defining a Formalism and a Semantics for Temporal HTN Planning. In *Proceedings of the International Workshop of Hierarchical Planning (ICAPS)*. Prague.

[21] Léon Planken, Mathijs de Weerdt, and Neil Yorke-Smith. 2010. Incrementally solving STNs by enforcing partial path consistency. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 20.

[22] Cédric Pralet, Charles Lesire, and Jean Jaubert. 2019. An Autonomous Mission Controller for Earth Observing Satellites. In *IWPSS 2019*.

[23] Cédric Pralet and Gérard Verfaillie. 2013. Time–dependent Simple Temporal Networks: Properties and Algorithms∗. *RAIRO-Operations Research* 47, 2 (2013), 173–198.

[24] Mark Roberts, Vikas Shivashanka, Ron Alford, Michael Leece, Shubham Gupta, and David W. Aha. 2016. Goal Reasoning, Planning, and Acting with ACTOR SIM, The Actor Simulator. In *Annual Conference on Advances in Cognitive Systems*. Evanston, IL, USA.

[25] Bernd Schattenberg. 2020. The Hierarchical Satellite Domain. In *10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track*. Nancy, France (Online), 40.

[26] M. Tipaldi and L. Glielmo. 2018. A Survey on Model-Based Mission Planning and Execution for Autonomous Spacecraft. *IEEE Systems Journal* 12, 4 (Dec. 2018), 3893–3905. https://doi.org/10.1109/JSYST.2017.2720682

A SATELLITE HDDL 2.1 DOMAIN AND SAMPLE PROBLEM

```
(define (domain satellite2)
  (:requirements
    :durative-actions
    :equality
    :negative-preconditions
    :typing
    :numeric-fluents
    :timed-initial-literals
    :method-constraints
    :hierarchy)

  (:types
    image_direction
    instrument
    satellite
    mode
  )

  (:predicates
    (on_board ?arg0 - instrument ?arg1 - satellite
        )
    (supports ?arg0 - instrument ?arg1 - mode)
    (pointing ?arg0 - satellite ?arg1 -
        image_direction)
    (power_avail ?arg0 - satellite)
    (power_on ?arg0 - instrument)
    (have_image ?arg0 - image_direction ?arg1 -
        mode)
    (observable ?arg0 - image_direction)
  )

  (:functions
    (turn-time ?d1 - image_direction ?d2 -
        image_direction)
  )

  (:task do_observation
    :parameters (?do_d - image_direction ?do_m -
        mode)
  )

  (:task activate_instrument
    :parameters (?ai_s - satellite ?ai_i -
        instrument)
  )

  (:method method0
    :parameters (?mdoatt_t_d_prev -
        image_direction ?mdoatt_t_s - satellite ?
        mdoatt_ti_d - image_direction ?mdoatt_ti_i
         - instrument ?mdoatt_ti_m - mode)
    :task (do_observation ?mdoatt_ti_d ?
        mdoatt_ti_m)
    :subtasks (and
      (task0 (activate_instrument ?mdoatt_t_s ?
          mdoatt_ti_i))
      (task1 (turn_to ?mdoatt_t_s ?mdoatt_ti_d ?
          mdoatt_t_d_prev))
      (task2 (take_image ?mdoatt_t_s ?mdoatt_ti_d
          ?mdoatt_ti_i ?mdoatt_ti_m))
    )
    :ordering (and
      (< task0 task1)
      (< task1 task2)
    )
    :constraints (and
      (not (= ?mdoatt_ti_d ?mdoatt_t_d_prev))
    )
  )

  (:method method1
    :parameters (?mdott_t_d_prev - image_direction
        ?mdott_t_s - satellite ?mdott_ti_d -
        image_direction ?mdott_ti_i - instrument ?
        mdott_ti_m - mode)
    :task (do_observation ?mdott_ti_d ?mdott_ti_m)
    :subtasks (and
      (task0 (turn_to ?mdott_t_s ?mdott_ti_d ?
          mdott_t_d_prev))
      (task1 (take_image ?mdott_t_s ?mdott_ti_d ?
          mdott_ti_i ?mdott_ti_m))
    )
    :ordering (and
      (< task0 task1)
    )
    :constraints (and
      (not (= ?mdott_ti_d ?mdott_t_d_prev))
    )
  )

  (:method method2
    :parameters (?mdoat_ti_d - image_direction ?
        mdoat_ti_i - instrument ?mdoat_ti_m - mode
        ?mdoat_ti_s - satellite)
    :task (do_observation ?mdoat_ti_d ?mdoat_ti_m)
    :subtasks (and
      (task0 (activate_instrument ?mdoat_ti_s ?
          mdoat_ti_i))
      (task1 (take_image ?mdoat_ti_s ?mdoat_ti_d ?
          mdoat_ti_i ?mdoat_ti_m))
    )
    :ordering (and
      (< task0 task1)
    )
  )

  (:method method3
    :parameters (?mdot_ti_d - image_direction ?
        mdot_ti_i - instrument ?mdot_ti_m - mode ?
        mdot_ti_s - satellite)
    :task (do_observation ?mdot_ti_d ?mdot_ti_m)
```

```
    :subtasks (and
      (task0 (take_image ?mdot_ti_s ?mdot_ti_d ?mdot_ti_i ?mdot_ti_m))
    )
  )

  (:method method4
    :parameters (?maissa_ac_i - instrument ?maissa_ac_s - satellite ?maissa_sof_i - instrument)
    :task (activate_instrument ?maissa_ac_s ?maissa_ac_i)
    :subtasks (and
      (task0 (switch_off ?maissa_sof_i ?maissa_ac_s))
      (task1 (switch_on ?maissa_ac_i ?maissa_ac_s))
    )
    :ordering (and
      (< task0 task1)
    )
    :constraints (and
      (not (= ?maissa_sof_i ?maissa_ac_i))
    )
  )

  (:method method5
    :parameters (?maisa_ac_i - instrument ?maisa_ac_s - satellite)
    :task (activate_instrument ?maisa_ac_s ?maisa_ac_i)
    :subtasks (and
      (task0 (switch_on ?maisa_ac_i ?maisa_ac_s))
    )
  )

  (:durative-action turn_to
    :parameters (?t_s - satellite ?t_d_new - image_direction ?t_d_prev - image_direction)
    :duration (= ?duration (turn-time ?t_d_new ?t_d_prev))
    :condition (and
      (at start (not (= ?t_d_new ?t_d_prev)))
      (at start (pointing ?t_s ?t_d_prev))
    )
    :effect (and
      (at end (pointing ?t_s ?t_d_new))
      (at end (not (pointing ?t_s ?t_d_prev)))
    )
  )

  (:durative-action switch_on
    :parameters (?so_i - instrument ?so_s - satellite)
    :duration  (= ?duration 1)
    :condition
      (and
        (at start (on_board ?so_i ?so_s))
        (at start (power_avail ?so_s))
      )
    :effect (and
      (at end (power_on ?so_i))
      (at end (not (power_avail ?so_s)))
    )
  )

  (:durative-action switch_off
    :parameters (?sof_i - instrument ?sof_s - satellite)
    :duration  (= ?duration 1)
    :condition (and
      (at start (on_board ?sof_i ?sof_s))
      (at start (power_on ?sof_i))
    )
    :effect (and
      (at end (not (power_on ?sof_i)))
      (at end (power_avail ?sof_s))
    )
  )

  (:durative-action take_image
    :parameters (?ti_s - satellite ?ti_d - image_direction ?ti_i - instrument ?ti_m - mode)
    :duration (= ?duration 2)
    :condition (and
      (over all (observable ?ti_d))
      (at start (pointing ?ti_s ?ti_d))
      (over all (on_board ?ti_i ?ti_s))
      (over all (power_on ?ti_i))
      (at start (supports ?ti_i ?ti_m))
    )
    :effect (and
      (at end (have_image ?ti_d ?ti_m))
    )
  )
)
```

Listing 1: The satellite HDDL 2.1 domain

```
(define
  (problem sat2_problem)
  (:domain satellite2)
  (:objects
    instrument0 - instrument
    instrument1 - instrument
    satellite0 - satellite
    infrared0 - mode
    spectrograph1 - mode
    infrared2 - mode
    site1 - image_direction
    site2 - image_direction
    site3 - image_direction
```

```
    site4 - image_direction
    site5 - image_direction
  )
  (:htn
    :parameters ()
    :subtasks (and
      (task0 (do_observation site2 infrared2))
      (task1 (do_observation site3 infrared2))
      (task2 (do_observation site4 infrared0))
      (task3 (do_observation site5 infrared2))
    )
  )
  (:init
    (supports instrument0 infrared2)
    (supports instrument0 spectrograph1)
    (supports instrument0 infrared0)
    (supports instrument1 infrared2)
    (supports instrument1 spectrograph1)
    (on_board instrument0 satellite0)
    (on_board instrument1 satellite0)
    (power_avail satellite0)
    (pointing satellite0 site1)
    (= (turn-time site1 site2) 149)
    (= (turn-time site1 site3) 513)
    (= (turn-time site1 site4) 704)
    (= (turn-time site1 site5) 681)
    (= (turn-time site2 site1) 149)
    (= (turn-time site2 site3) 370)
    (= (turn-time site2 site4) 621)
    (= (turn-time site2 site5) 558)
    (= (turn-time site3 site1) 513)
    (= (turn-time site3 site2) 370)
    (= (turn-time site3 site4) 678)
    (= (turn-time site3 site5) 490)
    (= (turn-time site4 site1) 704)
    (= (turn-time site4 site2) 621)
    (= (turn-time site4 site3) 678)
    (= (turn-time site4 site5) 237)
    (= (turn-time site5 site1) 681)
    (= (turn-time site5 site2) 558)
    (= (turn-time site5 site3) 490)
    (= (turn-time site5 site4) 237)
    (at 500 (observable site1))
    (at 1000 (not (observable site1)))
    (at 5 (observable site2))
    (at 500 (not (observable site2)))
    (at 450 (observable site3))
    (at 1050 (not (observable site3)))
    (at 500 (observable site4))
    (at 2500 (not (observable site4)))
    (at 1050 (observable site5))
    (at 2500 (not (observable site5)))
  )
)
```

Listing 2: A sample satellite HDDL 2.1 problem