GSOC's Planning Library: History, Generic Features and Lessons Learnt

Christoph Lenzen, Maria Th. Wörle, Sven Prüfer, Martin Wickler, Anna Fürbacher
Mission Technology Department, German Space Operations Center (GSOC)
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Münchener Straße 20
82234 Weßling, Germany

Abstract

Mission Planning at GSOC started, in cooperation with other agencies, with manually triggered processes. Within the mission *D-2*, first experiences have been gathered with the *Experiment Scheduling Program* of the Marshall Space Flight Center. For succeeding missions, the interactive planning application *Pinta* has been developed, together with additional tools which support event calculation and automated planning using simple heuristics. A major step forward was the implementation of a fully automated planning system for *TerraSAR-X*, where it was in charge of the whole mission, including payload and bus. Soon this *Mission Planning* system had been extended to also include a second satellite and additional mission goals for the *TanDEM-X* mission. In preparation of a successor mission, desires of internal and external users and operators of the *TerraSAR-X/TanDEM-X* missions have been analyzed. Even though no successor mission for *TerraSAR-X* has been selected yet, the *Mission Planning* team evolved its planning libraries according to the outcome of this analysis and to respond to further lessons learnt, which had been gathered in different other missions throughout the years, such as *FireBird*, *EDRS*, *Galileo* and several LEOPs.

This paper describes how GSOC's planning libraries evolved, presents the current status, and presents the current status. It discusses what generic features have proven beneficial, which features were less helpful, and describes obstacles which need to be considered in different missions. The paper concludes with an outlook on how the GSOC *Mission Planning* team prepares its systems for the future.

1 History

Spacecraft operations at GSOC date back to 1969 when the *AZUR* mission launched [2]. However, the first missions requiring an elaborate planning tool were the missions *Spacelab D-2* [44], the *German Modular Optoelectronical Multispectral / Stereo Scanner* (MOMS-2P [29]), *MIR '97* [28] and the *Shuttle Radar Topography Mission* (*SRTM* [41]), all in the 1990s, when GSOC gathered first experiences with planning tools such as the *Experiment Scheduling Program* of the Marshall Space Flight Center (see [42]). During this time, the need of further tool support was identified, and in-house *Mission Planning* software development started with the *Timeline Output Navigator TimON* ([45]), a tool to display the current timeline. Soon this tool was complemented by a graphical, interactive timeline editor, called *Pinta*, which allowed manual planning. In addition, *Plato*, our first scheduling prototype, which was based on the same kernel as NASA's planning tool *Spike* [40], was developed. It could be called by *Pinta* to run planning algorithms on the current planning model (see [33]). An orbit event calculation library, called *Seppl* has been implemented to generate the required input to the planning process.

In preparation of the *TerraSAR-X* mission, we realized that *Plato*'s variable/value approach, which it shares with other *CSP* based solvers such as *Spike*, would not work for a timeline with resolution of one second and a time horizon of three days. Computation would just not be sufficiently fast for 259200 second slots. *Plato* therefore never passed the state of a prototype. Instead, we implemented a new planning engine, *Plato-II*, whose resource profiles were represented in linear segments instead of by specifying values for each time slot. Together with *Pinta* and *Seppl*, *Plato-II* was used within the *TerraSAR-X Mission Planning* system.

Both *Plato* and *Plato-II* were written in *Lisp*. In preparation of the extensions which were required for the joint *TerraSAR-X/TanDEM-X* mission, we ported *Plato-II* to Microsoft's *C#*, which runs on the *.NET* platform, on which *Pinta* had also been developed. The new *.NET* version of the planning library was called *PlatoN*. Besides, we implemented a replacement of the *C++* library *Seppl*, called *SCOTA* (SpaceCraft Orbit and GroundTrack Analysis Tool [16]), such that the *TerraSAR-X/TanDEM-X Mission Planning* system comprised a tool suite, which was mostly running within the *.NET* ecosystem. Further successful applications of this tool suite became the *FireBird* [49], *Alphasat/TDP-1* [38] and *PIXL-1* [9] mission planning systems, the *EDRS Link Management System* [18], *ColKa* (link planning for the Columbus Ka-band antenna) [4], support for *Galileo* [5], and the GSOC internal *on-call shift planning* system and several others, some of which are described in [32].

Although our *Mission Planning* tool suite was now homogeneous, *.NET* seemed no longer a future-proof technology, due to uncertainties in Microsoft's future strategic direction. As most other state-of-the-art systems within ground operations, including those developed and operated at GSOC, are based on the Java platform, we decided to migrate once

more, from *C#/.NET* to *Scala/JVM*. This not only included a re-design of our existing components *Pinta* and *PlatoN*, whose successors were called *PintaOnWeb* [46] and *Plains*; it also included the introduction of a completely new framework called *Reactive Planning*, intermediately known as *Incremental Planning* [48]. In its latest versions, only *SCOTA* and the *cloud handler* currently remain on the *.NET* platform. Projects which are already using *Reactive Planning*, *PintaOnWeb*, *Plains* and *SCOTA* are *EnMAP* [13] [36] (since 2022), *TDP-1* [47] (since 2021), and *InTAS/ToUCAnS*, a GSOC internal project for Integrated Terminal and Antenna Scheduling, resp. our novel Tool for Unified Control room, Antenna and link Scheduling. It is planned that all upcoming missions supported by GSOC *Mission Planning* will be based on and benefit from this framework.

Overt time, many missions required various different features, and not only during those two platform migrations did the GSOC's *Mission Planning* team reflect which mission-specific features might be generalized to be re-used in future missions. In the following, we present several major features that have been provided generically, together with an assessment of their utility.

2 Example Planning Problem

In order to provide a better understanding, we sketch the planning problem of a typical Earth Observation mission: A satellite in low earth orbit shall be tasked with creating images of targets on ground. Targets are provided by the customers during the mission, e.g. for scientific purposes or disaster monitoring. Acquiring image data requires certain sub-activities, in particular:

- image data acquisition
- power-up of all parts of the instrument before image acquisition
- depending on gap size to succeeding image acquisition: switch off parts of the instrument after image acquisition
- down-link image data over ground station
- file deletion (possibly delayed until reception confirmation)

Obviously, image acquisitions and downlinks need to respect multiple constraints, e.g. target or ground station visibility, exclusive access of attitude control system or payload data memory capacity.

The task of the planning system usually is to generate a feasible timeline, which includes as many high priority image requests as possible, thereafter includes as many medium priority image requests, etc.

3 Generic Features and their Assessment

3.1 GSOC's Planning Modelling Language

Many scientific planning libraries use a pre-defined modeling language such as *PDDL* (see [15]), in order to allow comparing different algorithms for various planning problems. For GSOC, however, it is not crucial to calculate the best plan for a planning model which roughly matches reality, but to calculate a good plan, which matches the reality as good as possible. Therefore, GSOC developed its own descriptive planning modeling language (see [17]).

It allows defining activities, which may be grouped hierarchically, where parent-child relations must form a directed acyclic graph. Each activity may be given one or multiple timeline entries. This way, planning cycles such as the mid-term planning and the short-term planning for Mars Express (see [34]) may be supported: the 4-week mid-term plan provides timeline entries for the parent activities, where the resources are distributed. The short term plan adds timeline entries to the child activities, which may represent spacecraft commands. At GSOC, the hierarchy is mostly used to group activities of the same request or the same ground station, which simplifies navigating through the model. For requests, the root activity is given a timeline entry representing the request's planning horizon.

Relative temporal constraints may be defined. For example, an acquisition may be restricted to start not before the request root's start time and not to end after the request root's end time. This way the request's planning horizon may be enforced.

The most important constraint type is provided by resources and resource dependencies:

- Resource: a time profile predicts how the state of one aspect of the planning problem evolves over time, according to the current timeline.
- Resource Dependency: checks that around a given activity's timeline entry, the resource's time profile remains within a given range.
- Resource Modification: specifies how the time profile has to be modified when adding a timeline entry of a given activity

3.2 References and Offsets

Initially, resource dependencies allowed specifying a start-offset and an end-offset relative to the start-time or the end-time of the corresponding activity's timeline entry. For example a timeline entry of an image acquisition could increase the resource *energy debt* by 2Watt, beginning 5s after start until 10s before end of the timeline entry. Although such a model covers most of our use cases, we were able to generalize this to allow referencing any time t between or even outside the timeline entry's interval by using a *slider*, that is, barycentric coordinates:

$$\begin{aligned} t &= (1 - \text{slider}) \cdot \text{start-time} + \text{slider} \cdot \text{end-time} \\ &= \text{start-time} + \text{slider} \cdot (\text{end-time} - \text{start-time}) \end{aligned} \tag{1}$$

This way, e.g. slider $= 0$ represents the start-time.

The main challenge with this model is to convert a 2-dimensional set of timeline entries into a 2-dimensional set of start- and end-times of constraint profiles, which can be compared to the resource profile in order to find a valid subset of profile start- and end-times, which then needs to be converted back to a set of valid timeline entry intervals (figure 1). This transformation is crucial in order to allow the planning library to calculate a valid set of timeline entries for a given activity, which is one of the core features of our planning library. This is described in detail in [25].

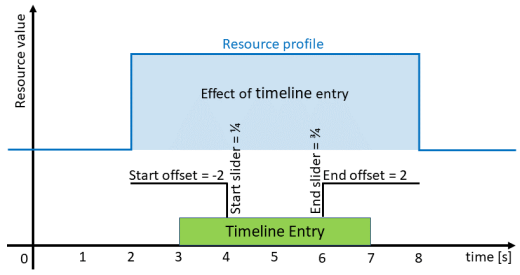

Figure 1: Conversion between timeline entry intervals and resource profile intervals

Although missions did not yet benefit from this model extension, the mathematical concept allowed simplifying code, which previously was rather hard to understand.

3.3 Release Slope - Sliding Windows

Resource modifications allow specifying a profile, which is active during an activity's timeline entry and whose end value may remain active for another configurable duration. When setting this configurable duration to infinity, we call this constraint an *accumulation*, which is used when a resource is used by one activity and supplied by another one, e.g. memory consumption and file deletion.

A release slope specifies that a resource modification's end value is not only extended for another configurable duration, but at the extended end, another segment starts with given release slope, which ends only when value 0 is reached.

This feature allows defining a *sliding window*, see figure 2:

- during the timeline entry, the resource profile is increased by $\frac{1}{sec}$
- the end value is extended until start $+$ window size is reached
- a segment with $\frac{-1}{sec}$ is appended, having the same length as the first segment with slope $\frac{1}{sec}$

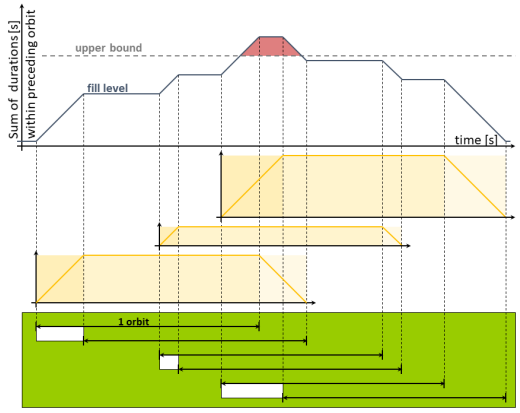

Figure 2: three timeline entries, which contribute to a sliding window

At each time, this resource specifies exactly the durations of all timeline entries within the preceding *window size* seconds. This model is used to handle thermal constraints within the missions *TerraSAR-X/TanDEM-X* and *EnMAP*, since their thermal constraints cannot be propagated sufficiently fast and precise to be considered otherwise within the planning process. It is also in use to implement rules like EnMAP *must not use more than 14 downlinks over Inuvik per week* or, within our on-call planning algorithm, rules like *an operator must not be on-call more than 14 days within 3 weeks*. In general, whenever insufficient information is available for a more detailed model, a sliding window may provide a good alternative.

3.4 Configuring Generic Algorithms

A main goal of GSOC's generic planning library has always been simple re-usability. The initial technique intended to provide this was via configuration files. With Plato-I and Plato-II, one type of algorithm has been implemented, which allowed configuring an initial-strategy, a repair-strategy, a deconflict-strategy and a final fill-in-strategy. For *PlatoN*, we improved this by providing multiple algorithm snippets, which allow calling one another as sub-algorithms, see [26], such that implementing complex algorithms became possible. Whereas the initial approach proved insufficient for complex planning problems, the latter worked fine and has been in use for the missions *TDP-1* and *EDRS*. However, such a configured algorithm turned out harder to implement, more error-prone and harder to debug than an algorithm written in code such as C# or Scala. We therefore concluded not to provide anything like a generic algorithm, which can be configured by the end-user, but instead provide generic sub-algorithms, such as described in 3.5, which can be reused by a programmer, who writes a custom algorithm for a mission's planning problem.

Examples for some aspects that make generic algorithms hard to apply:

1. For *TerraSAR-X/TanDEM-X*, *TDP-1* and *EnMAP*, we need to command sleep-levels between instrument activities (see 3.5). Each acquisition therefore is represented by multiple alternatives, which would significantly increase the complexity of the algorithm, unless the algorithm is aware of this multiplication.
2. For *TerraSAR-X/TanDEM-X* we need to generate a sub-plan for individual image file download. Obviously this sub-plan must match the sub-plan for image acquisition.
3. For *EnMAP* image acquisitions and down-links need to be provided with a guidance list for the attitude control system. Generation of such a guidance list is done outside the planning system and may fail, in which case the planning system has to un-plan the respective activity.
4. For *EnMAP* we need to merge consecutive image acquisitions such that they are served by the same ACS command. For a generic algorithm, which is not aware of such tuples, this increases the runtime complexity further in addition to 1.

5. For *EnMAP* we do not re-generate a timeline from scratch. Instead, we perform an incremental planning run each time new input is available. In order to keep the system responsive, we need to keep runtime of most such incremental planning runs short and therefore for each different type of input, we require a dedicated planning algorithm which is aware of the input type.

3.5 Timeline Entry Chain

Most sub-algorithms which can be re-used between different algorithms, either refer to the framework and are included in 3.7 or are trivial and don't benefit from a generic implementation. One prominent exception however is the Timeline Entry Chain, which is applied in the missions *TerraSAR-X/TanDEM-X*, *TDP-1* and *EnMAP*. This feature may be used when the timeline entries of different activities specify the operations and state transitions of a physical entity. When configured accordingly, the timeline entry chain allows adding an operational timeline entry without having to take care of state transitions before and after the operation.

More precise, we consider a set of *relevant activities* between whose timeline entries a *mode* shall be modelled. Each *relevant activity* can be planned by a specific *alternative*, which is uniquely defined by the *modes* before and after its timeline entry. For example, on *TerraSAR-X*, the *relevant activity* is a data-take. Between two data-takes, certain parts of the instrument need to be switched off in order to save energy and reduce thermal load. The size of the gap to the preceding data-take determines which parts need to be switched off during this gap, which is represented by a timeline entry of the corresponding *mode* activity, see figure 3. Obviously, the initiation sequence of the data-take must be selected according to the preceding *mode* and also the finalization sequence of the data-take must be selected such that the succeeding mode is reached, which means that multiple *alternatives* exist for a data-take. Although the durations of these alternatives may vary, they need to share a common *relevant interval*, in this case the net observation time. The rule, which modes to select between two data-takes, depends on the gap size between these *relevant intervals*. In order to allow variability in case further constraints may interfere with the mode selection, multiple modes may be valid for a given gap-size, in which case a rule of preference is used to decide which shall be used, if multiple modes are feasible. Having defined the *relevant activities*' *alternatives*, the *modes* and the rules when to select which mode, the *Timeline Entry Chain* allows finding valid timeline entries of *relevant activities* and placing them on the timeline without having to take care about the selection of *alternatives* and *modes* in between. These are selected and planned by the *Timeline Entry Chain*.

The *Timeline Entry Chain* proved a valuable tool to simplify the logic of a mission's algorithm, reducing cost and increasing robustness of the mission planning system.

3.6 Templates

A reoccurring task in an earth observation's planning system is the ingestion of new requests. One can solve this obviously by implementing an algorithm, which generates

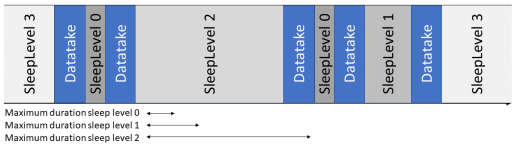

Figure 3: *TerraSAR-X* sleep levels between data-takes

the new model objects from the submitted parameters. For a fully automated system, there is little to complain about this approach, since changes to such a planning system will occur infrequently and all changes need to pass integration tests before activation, even if they would be implemented via re-configuration.

For manual planning and semi-automated planning, where an operator uses a front-end such as *PintaOnWeb* to generate multiple copies of the same structure, a more comfortable approach should be available. One way would be to implement a custom algorithm which generates the desired structure. This approach has been used in multiple missions like *TerraSAR-X/TanDEM-X*, *FireBird*, *TDP-1*, *ColKA*, *EDRS*, *CubeL* and *on-call shift planning*.

As proven by the *TerraSAR-X/TanDEM-X* missions, some constraints need adaptation during the mission's lifetime, see [43] and [3]. Within the fully automated *TerraSAR-X/TanDEM-X* mission, we therefore implemented a generic way to configure the generation of constraints, the only part of 3.4, which turned out beneficial.

However, a second approach had already been present within *Pinta*: *templates* allow an operator to quickly generate copies of a non-project-able activity. The great benefit of *templates* is that one can use *Pinta* or *PintaOnWeb* to generate and adapt templates as if they were normal activities. Besides, as they are part of the planning model, no further configuration is required. We therefore decided to elaborate templates rather than continuing work on a structure generation algorithm.

Within the *EnMAP* planning system, we already use templates in many places to generate new activities and resources, however we still use custom code in many places, because the template mechanism was not complete during their development.

Our current *Template* concept, which we believe to be the perfect trade-off between simplicity and re-usability, consists of the following features:

- *Template activity* A template activity is an activity, which cannot have timeline entries and whose child activities must be templates, too. On instantiation, a copy of the template is created, together with a copy of all of its descendants. Unless 3.6 is used, all copies are normal activities, which preserve a reference to their template.
- *Template resource* A resource may be defined for a *template activity*. This resource has no profile, but constraints may be defined between the *template activity* and its *template resource*. On instantiation of the *template activity*, a copy of the *template resource* is created and connected to the new *instance activity*.

- *Template constraints* Constraints may be defined on the *template activity*. On instantiation of the *template activity*, the new *instance activity* receives a copy of the constraints of the *template activity*. In case the *template constraint* refers to a *template resource* of the *template activity*, the constraint is defined between the new *instance activity* and its new *instance resource*.
- *Template variables* Most properties of an activity, including its constraints' values, may refer to *variables* instead of concrete values. A variable may define a rule how to read a value from the model during runtime (e.g. read a project parameter) or it may specify that on instantiation, a value must be provided, which shall be used for instantiation.
- *Delayed instantiation* Sometimes, a template needs to be instantiated only partially. For example, when generating the activity representing an *EnMAP* image request, the *request* template needs to be instantiated. However, its child template *opportunity* must be copied and remain a template, because on instantiation of the request, we don't know yet which target opportunities will exist for this request – they might all lie past the current planning horizon. We therefore need to be able to specify at which child template activities, instantiation needs to be broken off and a template must be copied instead.
- *Multi parents templates* Consider a mission with variable number of satellites and variable number of ground stations, for which contacts between satellites and ground stations need to be planned. In this case, we may define a template for a ground station and a template for a satellite. Additionally, we define a template representing the contact between a satellite and a ground station. We add this connection template as child to both, the satellite template and the ground station template. We may also define a ground station resource and a satellite resource as template resource of the respective template activity and define resource dependencies between the connection template and these resources. On instantiation of a ground station, the connection template is not instantiated, because no instance of the connection template's parent *template satellite* exists yet. On instantiation of the first satellite however, for each previously instantiated ground station, one *connection* instance is generated, including copies of all constraints between the *connection template* and the instances of its parent templates' resources.

With this set of capabilities, we could have implemented the whole instantiation process of the *EnMAP Mission Planning* system generically using *templates* only. Nevertheless, even if we had had the missing feature 3.6 in place in time for implementation of the *EnMAP Mission Planning* system, we probably would still not have used the variable mechanism to ingest the results of various *EnMAP* formulas into resource constraints' profiles, because that includes significant complexity. This would only make sense if a human operator should use these templates and possibly adapting them using *PintaOnWeb*. For such a case however, we still need to implement better support within *PintaOnWeb* in order to simplify the usage of template variables.

3.7 Reactive Planning

Several years after launching the *TerraSAR-X* satellite, GSOC and its commercial partner Airbus discussed what might be improved in the potential successor mission *TerraSAR-X II*. The main outcome was neither that the number of images should be increased, nor that some other optimization criterion should be maximized. Instead, the most important criterion was to allow ingesting a request and immediately receiving status updates of this and other affected request: Has the request been planned? What other requests had to be un-planned or displaced? If this information was available, customers may react in time before uplink. Additionally, the stability of the planning result and the possibility to explain why certain decisions were taken, would be very much appreciated.

In response to this finding, GSOC's *Mission Planning* team implemented its new *Reactive Planning* framework, which allows implementing a message-driven *Mission Planning* system, which may maintain an up-to-date timeline and which provides a standardized way to interact with the commanding system.

The first mission to implement a *Reactive Planning* system has been *EnMAP*, which indeed provides a permanent up-to-date timeline, see [12] and [22]. Together with the *Reactive Planning* framework, GSOC developed *PintaOnWeb*, a graphical front-end, which allows inspecting and modifying the planning model and the timeline, see [46]. Since both are based on the same planning model, *PintaOnWeb* can be fully integrated into any *Mission Planning* system, which is based on the *Reactive Planning* framework: *PintaOnWeb* will receive a patch each time an algorithm completes and *PintaOnWeb* itself can send patches to the *Reactive Planning* system in order to apply modifications, which an operator has created via *PintaOnWeb*. For *EnMAP* however, we restrict to displaying the model and editing project parameters, which serve as configuration settings of the algorithm, all other interactions are implemented via dedicated messages.

Shortly after activating the *EnMAP Mission Planning* system, we also upgraded the *TDP-1* planning system to run on the *Reactive Planning* framework. For *TDP-1* however, we do not provide an up-to-date timeline but instead only collect input and evaluate it twice a day, both times running a from-scratch algorithm, similar to the concept of *TerraSAR-X/TanDEM-X*, see [27], [14], [30], [21] and [31].

As described in [22], reactivity has its price in coding effort, because each input type must be considered as a separate use case. With a from-scratch planning approach like in *TDP-1* and *TerraSAR-X/TanDEM-X*, one can simplify the algorithm by sorting the input in a suitable order. Whether a mission's planning system should maintain an up-to-date timeline therefore must be decided depending on the missions requirements and the implementation effort one is willing to accept.

The reactive planning framework itself however does not prescribe which type of algorithm to implement. It is even possible to implement an optimization algorithm, which may get triggered for each new input or – if runtime performance requires so – which gets triggered each night at 10

pm. However, such a repeated optimization seems to contradict the user's desire for reliable information about the planning states of his requests. Instead, *Reactive Planning* implements clean interfaces and generic workflows, such as when and how to create a command set and how to consider feedback about up-linked commands. Additionally, *Reactive Planning* provides the programmer with all generic features that were implemented for previous missions. The *Reactive Planning* framework will remain the basis for all upcoming planning systems at GSOC.

4 Comparison to other planning frameworks

Obviously, GSOC is just one of many players who need to provide planning systems for complex space missions, which lead to a variety of planning frameworks in the space community, see [7]. At least within their scientific descriptions, most planning frameworks focus on the planning model and the algorithms. Whereas this is indeed the interesting part, the *Reactive Planning* framework explicitly provides a standardized way of how everything around the planning model and the algorithms may be set up to achieve a running planning service, which allows re-using large parts of the software, including solutions to maintain reactiveness, overcome network boundaries, avoid message buffer overflows, monitoring system health and provide insight into the current planning status. The *Reactive Planning* framework explicitly does not include a generic solver, see 3.4. Instead, mission specific code must be written to implement the planning algorithms, for which the planning library *Plains* provides useful functions.

Apart from this, the main differences to GSOC's planning library *Plains* are listed here:

- APSI (see [11]) supports resources with piece-wise constant values. This is slightly less than the piece-wise linear profiles of *Plains*, which makes it harder to model e.g. the continuous release of memory during a downlink with non-fix duration. Also the model of a sliding window (see 3.3) requires a slope in order to be precise. However APSI supports further concepts, which are not present in *Plains*, in particular state variables.
- ASPEN (see [6]): like APSI, ASPEN seems also only to support piece-wise constant values. Also, *Plains* does not distinguish between renewable and non-depletable resources. Instead, the resource dependency determines the way it modifies the resource, which allows combining these types of constraints. For example, taking an image may reserve memory, down-linking an image may release memory (both renewable resources), but onboard analysis of the image to detect certain events may temporarily allocate memory
- flexplan provides a comfortable way to configure the system without the need to generate code. While in simpler cases, this approach may save much effort. For more complex cases, flexplan allows implementing custom code, in which way it resembles the approach of *Reactive Planning*. An interesting task would be to compare the capabilities of the underlying planning libraries, however the authors didn't have access to the flexplan API.
- SPIKE (see [19]): as mentioned in 1, the kernel of spike had been used for prototyping *Plato* before implementing the first operational version of *Plato-II*. *Plains*' resource model therefore resembles the one of SPIKE. However, the integer-based time, which may provide high performance for the integrated CSP based solver in short time ranges, turned out to be a performance bottleneck for longer planning horizons with sub-second resolution. Also, the resource model has been improved by supporting piece-wise affine linear resource profiles and not distinguishing between types of resources, but instead distinguishing between types of resource constraints.
- MUSE (see [20]) tackles a different problem: its goal is to provide a way to allow users to select from different possible solutions. Although a future version of *Reactive Planning* will allow a user, who submits a request, to pre-view and select from different possible outcomes, no overall optimization with respect to multiple goals is foreseen in the design of *Reactive Planning*. This step would have to be implemented as mission specific code.
- SPIFe (see [1]) is designed to support a human planner. This way it resembles *Reactive Planning*'s *PintaOnWeb*, which can be used not only to display the current timeline but also to edit the timeline. It is also possible to trigger functions in *PintaOnWeb*, which may provide similar features as SPIFe's Plan Advisor. Again, SPIFe only supports piece-wise constant resources.
- CLASP (see [8], [10]) includes an event calculation functionality, whose initial goal was to cover given areas. *Reactive Planning*'s *SCOTA* component on the other hand was initially designed to calculate opportunities for given targets. Within the scope of the *EnMAP* mission, *SCOTA* has been extended to determine swathes for coverages. A proposal for a suitable algorithm to use this feature to generate a coverage has been given in [24]. For sure, CLASP will solve this use case better, however, *Reactive Planning*'s main purpose is not the generation of a timeline for ground coverage but to incrementally adapt the upcoming part of the timeline according to continuously ingested new point requests and to generate all commands to execute this plan onboard the satellite. This seems not to be a valid use case for CLASP.
- Planet's dove planning system (see [39]) creates timelines for 100 satellites and 30 ground stations. This system is intended on the one hand to provide service and downlink contacts for all satellites and on the other hand to provide image coverage from the whole earth. To manage such a large planning problem, a dedicated solver has been implemented, which first plans ground contacts and thereafter plans the image acquisitions. This way, the planning system of Planet is a highly specialized solution to a computationally extremely challenging problem.

5 Summary and Outlook

With every new mission, new requirements arise for a planning system and every mission provides valuable insight into how a planning system should or should not look like. At

GSOC, we try to merge all these lessons learnt into our code base, benefitting future missions as most developers are directly involved in operations. One major outcome is that one should not try to provide a generic configurable algorithm which intends to cover all possible missions. Instead, one should identify common sub-algorithms, which are extracted from existing planning systems' needs and make them available for future missions, as shown in 3.5, 3.6 and 3.7.

Also, we started to implement an adapter for the upcoming CCSDS standard for Mission Planning Services (see [35]), which allows using the CCSDS MPS standard to interact with the *Reactive Planning* framework, e.g. for sending requests, monitoring state transitions of such requests and communicating the plan.

Obviously, preserving lessons learnt is an important task to improve efficiency, but equally important is to anticipate what future systems might require. The *Reactive Planning* system was born from such a vision, as well as *PintaOnWeb*. Both are now in operational use with *EnMAP* and *TDP-1*, and provide significant benefit and on the long run will also reduce cost in development and maintenance. We therefore continue with our scientific exploration of operational concepts.

For example, the booking of ground station antennas currently follows a semi-automated workflow, in which ground station contacts are requested once a week, up to 2.5 weeks in advance. Adaptations to these ground contacts can only be considered in emergency cases. The new development *InTAS/ToUCAnS* will support planning all missions' ground contacts in a reactive manner, which allows individual missions to request new passes on short notice and cancel them in order to provide other missions fresh up- and downlink capacities. This way the available ground antennas may be utilized in the overall best possible way. We consider this an important prerequisite for integrating further satellites into GSOC's operational environment.

A topic which until now has been neglected by GSOC's *Mission Planning* team is optimization, because none of GSOC's missions had an strong requirement for it. In consequence, whereas planning domains like PDDL have been designed with algorithmic optimization in mind, GSOC'splanning modelling language focuses on simple and direct modelling of real world's problems. There also don't exist generic optimization algorithms for the GSOC planning domain, instead features such as detecting conflicts and finding non-conflicting timeline entries for given activities allow implementing heuristic algorithms.

In order to fill the gap to optimization, the GSOC *Mission Planning* team intends introduce a hybrid approach: The full planning problem is defined in GSOC'splanning domain and a simple heuristic and possibly randomized algorithm is used to generate one or multiple plans, in order to determine which parts of the planning problem are critical ones and which usually can be omitted. From these critical parts of the planning model, a reduced planning model shall be derived within a planning domain for which there exist optimization algorithms. The results of these optimization algorithms will then be used as a starting point for a heuristic algorithm, which runs on the full GSOC planning model.

To be prepared for the possibilities which the availability of quantum computers may provide in the future, GSOC's *Mission Planning* team has started work on quantum algorithms: A quantum algorithm to solve the *on-call shift planning* problem is currently being developed, see [37]. Even though currently available hardware does not yet allow for solving the complete current real-world problem, we expect this project to help us understand which problems may be solved using quantum computers in the future. It allows us to find suitable mappings of problems encoded in our descriptive modelling language to problems which can be handled by a quantum algorithm.

Other generic developments include coverage splitting for coverage orders, one approach of which has been presented at IWPSS 2021 (see [24]). Further work refers to concurrency and how to merge requests and planning model edits into a common database, see *PintaOnWeb*([46]) and *Reactive Planning*'s connection scheduler ([48]). Incorporation of weather forecast has already been introduced in the *EnMAP Mission Planning* system, however it still needs refinement. Also, a special approach of integrated on-ground / onboard planning, which has first been described in [23], waits to be implemented.

References

[1] Aghevli Arash, Bencomo Alfredo, and Mccurdy Michael. *Scheduling and Planning Interface for Exploration (SPIFe)*. 2011. URL: http://icaps11.icaps-conference.org/demos/system_demo_proceedings/mccurdy-et-al.pdf (visited on 11/12/2018).

[2] *AZUR*. URL: https://www.dlr.de/rb/en/desktopdefault.aspx/tabid-12671/4262_read-6343/ (visited on 04/05/2023).

[3] Arvind Kumar Balan and Christoph Lenzen. "Li-Ion Battery Operations and Life Optimization". In: *68th International Astronautical Congress*. URL: https://elib.dlr.de/116831/.

[4] Florian Bender et al. "First Experience With Columbus DMS Modernization, COL Ka Operations And IP-Based Communication". In: *72nd International Astronautical Congress (IAC 2021)*. 2021. URL: https://elib.dlr.de/146816/.

[5] Sandra Brogl et al. "How Galileo Planning became automated". In: *16th International Conference on Space Operations*. May 2021. URL: https://elib.dlr.de/142237/.

[6] S. Chien et al. "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling". In: *International Conference on Space Operations (SpaceOps 2000)*. Toulouse, France, June 2000.

[7] Steve A. Chien et al. "A Generalized Timeline Representation, Services, and Interface for Automating Space Mission Operations". In: *Proceedings of 12th International Conference on Space Operations*. Stockholm, Sweden, 2012. URL: https://arc.

aiaa.org/doi/10.2514/6.2012-1275459 (visited on 06/15/2023).

[8] *CLASP*. URL: https://ai.jpl.nasa.gov/public/projects/clasp/ (visited on 07/01/2021).

[9] *CubeLCT*. URL: https://www.dlr.de/content/en/articles/news/2021/01/20210124_pioneering-launch-compact-satellite-with-smallest-laser-terminal.html.

[10] Joshua R. Doubleday. "Three Petabytes or Bust: Planning Science Observations for NISAR". In: *SPIE 9881*. New Delhi, India, June 2023. DOI: 10.1117/12.2223893. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/9881/988105/Three-petabytes-or-bust-planning-science-observations-for-NISAR/10.1117/12.2223893.full.

[11] Simone Fratini and Amedeo Cesta. "The APSI Framework: A Platform for Timeline Synthesis". In: *1st Workshops on Planning and Scheduling with Timelines PSTL-12*. 2012.

[12] Thomas Fruth et al. "The EnMAP Mission Planning System". In: *15th International Conference on Space Operations*. Marseille, France: American Institute of Aeronautics and Astronautics, May 2018. DOI: 10.2514/6.2018-2525. URL: https://elib.dlr.de/120448/.

[13] Thomas Fruth et al. "The EnMAP Mission Planning System". In: *Space Operations: Inspiring Humankind's Future*. Ed. by H. Pasquier et al. Tiergartenstrasse 17, D-69121 Heidelberg: Springer-Verlag GmbH, Heidelberg, 2019, pp. 455–473.

[14] Michael P. Geyer, Falk Mrowka, and Christoph Lenzen. "TerraSAR-X/TanDEM-X Mission Planning - Handling Satellites in Close Formation". In: *SpaceOps 2010 Proceedings AIAA-2010-1989*. Apr. 2010. URL: https://elib.dlr.de/67873/.

[15] Malik Ghallab et al. *PDDL - The Planning Domain Definition Language. Version 1.2*. Tech Report CVC CVC TR-98-003/DCS TR-1165. Version 1.2. AIPS-98 Planning Competition Committee. Yale Center for Computational Vision and Control. URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi334Kw84_-AhWrM-wKHbBSBPcQFnoECA8QAQ&url=https%3A%2F%2Fwww.cs.cmu.edu%2F~mmv%2Fplanning%2Freadings%2F98aips-PDDL.pdf&usg=AOvVaw0Zu_4juxBDE-OEiMzgwgA7.

[16] Elke Gross et al. "SCOTA – The Mission Planning Orbit Analysis Tool at GSOC". In: *72th International Astronautical Congress 2021*. Accepted. Dubai, United Arab Emirates, Oct. 2021.

[17] *GSOC Planning Modeling Language*. 2010. URL: https://www.dlr.de/rb/Portaldata/38/Resources/dokumente/GSOC_dokumente/RB-MIB/GSOC_Modelling_Language.pdf (visited on 07/08/2021).

[18] Tobias Göttfert et al. "Operating and Evolving the EDRS Payload and Link Management System". In: *2018 SpaceOps Conference*. SpaceOps Conferences. American Institute of Aeronautics and Astronautics, May 2018. DOI: 10.2514/6.2018-2688. URL: https://doi.org/10.2514/6.2018-2688.

[19] M. D. Johnston and G. E. Miller. *Intelligent scheduling*. Ed. by M. Aarup, Monte Zweben, and Mark Fox. 1st ed. Morgan Kaufmann, San Francisco, Calif, 1994. ISBN: 9781558602601.

[20] Mark Johnston and Mark Giuliano. "MUSE: THE MULTI-USER SCHEDULING ENVIRONMENT FOR MULTI-OBJECTIVE SCHEDULING OF SPACE SCIENCE MISSIONS". In: *IJCAI - 09 Workshop on Artificial Intelligence in Space*. July 2009. URL: https://www.esa.int/gsp/ACT/projects/workshop_ai09/papers/s3_2johns.pdf (visited on 06/15/2023).

[21] Christoph Lenzen et al. "Automated Scheduling for TerraSAR-X/TanDEM-X". In: *IWPSS 2011*. June 2011. URL: https://elib.dlr.de/74101/.

[22] Christoph Lenzen et al. "EnMAP MPS: Challenges, Enhancements and Evaluations of the Early Mission Phase". In: *17th International Conference on Space Operations*. Dubai, United Arab Emirates, 2023.

[23] Christoph Lenzen et al. "Onboard Planning and Scheduling Autonomy within the Scope of the FireBird Mission". In: *SpaceOps 2014 - 13th International Conference on Space Operations*. 2014. URL: https://elib.dlr.de/89409/.

[24] Christoph Lenzen et al. "Planning Area Coverage with Low Priority". In: *12th International Workshop on Planning & Scheduling for Space (IWPSS)*. July 2021, pp. 80–89. URL: https://elib.dlr.de/143762/.

[25] Christoph Lenzen et al. "Polygon stacks and time reference conversions". In: *CEAS Space Journal* (Jan. 2020). URL: https://link.springer.com/article/10.1007/s12567-020-00296-7.

[26] Christoph Lenzen et al. "The Algorithm Assembly Set of Plato". In: *SpaceOps 2012 Conference*. SpaceOps Conferences. American Institute of Aeronautics and Astronautics, June 2012. DOI: 10.2514/6.2012-1255731. URL: https://doi.org/10.2514/6.2012-1255731.

[27] E. Maurer et al. "TerraSAR-X Mission Planning System: Automated Command Generation for Spacecraft Operations". In: *IEEE Transactions on Geoscience and Remote Sensing* 48.2 (2010), pp. 642–648. URL: http://elib.dlr.de/63512/.

[28] *MIR '97*. URL: https://www.dlr.de/rb/desktopdefault.aspx/tabid-12669/4263_read-6347/.

[29] *MOMS-2P*. URL: https://www.dlr.de/rb/desktopdefault.aspx/tabid-12671/4262_read-6334/.

[30] F. Mrowka et al. "The Joint TerraSAR-X / TanDEM-X Mission Planning System". In: *Symposium Proceedings of IGARSS 2011*. Vancouver, Canada, 2011, pp. 3971–3974. URL: http://elib.dlr.de/74917.

[31] Falk Mrowka et al. "The TerraSAR-X/TanDEM-X Mission Planning System: Realizing new Customer Visions by Applying new Upgrade Strategies". In: *14th International Conference on Space Operations*. Deajeon, Republic of Korea, 2016. URL: http://elib.dlr.de/104596.

[32] Rainer Nibler et al. "PINTA – one Tool to plan them all". In: *16th International Conference on Space Operations*. May 2021. URL: https://elib.dlr.de/142842/.

[33] Rainer Nibler et al. "PINTA and TimOnWeb - (more than) generic user interfaces for various planning problems". In: *IWPSS 2017 - 10th International Workshop on Planning and Scheduling for Space*. 2017. URL: https://elib.dlr.de/114095/ (visited on 11/12/2018).

[34] Rene Pischel and Tanja Zegers. *Science planning and operations for Mars Express*. URL: https://sci.esa.int/c/portal/doc.cfm?fobjectid=41765 (visited on 06/15/2023).

[35] Peter van der Plas et al. "CCSDS Mission Planning And Scheduling Services Opening Door For Cross-Agency Interoperability". In: *16th International Conference on Space Operations (SpaceOps 2021)*. 2021. URL: https://elib.dlr.de/142188/.

[36] S. Prüfer et al. "Use Cases and Algorithms of the EnMAP Mission Planning System". In: *16th International Conference on Space Operations*. May 2021.

[37] Sven Prüfer et al. "Evolving Spacecraft Quantum On-Call Scheduling". In: *17th International Conference on Space Operations*. Dubai, United Arab Emirates.

[38] Gregor Rossmanith et al. "Laser Communication in Space: The TDP-1 Mission Control Center and its current operational experience". In: *14th International Conference on Space Operations*. Daejeon, Republic of Korea, May 2016. DOI: 10.2514/6.2016-2522.

[39] Vishwa Shah et al. "Scheduling the World's Largest Earth-Observing Fleet of Medium-Resolution Imaging Satellites". In: *International Workshop for Planning and Scheduling for Space (IWPSS 2019)*. July 2019. URL: https://drive.google.com/file/d/1awd_Ip64ukFeHHRkspph4pOvLxxKv2ZD/view (visited on 06/15/2023).

[40] *Spike*. URL: https://www.stsci.edu/scientific-community/software/spike.

[41] *SRTM*. URL: https://www.dlr.de/eoc/en/desktopdefault.aspx/tabid-5515/9214_read-17716/.

[42] Kenneth L. Stacy and John P. Jaap. "Space station payload operations scheduling with ESP2". In: NASA. Lyndon B. Johnson Space Center, 1988. URL: https://ntrs.nasa.gov/citations/19890010448.

[43] Fotios Stathopoulos et al. "Operational Optimization of the Lithium-ion Batteries of TerraSAR-X/TanDEM-X". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (14 2021), pp. 3243–3250. DOI: 10.1109/JSTARS.2021.3056174. URL: https://elib.dlr.de/139195/ (visited on 04/04/2023).

[44] *STS-55 D-2*. URL: https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Spacelab_D-2.

[45] M. Wickler and G. Zoeschinger. "TIMON - The Timeline Output Navigator". In: *SpaceOps 1998 Conference, Tokio, Japan, 1-5 June 1998*. 1998. URL: https://elib.dlr.de/11428/.

[46] A. Wiebigke et al. "PintaOnWeb - The Front End of GSOC's Next Generation Mission Planning Systems". In: *17th International Conference on Space Operations*. Dubai, United Arab Emirates.

[47] Maria Th. Wörle et al. "Replacing the TDP-1 Mission Planning System – more than just another Technical Demonstration Project". In: *72th International Astronautical Congress 2021*. Dubai, United Arab Emirates, 2021. URL: https://elib.dlr.de/145871/.

[48] Maria Theresia Wörle et al. "The Incremental Planning System – GSOC's Next Generation Mission Planning Framework". In: *Space Operations: Innovations, Inventions and Discoveries*. Ed. by C. Cruzen, M. Schmidhuber, and L. Dubon. Tiergartenstrasse 17, D-69121 Heidelberg: Springer-Verlag GmbH, Heidelberg, 2015, pp. 285–307.

[49] Maria Theresia Wörle et al. "The Mission Planning System for the Firebird Spacecraft Constellation". In: *14th International Conference on Space Operations*. Daejeon, Republic of Korea, 2016. URL: https://elib.dlr.de/105950/.