POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Generalized On-Ground Mission Planning System in the New Space paradigm: an overview

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Marco Morgese**

Student ID: 953830
Advisor: Prof. Francesco Topputo
Academic Year: 2022-23

Abstract

The aim of this thesis is to analyse the Mission Planning System in the context of the New Space Economy, in order to identifies which needs, architectures and technologies have shifted with respect to classical space missions.
In particular, the CCSDS recommendations for Mission Planning have been used as starting point to provide a complete system overview, followed by the decomposition in Functional Architecture Modules realized by Chazy et al. in the 2009.
The use cases analysis and the report of the common models and algorithms used in satellite planning are then discussed to provide the reader of a complete view on the topic.
Particular attention is dedicated to the generalization of a Mission Planning application, with a detailed description of timeline-based planning and the evolution of description languages for planning.
As conclusion, an overview of the technological opportunities in the IT field for the development of a Mission Planning System is presented, focusing on cloud-computing, OS-level virtualization, microservice architectures and REST APIs.

Keywords: Mission Planning, New Space, timeline-based planning

Sommario

Lo scopo di questa tesi è quello di analizzare il sistema di pianificazione di missione nel contesto della New Space Economy, al fine di identificare quali esigenze, architetture e tecnologie sono emerse rispetto alle classiche missioni spaziali. In particolare, le raccomandazioni prodotte dal CCSDS per la pianificazione di missione sono state utilizzate come punto di partenza per fornire una panoramica completa del sistema, seguita dalla decomposizione in moduli funzionali d'architettura realizzata da Chazy e altri nel 2009. L'analisi dei casi d'uso e il resoconto dei modelli e degli algoritmi comunemente utilizzati nella pianificazione di missione sono poi stati discussi per fornire al lettore una visione completa dell'argomento. Particolare attenzione è dedicata alla generalizzazione di applicazioni dedicate alla pianificazione di missione, con una descrizione dettagliata della pianificazione basata su linee temporali e dell'evoluzione dei linguaggi di descrizione per la pianificazione. In conclusione, viene presentata una panoramica delle opportunità tecnologiche in campo informatico per lo sviluppo di un sistema di pianificazione di missione, con particolare attenzione al cloud-computing, alla virtualizzazione a livello di sistema operativo, alle architetture basate su microservizi e alle REST API.

Parole chiave: Pianificazione di missione, New Space, pianificazione basata su linee temporali

Contents

1 | Introduction

In the last two decades, the term New Space has become trending in the space industry. In order to avoid falling into the pitfall of a *buzz-word*, a specific definition and comprehension of the meaning of New Space is needed. In [31], the author reports a survey over a sample of 20 experts in the field and analyzes the results in search of commonalities. The most common answers were, in order of count:

1. "Private customers as primary market, as opposed to government customers as primary market";
2. "Unconventional system development approaches";
3. "New business models / Private funding sources".

New Space is often confused with the commercialization of space, but this is clearly not true. As Calderoni reports in its article: "When we say NewSpace, we are not talking merely of the general commercialization of space, as there has been a commercial element in space activities for decades, but rather the cultural and philosophical shift towards greater private entity participation." [8]. It is more correct to see the New Space Economy as the ensemble of shifting paradigms within the Space Economy.

The participation of private entities emerges from the shift of funding methodologies. NASA Public-Private Partnership (PPP) model and the entrance of Private Equity (PE) investors and Venture Capital (VC) are clear examples.
Shifting the funding model, also the business model needs to change. In traditional space, cost-plus agreements were typical. In these agreements the institutions refund private entities of their project costs while guaranteeing a profit margin. In New Space contracts, fixed price agreements are more common. Project cost is fixed and payments are made when a project milestone is reached.

Another shift is also relevant in the business model. While in traditional space the purpose of a project was the delivery of a product, in the New Space context the product is a service (*as-a-service* model), that typically delivers data and insight to end-users. In fact,

in a fixed price agreement, companies have a bigger incentive to maximize operational efficiency in order to maximize profits.

The desire for efficiency improvements, combined with a market-pull approach, often results in engineering approaches that are different from the practice in traditional space (e.g. acceptance of in-orbit failure, Agile product development methodologies,...). In emerging engineering approaches, the miniaturization of space platforms and the distribution of mission functionalities over a significant number of spacecraft have significant repercussions on the mission needs and design. In this context, the CubeSat standards defined by CalPoly [46] prosper, not only in the educational field, but as standard at industry level.

In [4], the author identifies the consequences of the higher complexity due to this shift, which mainly involve:

- basic communication tasks and ground resources allocation;
- coordination and higher probability of anomalies;
- mission objectives;
- Space Situational Awareness (SSA).

In the same article, the author identifies the element of stress when scaling a satellite constellation:

- resource allocation: planning and scheduling;
- satellite design, manufacturing, and launch;
- commissioning;
- nominal operations, monitoring, and anomaly handling;
- software and configuration;
- orbit management and collision avoidance;
- calibration/validation.

The rapid spread of New Space missions requires infrastructures to operate them. The trend translates in drivers for the design of complex ground infrastructure.

First, it becomes too costly for each new mission provider to develop a whole ground infrastructure dedicated to their mission. As stated before, usually a mission starts with few satellites with the intent of scaling later, and the implementation of a complete ground

segment plus the software to operate it, both needed to be capable of scaling, would cost as much (if not more) as the space segment. The need to switch from Mission Dedicated Architecture to Service Oriented multi-mission Architecture is then clear and proved by the industry evolution in the last 15 years. Consultative Committee for Space Data Systems (CCSDS) recommends operating the space segment on a service basis, and Ground Station as a Service concept is nowadays widely adopted. In this context, it is possible to identify two main drivers for the ground segment design and implementation:

- **Scalability**: the user can pass, in few years, from few units to thousands of satellites;
- **Multi Mission**: tailored infrastructure are costly to develop and maintain. As much as possible of the infrastructure should be shared and, indeed, provided as a service.

Scalability driver

Automation of the ground segment becomes a key enabler for scalability. Thousands of satellites, with inter-dependencies in their tasks, can not be operated manually. Automating the planning of a constellation involves a number of technological challenges:

- **Planning and Scheduling for satellites is known to be a NP-hard problem**. While exact algorithms able to solve the problem for a single satellite in acceptable runtime exist, this is no more true when the planning involves constellations with hundreds/thousands of satellites with interdependency between their tasks. The solution has then to be provided by approximated algorithms, capable of returning a non-optimal, but feasible solution. In this context meta-heuristic or domain specific heuristic approach should be considered when implementing a solution. A strategy to reduce complexity is the decomposition of the planning problem in subproblems. A classic example is decomposing the constellation planning in a task allocation problem (among the different individuals) and in planning for operations problem (one instance for each satellite). A trade-off between desired runtime (and so responsiveness) and desired optimality of the plan should be performed in order to identify the best decomposition strategy for the mission.
- **A planning and scheduling solution usually reasons on some levels of abstraction**.

  To ensure the validity of the produced plan a validation process is needed. High fidelity simulators have to be provided in order to enable full automation of the process. Results of the simulations will be checked against Flight Rules. This is also a driver for the platform provider: Flight Rules has to be provided in a script-

like form, instead of the classical human readable form, to allow automatic checks.

- **Pre-planning for booking contacts**.
  In the context of Ground Station as a Service (GSaaS), the booking for antenna could be required to be performed before all the requests have been submitted. A method to evaluate what are the most convenient slot to be booked should be provided. As the booking has been performed and the requests received by the mission operators, the planning and scheduling problem will present a lower complexity w.r.t. planning for contacts and tasks at the same time.

- **Providing a "light-off" automation level means implementing execution strategies able to handle uncertainty**.
  Cloud coverage considerations, last minute requests, anomaly reaction and exogenous events are few examples. In [6], the authors identify three strategies to front uncertainty in Planning and Scheduling: proactive, revision and progressive techniques.

  proactive: building a global solution at generation time, which will be never be reconsidered at execution time. It is proactive in the sense that uncertainty is considered in order to generate a reliable schedule. Possibilities are:

  – generate a generic schedule which is proved to cover most cases (typically conservative and thus sub-optimal);

  – generate a flexible solution: decisions have not been made and are postponed until execution time;

  – generate a conditional solution: various mutually exclusive decision are developed and one choose between them at execution time.

  revision: complete schedule is generated in classical predictive way and adjusted at execution time if it does not match the observed situation.
  progressive or continuous: the plan is generated and executed in the classical prediction way, but locally (short term). A new part of global schedule is generated online (either at predefined timestamp or when conditions trigger it).
  As can be noted from what stated above, the uncertainty handling is strictly coupled with the level of plan execution monitoring.

Multi-Mission Operations driver

The multi mission operations driver implies other needs in the development of mission planning solutions.

- **Reusability of planning solutions**
  Up to now, most of the mission use mission planning strategies tailored for the specific mission. The way ground contact booking can be performed, the payload acquisition strategy, the operated platform and the Mission Operations Center architecture are some of the reasons those lead to this diversity. The simplification and standardization of operated platform provided by the MiniSats scenario and the shift to multi-mission Service oriented Mission Operations could allow a common representation of the planning problems and so a common library of solvers that will require only tuning as mission specific activities. The adoption of an expressive enough domain description language among the Mission Operations provider could reduce the efforts for integration of new researched techniques within commercial applications. In particular, it seems to be prevalent the concept of timelines as main planning representation. Both ESA and NASA promote this approach in the context of NASA AMMOS 2.0 ([7], [13], [19]), and the ESA APSI initiative ([25], [37]. It has to be noted that this kind of representation is highly preferable for explicit resource reasoning, but can be integrated with first order predicate reasoning (as ANML, description language for EUROPA planner [44]) as well as with path planning (timeline framework, path planning and PDDL integration, [26]). However, it has to be considered that a bigger expressiveness requires a bigger implementation efforts for the solver. In this sense is interesting the approach of iterative conflict resolution, that can provide a context for modular solvers tailored for the deconfliction of a single component of the representation formalism (for example resource deconfliction or state variable deconfliction) [40].

- **Request handling**
  As the Mission Operations have to be provided as a Service, the capability to satisfy and prioritize user requests, those compete among the service provider resources, has to be considered. Ground Station booking and imaging requests are typical examples in this field. New interest are emerging in the direction of coupling pricing strategies for the allocation of these services with scheduling algorithm, usually exploiting game theory concepts (e.g. Marked Oriented Programming [34]).

1.1. Thesis outline

The work is organized as follow.
Chapter 2 will describe the Mission Planning as a sub-system in the complex Mission Operation System.

Chapter 3 will describe the Mission Planning in different use cases.
Chapter 4 will introduce common modeling and solving algorithm in Mission Planning architecture.
Generalized Mission Planning applications will be described in Chapter 5, followed by a discussion on relevant technological opportunities in the IT field for a Mission Planning System.
Conclusions are finally reported in Chapter 7.

2 | Mission Planning in Space: a system overview

The Mission Planning and Scheduling (MPS) in space missions is a subsystem of the whole Mission Operations System (MOS). In the CCSDS standardization [9], the role of the MPS is identified by the required functionalities. As the standardization proposes a service-based architecture, it is relevant to underline the four correspondent main services:

- **Planning Request Service**
  It handles the asynchronous submission of Planning Request and associated responses and feedback. It allows the editing of a Plan at the activity level and updates Planning Events and resources.
- **Plan Distribution Service**
  It provides access to Plans and their current status.
- **Planning Management Service**
  It manages the planning process (initialization, status feedback and control) and updates the plan status.
- **Plan Execution Service**
  It controls and manages the execution of a plan, including start/stop and pause/resume actions.

Their capabilities and consumed data are reported in table 2.1.

Service	Capabilities	Data
Planning Request Service	• submit request • update or cancel requests • edit plan content • update planning events and resources • provide request status feedback • manage request definitions	• Planning Request • Plan • Planning Activity • Planning Event • Planning Resource • Planning Constraint
Plan Distribution Service	• retrieve plan or plan status • subscribe to Plan or plan status	• Plan • Planning Activity • Planning Event • Planning Resource • Planning Constraint
Planning Request Service	• Initiate, Monitor and Control Planning Processes • Update Planning Status • Manage Planning Definitions	• Plan • Planning Activity • Planning Event • Planning Resource • Planning Constraint
Planning Request Service	• Initiate, Monitor and Control execution of a Plan • Manage Planning	• Plan • Planning Activity • Planning Event • Planning Resource

Table 2.1: Capabilities and consumed data in the services recommended by CCSDS standards for a MPS.

In the same standardization documents, the three main constitutive elements of the plan-

ning process are identified:

1. Planning User: it submits requests to the Planning function and/or controls the Planning process. It usually receives feedback on the status of Planning Requests, the generated plans, and the status of the planning process. It is not a Planning function itself, but it is a user of Planning data and services. A real system may contain multiple types of Planning User function.

2. Planning: The function responsible for performing Mission Planning. Planning Requests are received and feedback on their statuses are provided. Planning Users may perform high-level control on the planning processes. The output of the Planning function is plans, which may be retrieved by Planning Users and distributed to Plan Execution functions. Planning is typically a user of the Navigation function and may receive predicted Planning Events related to orbital information, attitude, or slew time information. It often negotiates the scheduling of ground stations.

3. Plan Execution: It is the function responsible for executing a Plan (or part of one). This is typically distributed between space and ground segments. It is not directly part of the MPS but it does support a common model of the Plan in its interface with Planning. It allows external control of the Plan Execution process and provides feedback on the execution status of the Plan to Planning.

Their high level interactions are depicted in fig. 2.1.

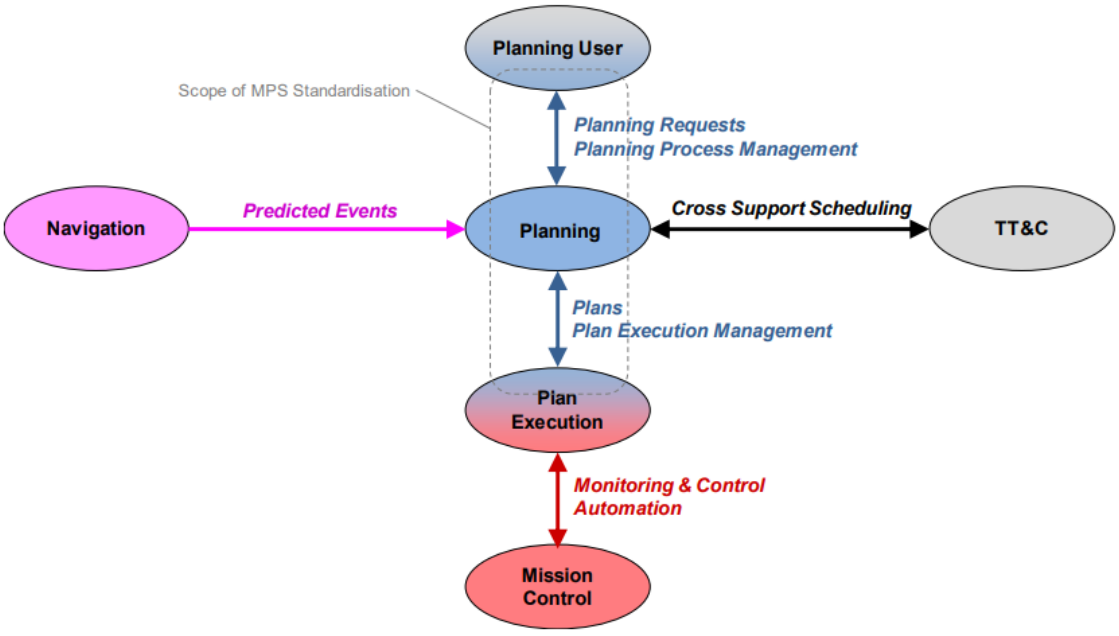

Figure 2.1: Function involved in mission planning [9]

In order to provide an example of the complexity of the integration of the Mission Planning System in the Mission Operation System, in fig. 2.2 is reported the functionalities involved in a typical NASA Deep Space missions.

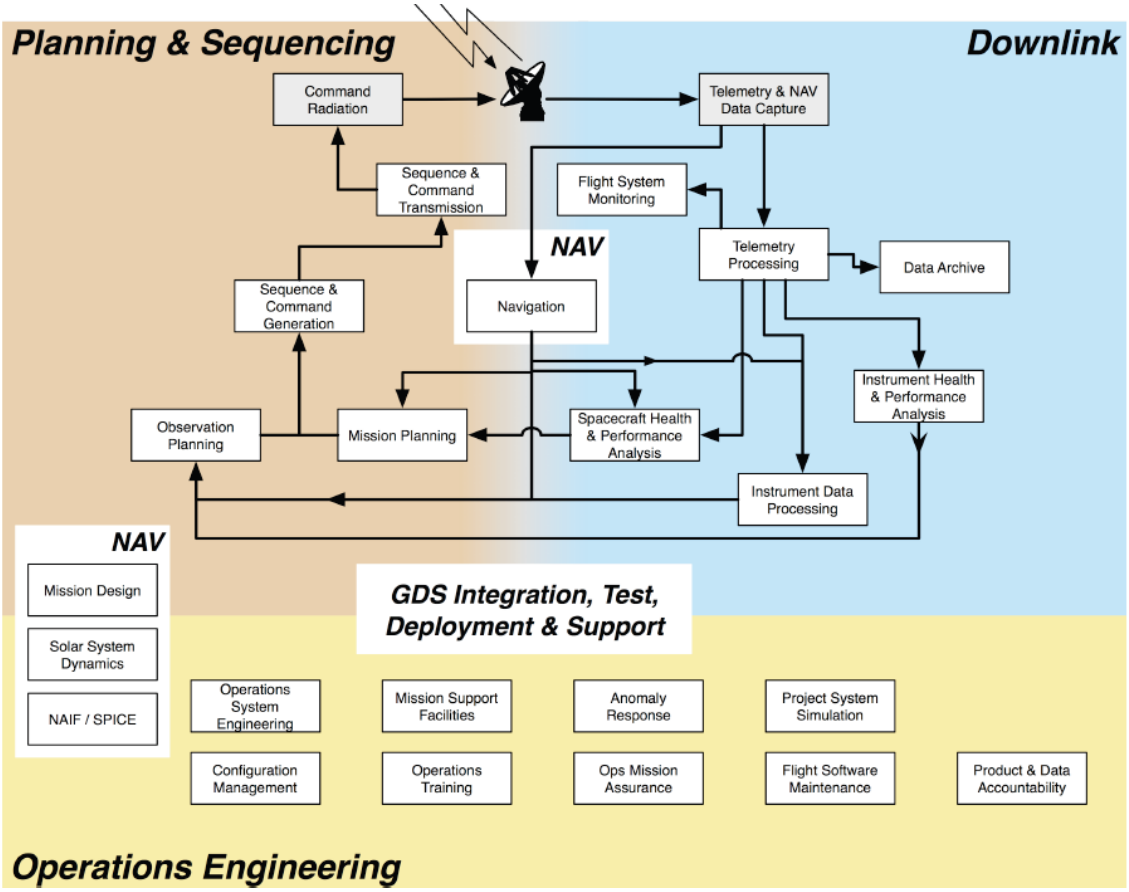

Figure 2.2: Common function of a Mission Operation System for a NASA Deep Space mission [7]

2.1. Exchanged messages with the Mission Planning System

As part of the MOS, the MPS has to interface with other entities. The main messages exchanged are:

- Mission Planning Data:
  - Planning Request;
  - Plans;
  - Planning Process Management Data;

  - Plan Execution Management Data;

- Navigation and Timing Data:
  - (Predicted Orbital) Event Message;
  - Orbit Data Message;
  - Conjunction Data Message;
  - Spacecraft Maneuver Message;
  - Pointing Request Message.

Other types of navigation data, such as slew time estimation, can also be an input to the planning function. The exchange of messages implies the need of a messaging layer within the mission entities, in CCSDS standard reffered as Message Abstraction Layer. The standardization identifies the following technological needs to implement it:

- a language binding the services to the implementation of an API;
- a technology that defines how the resulting messages are carried over a concrete message transport protocol;
- a technology binding that defines how the abstract messages are encoded in a concrete format.

It is important to use a common representation between entities of the MOS. This allows the reconstruction of the mission history and an easy interface.

2.2. Planning cycle

The CCSDS Green Book on Mission Planning illustrates how in order to handle the complexity of the planning process, it is usually divided into multiple planning cycles with progressive levels of details. This means the adoption of different planning horizons (temporal distance from the first to the last instant in the plan) and different time resolution (atomic time unit considered while planning). Commonly adopted planning cycles are:

- Long Term Planning: the horizon is in the order of several years to several months (in specific case, even weeks). It is typically concerned with the achievement of Mission Objectives, impacted by the long term spacecraft orbit and attitude planning, with some initial consideration on resource usage and constraints.
- Medium Term Planning: typical horizon in order of several months to several weeks. In this cycle a higher level of detail is considered while dealing with orbit and attitude

planning, as well as the allocation of resources.

- Short Term Planning: typical horizon is in order of several weeks to several days (even hours in specific cases, as robotic surface operations). In this cycle the detailed planning of spacecraft and payload is carried on. Final orbital and attitude information are utilized and resources and constraints are considered at the bigger level of detail, to ensure the plan executability.

Additional cycles can be utilized (as the very short term planning, mainly used for scenarios which require fast responsiveness and surface operations).

2.3. Centralized and distributed planning

Planning may be centralized in a single function or distributed over multiple entities (ground or on-board), where each entity is part of the planning process. The decentralization may be driven by various factors, such as the existence of entities with specific knowledge, ownership or governance. With functionalities, also responsibilities are distributed. Distributions also means need for iterations: changes, conflicts and assumptions often require feedback from other planning entities.

2.4. Information Model and Data Model

As it has made evident through the previous chapter, the Mission Planning System is tightly coupled with other mission entities. In order to guarantee a correct information exchange between them, an information model is hereafter described. It is possible to evidence three characteristic for a Planning Data item:

- a unique identity, which identifies the data item over the mission lifetime;
- a definition that is associated with that data item and that may be changed over the mission lifetime. Static information about the data item are included, as well as variable parameters, hierarchy of contained data items and fixed constraints;
- the instance of a data item, which specifies the value of parameters, relationship to other data item instances and dynamic constraints.

The Data Model to be exchanged can be mainly categorized as reported hereafter.

2.4.1. Planning Request

Planning Requests are the principal inputs to the planning process. It is the exchange of information between a requester and the planner. Different types of request can be allowed:

- request to plan an activity or a set of activities;
- request to achieve a goal;
- request to use a Plan as an input to the planning process;
- request to modify the content of a Plan;

Planning Requests typically needs to hold reference to the constituent information items that are required by the planner and agreed by the interacting parties. A request refers to Planning Activities and, as basis of the request, may contain references to Planning Events. Information about resources can be exchanged as part of the Planning Request.

2.4.2. Plan

A Plan is the principal output of the planning process. The Plan is a container of selected Planning Activities, optionally associated with Planning Events. The usage of Planning Resources may be contained in the Plan as well. Hierarchical and distributed planning use a plan as input for another planning process. In this case, the Planning Request will refer to an entire Plan. As planning can be iterative, a Plan may overlap with a previously defined one. This introduces the idea of the identified predecessor of a Plan, as well as the unique identity of the same planning data item utilized in the following iteration of a Plan (to avoid ambiguity and duplication).

2.4.3. Planning Events

Planning Event indicates the achievement of a given condition, typically temporal or positional. A Predicted Event is the instantiation of a Planning Event with an associated value, which will be used during the planning. The Predicted Events are provided mainly from external services (as can be the Navigation function). These elements are not part of the Plan but referenced from the Plan. If the Predicted Events is generated internally, it will be contained in the Plan. Values associated with Predicted Events can be extended by an uncertainty delta range. It is expected that this range decreases as the incoming events become closer. Planning Events should support multiplicity (e.g. multiple observation window for a given target).

2.4.4. Activities

Activities are the atomic commands of which the planner is aware of. They transform the modelled world from a set of states to another. In the branch of classical planning, this is often modelled as a set of Preconditions and Effects on the world states. Planning Activities should support hierarchy, as well as arguments (parameters) used to instantiate a specific Planning Activity from its generic definition. Definition of a Planning Activity can change over time. It is then important to maintain a unique and unambiguous relationship between Planning Activity and proper definition. The definition of a Planning Activity should be known by both the planner and the interacting part. In [33], the authors identify three main categories for the Satellite Planning and Scheduling problem:

- Mission Activities are the activities that accomplish mission objectives, support mission objectives, manage system resources and maintain system assets.
- Maintenance activities are payload calibration, power gathering, thermal dumps and orbit adjusts.
- Support activities are uplink commands, downlink telemetry, downlink mission data, space ground communication link.

2.4.5. System Constraints

Constraints are central to mission planning systems, and many of these constraints come from the Mission Operations System.
Mission Operations System is the integrated system of people, procedures, hardware and software that executes space missions. It has several planning functions: Mission planning (how and when the s/c will act), Activity planning (command sequence generation), Attitude Determination and Flight Dynamics planning (typically distinct from mission and activity planning, determine where the s/c is and where it maneuvers), Communication planning (again distinct, who to communicate and when).
Science targets, science instrument or payload constraints, and preferences for science payloads and instruments are typically input to mission and activity planning. Attitude and flight dynamics, communications, surface operations, and science planning provide input to mission and activity planning, but can also be constrained by them.
Before commanding the spacecraft, mission operators typically transform plans into sequences that simulators and other tools validate. Finding discrepancies in this process affects cost, schedule, and risk. Sequences are lists of fine-grained spacecraft commands. Simulation is used to determine sequences' time and other resource constraints. The exact

simulation used depends on the sequence's origin (e.g. from instrument team or from the manufacturer team). Often, simulation is used only to check against flight rules.
Flight rules are constraints and best practices for system operations to ensure mission safety and success. These rules provide essential planning system input, but they are typically stored as "human-readable" documents.
Constraints can change greatly before a mission. Mission planning systems must accommodate these changes and be validated at low cost. As an example, a mission's target can change. This can eventually imply an orbit change, having consequences on the planning system. Same for communication coverage change or s/c communication changes. It can appear that these changes only affect the "plan" itself, while often rules and constraints governing the mission may need to change as well.
In [33], the author categorizes constraints as:

- Time constraint: imposes a relation between an activity and another activity or a time instant. In order to express them temporal relations, Allen's interval algebra [2] is often used;
- Parametric constraint: imposes a maximum or minimum level for a parameter;
- Capacity constraint: defines the maximum number of occurrence of a specific activities over a time interval;
- Geometric constraint: imposes a specific geometry related to the spacecraft, payload, antennas, ground points and celestial bodies;
- Connectivity constraint: imposes the existence of other activities at the same time.

For Earth Observation Satellite scheduling problem, [30] lists some of the most common constraints:

- Revisit limitations: observation targets must be visible to the satellite in order to perform a data acquisition. Due to orbit conditions, this means that only few observation opportunities are available for a target at a given time;
- Time required to acquire data: image acquisition requires time. For example, the LandSat imaging task requires 24 seconds;
- Limited on-board data storage: acquired data are stored on-board and need to be sent on ground to allow acquisition of other data;
- Ground station availability: in order to downlink data, the satellite needs to establish a communication channel with a ground station. This requires visibility and availability of the ground station;

- Transition between look angles (slewing): both instruments or satellite can change attitude (depending of the platform design). Proper time distance needs to be provided between task in order to guarantee feasible slewing;
- Pointing angles: Highest resolution is obtained when the target is on the satellite ground track;
- Power availability: satellite platform have restrictive power budget. Furthermore, the power production with solar arrays is often dependent by the attitude of the platform;
- Thermal control: shadow/light cycle exposes the spacecraft to frequent changes in the thermal environment. This can affect the usability of payload sensors;
- Coordination of multiple satellites: in constellations it is needed to split the workload between the different agents. In doing so, resource balance and reliability of the plan should be considered;
- Cloud coverage: if cloud are limiting for the mounted payload sensor, observation of a target should consider cloud condition above it;
- Stereo pair acquisition or multiple observation with different sensors: these kinds of tasks require coordinations between activities.

2.4.6. Planning states and resources

Planning Resource is the modelling of a state of the system through the adoption of abstractions and assumptions. The implementation of such elements is needed in order to trigger the execution of Planning Activities and constraints them with respect to the desired resource time evolution. While Planning Resources are mainly handled internally, it should be accessible by other MOC elements. Initial value of a given Planning Resource should be settable by the proper entity.
A first distinction on spacecraft state and resources made by Kaslow [33] is between discrete and integral. Discrete states have a finite domain and their instantaneous value is subject to constraints. Integral states are instead constrained considering their integral. An example of the former is the On-Off state of a camera. For the latter, it can be the data stored on-board.
Some classes of Planning Data, including both Planning Activities and Planning Events, may be instantiated multiple times. In this case, there may be multiple instance objects associated with the same definition, each requiring its own unique identity in addition to that associated with the definition. Planning Resources are singletons with a current or

predicted state. Stand-alone Planning Constraints have no associated definition, but may appear in the context of a Planning Request or Plan.

2.5. Solution Approaches

In general the solving loop can be divided in : Decision, Simulation, Reasoning. The reasoning can be on human-side, on software-side or performed by a combination of the two.

2.5.1. Manual Planning

Decision: The Mission Planner will generate a set of activities to be performed in order to fulfill requests. This list of activities will be then scheduled (i.e. starting time, duration and any other decision parameter are assigned for each activity).
Simulation: The so generated schedule is passed to a simulator in order to propagate different aspects of the plan (as onboard resources, visibility windows, physical model,. . .).
Reasoning: As the plan is simulated, manual adjustment are made to the plan and the process is repeated until a valid solution is generated. Then, the generated Plan is usually discussed with the rest of ground segment for approval.

It should be noted that, for some missions, the Mission Plans can be defined during mission preparation as blueprints for Operational Plans. This is typical for missions which require the execution of a set of activities mostly defined during design phase.

2.5.2. Automatic Planning

The process is very similar to the one mentioned in the Manual Planning with a main difference: the decision on which values assign to starting time and duration of activities (or other input variables) is made through an iterative search process executed by a software component. The assigned values are used to simulate the plan and the results of the simulation are then returned to the software component, that can perform some sort of reasoning and iterate on the choice.

2.5.3. Mixed Initiative Planning

Different shades exist in between the fully manual and fully automatic approaches. These are typically referred as mixed-initiative Planning: the Mission Planner will perform manual planning, and a software component will provide possible strategies to solve the

plan's conflicts.

2.5.4. Early Commitment Planning / Least Commitment Planning

The difference between these two approaches is whether a task is assigned to an instant (early commitment planning) or to conjunction of intervals (least commitment planning) during the decision step.
The first kind is the most immediate: the tasks are assigned a specific values to each decision variable and the so produced plan is evaluated.
In Least Commitment Planning instead, each input variable is assigned to an interval of possible values. At each iteration the assigned interval is shrinked by imposing additional rules (on which it is possible to perform backtracking) in order to resolve conflicts. Due to the difficulty and specificity of the second method (as conflict detection and deconfliction can be tricky to implement), applications typically implement Early Commitment Planning.

2.6. Problem types

2.6.1. Single/Multi-agent

The main difference between the a single agent and a multi-agent scenario is the presence in the latter of multiple assets which can perform activities. Multiple agents can share the same capabilities or be differentiated. In multi-agent scenario, which satellite should perform the activity became an input variable of the planning problem.
Dealing with Multi-agent planning means facing an increased complexity. In order to reduce complexity it is possible to perform a decomposition of the planning problem: activities are assigned to the assets as first step under some sort of simplification, and then planning is performed for each asset with a more refined approach.

2.6.2. Task-oriented/Goal-oriented

In a task-oriented problem formulation the scope is to assign input values to a list of defined tasks, subjected to some constraints, typically trying to optimize an objective function.

In a goal-oriented problem formulation the scope is to achieve a goal with a series of tasks. The type and the quantity of tasks which have to be executed are not defined, while it is defined the output what the plan should achieve. This problem formulation requires a relevant level of details to model and simulate the effects of each task. It is also clear that the search space is larger than in task-oriented formulation.

2.7. Functional Architecture Module (FAM) Decomposition of the Mission Planning System

In the context of Satellite Operations Group (SOG) located at the Rutherford Appleton Laboratory (RAL), the decomposition of the Mission Planning system in terms of implementation agnostic Functional Architecture Modules has been executed [12]. It is relevant for the work of this thesis as it helps identify main functions and functionalities involved in a planning iteration. The identified FAMs are:

- Final plan building method architecture;
- Adjustable specific plan building method architecture;
- MS iterative architecture;
- MS anchoring object generation architecture;
- PMS plan update architecture;
- Managing PMS architecture;
- Sequential and parallel PMS architecture.

2.7.1. Final plan building method architecture

The purpose of a planner is to build a plan. A planning process can be decomposed in the problem formulation and the Plan Building Method (PBM). The problem formulation expresses a set of constraints describing what rules can be checked to identify a situation that invalidates the plan. A Final Generic PBM is then typically a constraint based algorithm.
In this analysis, the authors identify *Generic PBM* as a method that can be used regardless of the content of the constraints, while a *specific PBM* varies as the system constraints have to variate. The latter is often based on backwards reasoning, resulting in a rule-based algorithmic implementation.
Generic PBMs are less costly to configure, it is easier to reconfigure them and provide

faster reaction to unpredicted scenarios. However, the required work of abstraction of the constraints and their expression is a very complex task, bringing often the implementation towards Specific PBMs.

2.7.2. Adjustable specific plan building method architecture

An adjustable PBM can be used to solve Generic PBM or Specific PBM. A Specific PBM Generator performs backwards reasoning on the planning input (as the constraints) in order to generate a Specific PBM. From an implementation point of view, the PBM generator is likely to be executed by human, also if programmatic implementation could be considered. This FAM should be used if at the moment of decomposition the constraints are not completely known or are in constant change.

2.7.3. PMS iterative architecture

The scope of this FAM is to produce a plan compliant with the constraints provided by means of iterations. It can be decomposed in two sequential component: an Initial Plan Generator, which generates the first guess for a plan (also if not constraints-compliant) and an iterative plan adjustor. It checks the content of the initial plan, correct it in order to resolve a constraints violations till the plan satisfies the constraints.
The adoption of an iterative approach to the planning input is particularly suitable for scenarios where planning inputs change frequently, and thus is not convenient to anticipate all the possible problems as required for a sound PBM.
While selecting the proper iterative architecture, complexity of the handled constraints and time and cost available to produce a plan has to be considered as driver.

2.7.4. PMS anchoring object generation architecture

An anchoring object is an element of the plan to which the planning objects can refer. It is typically the case of events, in the form of both intervals and instants. The purpose of an anchoring Object Generation Architecture is to produce such entities though a well identified process.
The anchoring object generation architecture, other than producing the events, has also the task of producing the constraints that refer specifically to that event. Whether historically this task has been tackled by humans for most of the mission, the emerging complexity of new space mission is requiring more and more a programmatic approach.

2.7.5. PMS plan update architecture

A plan update is the result of a combination of two factors: the modification of the inputs of the plan generation system after the plan generation has started and the impact on the quality of the new produced plan. Therefore, the purpose of the PMS plan update architecture is to adjust the plan content in response to planning input changes before the latest time available for update.
In a Plan Update Architecture, the Plan Update System (PUS) component is ready to be triggered at any time planning inputs vary. It can be executed in parallel with the PMS. When designing for human or informatic resources to be allocated on this component, it should be considered the trade-off between cost and update time. A faster update will guarantee fast response to unpredicted input changes.

2.7.6. Managing PMS architecture

We can distinguish the Managing MPS and the operation MPS. The former controls the planning of the latter through the issuing of a managing plan. This is an operation plan that informs on how and when scientific operation planning should be performed. The managing MPS needs to be considered only if the complexity of the managing plan is considerable. It will typically increase with the frequency of scientific planning. The planning domain size is also relevant: if it is big, it will be capable of handling more aspects of the mission lids, thus reducing the need of a Managing MPS.
In sequential planning the output of a PMS will be the input of another PMS: the planned objects of the former become anchoring objects of the following one. In parallel planning instead, each planning process is executed independently from the others (however their output can all be input of another PMS).

2.7.7. Sequential and Parallel PMS architecture

The intent of sequential and parallel planning is to orchestrate a PMS as an ensemble of Sub-PMS executed sequentially or in parallel.

3 | Use Cases

The following chapter will report and discuss the different use cases identified by the CCSDS recommendations.

3.1. Constellation

Constellations are missions with more than one space asset flying in a certain formation. The size of a constellation is progressively growing over the last decades. The scope of these missions spans from earth observation to telecommunication and navigation.
The main need while planning for a constellation is to consider the operational life of each single spacecraft while coordinating them in order to achieve mission goals and desired constellation performance.
While producing Long Term Plan (LTP), main limitation is the launches availability and commissioning [4]. Medium Term Plan (MTP) is usually concerned by the mission events, while Short Term Plan (STP) produces detailed timeline scheduling.
During planning execution, feedback from each satellite and ground system are collected and used to update the state of the constellation plan. A regular update on orbit information from flight dynamics service for every satellite is needed in order to produce a schedule with the correct timing of orbit events and tasks.
Constellations which provide a service to end users have to pay special attention to Planning Request handling. For example, the execution of a station-keeping maneuver on one satellite would exclude this satellite from service provision for a certain time span and at the same time require more communication resources because of critical operations requirements. This needs to be compensated by the availability of a sufficient number of other in-service satellites and potentially the provision of additional ground-segment resources.
Significant for a constellation are the resources shared among the spacecraft, as these create interdependencies betweens plans. One example is the limited availability of telecommunication contact time, which depends on the stations available and the duration of routine and special contacts. Other constraints can be imposed over the constellation,

such as the adherence to some formation flying constraints (phase angle, distance,...).
The output of the planning process is a file (or multiple files) containing the detailed plan. It is usually automatically loaded from the Mission Control in order to be executed.

3.2. Earth Observation

Most of the earth observation missions are in Low Earth Orbit (LEO), but some can be located in geostationary ones (GEO).
Eventually, multiple spacecraft can be utilized and coordinated. Most of the tasks requested to space segment require specific physical location (either on surface or in orbit) rather than to timed events. Some EO missions use an orbit with a repeat cycle concept (i.e. same geographical location or illumination condition after N orbits). This Orbit Position Scheduling planning concept can be defined either within time intervals or statically uploaded on board for the whole orbital cycle in which the operation profile is statically position-fixed, and an orbit with a repeat cycle is used (e.g., always observe Germany on the third orbit of the cycle).
Regarding the extent of the planning interval, a shorter and ad-hoc planning mechanism is defined for periods when intense/dynamic interaction is needed (e.g., in-orbit commissioning) or in case observations are not based on a static or predefined profile but need to take into account user requests, e.g., to observe a specific area for steerable payloads.
Planning for EO can be implemented both as an open or a closed loop (respectively, with or without feedback). Due to the operational simplification of EO mission, it has been pushed towards open loop planning approach. The execution feedback will consist of offline reporting, as TC acknowledgement, that will be used as input on future planning.
Navigation for EO satellite mostly relays on GNSS data. Usually planning is performed on-ground, considering a static orbit or, if not possible, using a predicted one.
Payload Planning Requests for EO mission usually defines high level activities, which can be both time or position tagged. These requests are deconflicted and then passed to the Satellite Mission Planning center, to incorporate them with the needed platform operations, expanding the so produced plan into command sequences and deconflicting it again. Uplink/downlink activities have to be requested to the Ground Station Service.
Planning for EO mission typically involves conflict resolution for resources allocation (e.g. power). While the Payload Planning system will plan in conflict by design, the platform planning system will produce a plan which is compatible with operational requirements and capacity of the platform.

3.3. Fly-By

During a flyby the spacecraft follows a trajectory bringing it close to the target body, but not into orbit around the target body. A flyby may be part of a larger space mission, in which it only concerns a specific mission phase. During a flyby mission the duration of the science opportunity windows is of short duration. Therefore activities are usually pre-planned for the estimated time of closest approach. Due to the short duration of close approach, the plan is typically executed in open loop. Examples are the Cassini flybys, the New Horizon Pluto flybys, and the Rosetta flybys of Lutetia and Steins, as well as the flybys of comet Churyumov-Gerasimenko.
Navigation measurement and updates are executed during far approach and the relative adjustment made at closest approach. Sequencing is usually performed relative to navigational landmarks to facilitate updates to the planning due to the new position information. Due to the big relative velocity with the observed body, geometry of the orbit and pointing of the probe are usually crucial trade-off in planning.

3.4. Inter-satellite telecommunication

Typically located in geostationary orbit. The communication link can be provided from ground to space or back, or space to space in the case of a data relay mission. Communication can be radio or optical. Typically this kind of mission is more service oriented with respect to Planetary or Earth Observation missions.
Since inter-satellite comm involves two satellites, the frequency of each planning cycle depends on the planning cycles of each involved satellite.
In GEO, constant availability of telecommunication with the spacecraft allows replanning adjustment also 15-25 before transmission.
The resources for a inter-satellite communication mission are usually pointing capabilities, on-board memory and opportunity of the spacecraft and the other terminal for the communication. The mission feedback is typically an update of the status of the execution on-board.
Navigation data need to be precise and are provided to the Mission Planning System from Flight Dynamics.
The output of the planning process needs to be interfaced with the other services depending on the mission: telecommand, plan data, status of planning request need to be provided in proper format.

3.5. Observatory

An Observatory mission is normally located in a High Earth Orbit or Lagrange Point, to reduce the effects of the Earth's environment. Sometime also LEO can be used. The Mission Planning in this context is driven by the scientific needs behind the mission. Individual scientists can request observations.
The planning cycles are composed of some long-term activities (i.e. routine calibration, recurrent high-priority tasks) that are not movable or, if so, have a higher priority with respect to other tasks. Long term plans are influenced mainly by the scientific opportunity, medium term cycles by certain mission conditions (orbit stability forecast, availability of spacecraft maintenance windows,...) and short term cycles represent the plan request to the mission operation center for execution. Lately discovered opportunities can be answered only in case of fast replanning.
During Long Term Planning, the request are screened in order to extract a pool of entries for later planning, based on a priority value. In Medium Term Planning, pointing and visibility conditions and requirements are considered. Short Term Plan has to ensure the final optimization and feasibility of the medium term plan with respect to environmental conditions. A Short Term Plan usually lasts from a day to several days.
Execution feedback is provided not only in terms of execution status, but with information regarding pointing and stability, as well as environmental conditions (radiation, thermal,...). Capability to return feedback also to scientific institutions that make planning requests should be provided.
It is not atypical in this kind of mission to have the navigation solution having access to the planning service in order to save iterations in the planning problem, as this is typically demanding in terms of pointing and slewing.
Requests are typically made for an instrument activity at a fixed time /fixed target/open time/open target in the sky. Usually requests also incorporate the conditions and constraints to perform the activity. Pointing is also included, either in form of of celestial coordinates or as quaternions that performs translation from a previous attitude or reference frame.
The requests typically use template structure or fixed upper level construct to simplify the translation from activities to telecommand.
Activities regarding a specific instruments can be mutually exclusive of the usage of another instrument. While planning, this has to be taken into account, as well as classical resources (power, memory, downlink bandwidth, attitude/slewing, antenna pointing,... Pointing constraints should be taken into account in order to avoid illumination in sensitive instruments, thus defining the region of sky allowed for observation.

The planning process often consists in the optimization of the priority of the pool of the selected observations for the long-term Plan versus the resources and constraints, ensuring to the local medium-term planning an high optimization level, that maintains a good level of overall optimization in the long-term process.
The output is a request to the Mission Operations Center consisting of a Plan covering the short-term period. This Plan addresses either activities (i.e., higher blocks) or TC sequences directly, and can be time based or event based. It is typically composed of a pointing subset of information linked to another subset dealing with instrumental configuration.

3.6. Planetary

In planetary missions the spacecraft typically carries multiple payload instruments. Mission Objectives and instruments greatly influence the spacecraft schedule.
In addition, non-imaging instruments are often used. The planning process typically follow the three phase planning approach (long, medium and short term planning), as in the BepiColombo and Cassini missions.
At Long Term Planning stage consideration are made on key geometric parameters, as distance with the planet and illumination condition. Medium term planning horizon varies from few days to months. The common aspect of Medium term planning is the detail at activity level. The plan is often based on events, which can be directly correlated to time. Short term plan consider days to weeks. Updated orbit and constraints information are collected and used as input in order to generate a detailed list of command.
The execution feedback is provided mainly by the TC of Science operations. It could also be provided for the operation of eventual relay for telecommunication.
In planetary, the spacecraft pointing is typically an important constraint. In order to avoid pointing conflicts, the Mission Planning system has to interact with the Navigation capabilities, manually or automatically.
In the iterative planning process, pointing requests are provided from the Mission Planning process and navigation validation is asked. In order to reduce the number of iterations, rules based on the worst case boundary can be applied as a baseline for the planning system.
In more automated approaches Navigation function can be exposed by a dynamic interface to the MPS (e.g. Web service, API,...).
Typically, a Planning Request for planetary mission contains:

- references to operations templates for the requested activity;

- information about resource consumption (either as explicit information or implicitly through use of operational templates): global constraints, geometric constraints (e.g., distance, phase angle), and dependency constraints on other activities (e.g., not in parallel with S/C maintenance slots) should be considered;
- scientific justification or rationale for the requested activity;
- priority of the request is provided in different formats, e.g., a numerical priority level, a prioritized target list, etc.

Main resources and constraints in a Planetary mission are:

- Pointing
- Power
- Data Storage
- Geometric constraint, affected by illumination conditions
- Thermal constraints, affected by illumination conditions
- Logical constraints, considering succession of instruments to achieve scientific objective
- Timing constraints

Planetary mission have very clear high level scientific objectives. These lead to the choice of the scientific instruments to be carried on board, typically provided by different institutions. This can imply the implementation of a federate planning, where knowledge and responsibility for the activities of the instruments are handled by a team, to be then coordinated by the platform planning team. Planning request justification should often contain complementary information on the scientific justification. The rationale behind the Planning request however is often captured through keywords (e.g. activity name) that allow tracing to operational rules.

3.7. Robotic Servicing

In this kind of missions the spacecraft performs dedicated activities on another spacecraft (served spacecraft).
MPS of Robotic Servicing missions may be very specific and require real time operations with ground in the loop, or may be based on more autonomous capability of the servicing vehicle.

Three level classical planning cycle is often implemented. Execution feedback is often provided on low level activities, due to the needs of robotic operations.
The Navigation service should provide specific approach vectors and angles, as well as spatial reasoning. Navigation is clearly a central aspect in the mission.
Requests can space from high level requests (e.g. refueling, module replacement,...) to detailed activities.
In addition to typical resources, other 'robotic' resources may be relevant, such as occupation of portions of space.

3.8. Surface Operations

Surface operations missions include the utilization of rovers on Solar System bodies. It is frequent in these applications to have the need for relay of telecommunication.
Mission Planning is typically organized in hierarchical levels. The MSL (Mars Science Laboratory) has Long Term Plan (LTP), supra tactical (several sols, manage mainly downlink cycles and time crucial operations) and tactical planning (one to several sols handled in a single uplink cycle).
Execution feedback includes rover position and robotic state. Resources states (e.g. energy, data volume,...) are returned as well. Navigation service should provide rover location and arm position estimates.
Planning requests can include a position to be reached, a measurement to be taken or an activity to be carried on. With respect to classical resources, path planning and planning for robotics are typically considered as additional resources.

3.9. Survey

A survey mission is normally located in a high earth orbit or lagrange orbit. It is a typical mission type in space astronomy missions but might be used as well for Solar System exploration missions.
Mission Planning for these missions is usually based on predefined rules for scanning, which determine how the celestial sky or target body is being observed. This is typically the case for cosmological missions or exoplanet finder missions.
Planning cycle is based on predefined mission planning rules, but in some cases latest knowledge of the environment or platform conditions can bring to last minute changes or updates. Execution feedback is provided by telemetric history and housekeeping telemetry, as well as ancillary products as attitude and orbit history.

Flight Dynamic routines make requests to Mission Planning, while it is uncommon to find interfaces in the opposite direction. Resources are implicit in the definition of the scanning law, following operational templates.

4 | Satellite planning model and solving algorithms

In the space domain, Mission Planning and Scheduling can differ from the typical conception it has in the Artificial Intelligence (AI) domain.
Planning is typically conceived as the problem of finding a sequence of actions that brings the world from an initial state to a desired state. In scheduling instead, the set of activities to be performed is already known, and their execution times and dedicated resources need to be defined.
In space, this distinction is not so clear. It is typically referred to as Plan for high level representation, and as Schedule for executable list of activities.
As whole, Planning and Scheduling has the objective to assign resources to tasks, while respecting a set of constraints and requirements.
Tasks are abstractions of real word actions, and map the effects of an action in the model scenario. Resources are a tool to represent anything that can be allocated, produced or consumed in the modelled scenario (e.g. money, personnel, energy, time,...).
In order to provide Plan Building Methods (module identified in the FAM [12]), it is necessary to model the planning problem and provide algorithms able to solve it.
A characterization of a planning application is the search space that will be built. In particular, it is possible to identify:

- **state space representation**: it is the closest to classical planning as it relies on logic and in particular first order predicate. Action are usually characterized by preconditions and effects on state variable. The solution method are still derived by classical planning: Boolean satisfaction, model checking and so on
- **plan space representation**: it is the search space composed by all possible plan. The usual way to represent them relies on the chronicles (or timeline) formalism (see section 5.1 for more). A classic technique in this sense is the iterative refinement planning ("refining" one plan violation at the time).
- **policy space representation**: it is usually more related to online decision making,

and most of the studied techniques rely on some sort of learning.

After building the search space, a method to explore it is needed. Here we lie in the field of search techniques and their typical classification, not discussed further.
Since nearly all the planning problem are NP-hard, a way to reduce complexity is needed. In this context emerges the application of heuristics, both domain independent (usually derived from classical planning, e.g. ASPEN incorporate some implementations) or domain dependent (this is what is usually done in space through Knowledge Engineering). While research is typically interested in the former, industries mostly applies the latter.
Reasoning and inference in planning provides algorithms and structures to handle the constraints and propagate state variables and resources. In particular, constraint management allows to rapidly check the validity of a plan and define the feasibility region for a planning problem.
At a representation level instead, different type of entities exists. Time constraints are a specific type of constraints. A big research effort has been carried on since the first nineties to identify structures and algorithms to manage them. The most relevant implementation is the Simple Temporal Network (STN [20], extended to Simple Temporal Network with Uncertainty STNU to handle executability in an uncertain environment [49]).
Actions are usually expressed as durative, and can be atomic or decomposable in other actions (HTN).
Resources are derived by the scheduling research field, and they have their own classification, in particular:

- depletable/not depletable (if depletable they are consumed after the reservation period),
- renewable/not renewable (renewable if they can be somehow produced)
- reserved/share (if they are composed by one or more).

Last elements of the representation are the state variables. These are used to represent the state of the system and of the environment. Values are typically discrete, but extensions are provided to handle continuous ones.
With the use of state variables the concept of controllability emerges: their value can be uncertain at a given time (both for an uncertain action duration that has effect on that state variable or for exogenous events).
Furthermore, the probabilistic aspect of the problem can be introduced. This is typically represented in different structures, as a Markov Decision Process with a stochastic transition function, or can be introduced as extension of the already discussed problems (such as with the introduction of Fuzzy logic).

[53] identifies four main parts in building a planning solution:

- Task Details: tasks need to be formally described. If the mission architecture requires so hierarchical decomposition should be provided;
- Model Building: a model should be provided in order to simulate the effects of tasks in the plan;
- Algorithm solution: such as deterministic algorithms, heuristic algorithms, search algorithms,...
- Evaluation Optimization: model evaluation, algorithm optimization

It is evident how between the parts there is a tight correlation and it is not possible to completely separate these topics. However the subdivision provides a background to analyze the different aspects of Mission Planning.

4.1. Task Details

A task is the translation in the planning model of a real world action. This modelling requires some sort of abstraction.
Creating the abstraction formalism for the mission will be the main task of the Mission Planner during the Mission Preparation Phase. In fact, the formalism should be flexible enough to adapt to the changes and challenges during the design phase and the operational phase. Defining the abstraction becomes than crucial: definition of tasks parameter and variables means deciding on which values the planning system should operate.
In the space domain the plan is often refined using a more detailed model as the plan horizon becomes shorter. Tasks should provide a hierarchical decomposition from high level granularity to commands.

4.2. Model Building

Modelling a planning problem means translating the real world scenario in a mathematical model. In order to do this, abstraction and assumptions needs to be provided, as the mathematical model can not be a perfect representation of the world. Hereafter are reported the most notable modelling strategies for satellite planning problems.

4.2.1. CSP

Born within AI research fields, the Constraint Satisfaction Problem (CSP) model is a set of mathematical questions over a set of objects whose state must satisfy a set of constraints.
The set of constraints can be encoded among tasks and resources in order to model the planning domain. Back in the 1994, Hall et al. [32] presented the visible time constraints in a Constraints Satisfaction Problem to model the space mission planning problem.
When a CSP is formulated in a disjunctive form, it is referred to as meta-CSP.

4.2.2. Graph theory model

In mathematics, graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices (also called nodes or points) which are connected by edges (also called links or lines). A distinction is made between undirected graphs, where edges link two vertices symmetrically, and directed graphs, where edges link two vertices asymmetrically.
In [28], the authors propose the modelling of a single satellite scheduling problem through the use of Direct Acyclic Graph.
The work reported in [38] introduces a similar modelling considering also Energy and Memory constraints.The advantage of the Graph Theory Model is that it is very easy to grasp and can be solved by the mature and effective polynomial time graph algorithm. But this model can't figure out some actual constraints, such as regional observation and the numbers of satellite side swings.
[21] introduced task clustering strategy, taking in consideration the drift angle, as needed for Agile satellites. The problem is modelled as a Travel Salesman Problem over a grid discretization, solved with an ant colony optimization method with branch and cut strategies.

4.2.3. Knapsack problem model

In [48], Vasquez proposed a multidimensional knapsack problem for the modelling of an Earth observation satellite planning problem. Knapsack problem has the advantage of being simple, resource constraints can be expressed with multiple dimensions each and have efficient optimal or near optimal algorithms. The model's disadvantages are similar to the graph theory: it can't express the complex constraints of the real world and is not suitable to be extended to multi-satellite mission planning.

4.2.4. Integer programming model

A more general problem can be provided with the Integer Programming Model. It can describe the satellite planning problem through linear constraints. Main advantage is the existence of commercial reliable solvers. Main drawbacks are the loss in efficiency as the problem size increases and the inability to model some nonlinear constraints.
In [5], Bensana solves an Integer Linear Programming model through exact algorithms (such as depth first branch and bound, Russian dolls search and best first branch and bound) and Approximate methods (e.g. Greedy algorithms and Tabu Search). In [51], the authors exploited three different algorithms to solve the mission assignment method on a single orbit. The algorithms are a simple priority dispatch method, a look ahead search and Genetic Algorithms.

4.3. Algorithmic solutions

To solve the previous model, different algorithms can be exploited.

4.3.1. Genetic algorithm

Genetic Algorithm (GA) is an optimization algorithm. It is characterized by global search ability and the possibility to parallelize computations. In this algorithm, an individual represents a possible solution to the planning problem. Each variable value in the individual is a gene. At each iteration, the best individuals will be selected and combined, to be then re-evaluated to select the new individuals which will produce the next generation. Mutations, as crossing over techniques, are often provided to increase the global search capability of the algorithm (and thus avoid local optimum).

4.3.2. Ant Colony algorithm

Ant Colony (AC) has been used to solve data transmission scheduling problem. It is a probabilistic technique which aims at finding good paths through graphs. Combination of ants and local search have become common in the optimization applications. In the application provided by [52], the algorithm has shown good scalability capabilities.

4.3.3. Simulated Annealing algorithm

Simulated Annealing (SA) is a global optimization technique that exploit the concept of annealing in metallurgy, which is a process used to remove defect from the crystal trough

heating and slow cooling down.

5 | Generalized Mission Planning

Automation of the planning process requires a significant amount of time and capital in order to develop a valid solution. In order to guarantee this type of service leveraging heritage code, NASA promoted the MOS 2.0 within the AMMOS initiative. In [7] the following AMMOS mission statement is reported:

> *"The AMMOS exists to provide multi-mission tools and services that enable mission customers to operate at a lower total cost to NASA, with comparable or higher reliability and performance, than would be the case if these customers acquired their own unique tools and services."*

This chapter is intended to discuss relevant topics to the generalization of a Mission Planning architecture. In particular timeline-based planning will be presented, with references to the most relevant application which relies on it (in particular APSI, ASPEN and EUROPA). It will follow a brief dissertation on the domain description languages for Mission Planning and their evolution.

5.1. Timeline-based Planning

A generalized Mission Planning system is based on the orchestration of software. In order to face a generalized approach, the use of Timelines has emerged as valid tool. Formalization of the Timeline concept can be found in [42]. A timeline is intended as common data structure for storage and communication between spacecraft planning and operations software elements. In fact, most of the spacecraft planning and operations processes are naturally expressed in terms of software tools that read timelines.
In fig. 5.1 can be noted how the operational process in a space mission can be based on the concept of timelines. The dotted line at the top represents the boundary between the Flight System and the Ground System. The arrow coming out of the Flight System represents the flow of telemetry. The purple and blue ovals represent the downlink and uplink process, respectively. These processes exchange information via the timeline storage in the middle.

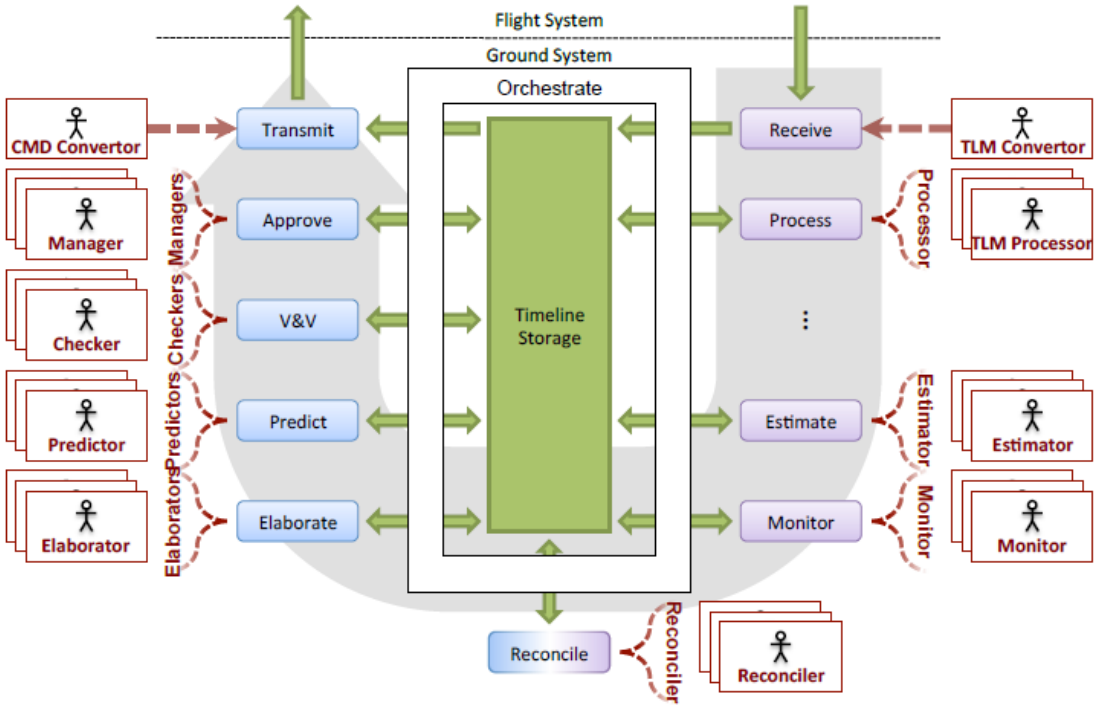

Figure 5.1: Timeline-based Mission Operations System Architecture [19]

Timelines are versioned and each version is immutable. Timelines thus form the syntactic and semantic method of integration of software elements, leading to decreased adaptation cost.
The authors identify major advantages in the use of this approach:

- decoupling of software segment, in order to allow independent adaptation and evolution, and avoidance of ad-hoc interfaces and data structure is provided;
- explicit interactions between software segment, in order to managed them as a system, thus reducing maintenance cost;
- Communications, Coordination, Information Model, and software are separable: it makes easy to switch to a REST API or deploy the software on cloud without the need or re-implementing the whole architecture;
- It facilitates the addition of smaller and more cohesive software segment to the ensemble, avoiding the implementation of a large monolithic application.

In [19], the author offers a detailed characterization and categorization of the concept of timelines. The categorization is organized in relation of the final user of the timeline, in

particular it is divided for Mission Operations and Control System.
Mission Operations timelines are:

- State Timeline: A state timeline is a continuous timeline that represents a state (a particular aspect or property) of the system being controlled over time. The codomain of a state (the values over which it varies) may be a continuous set of values (e.g., voltage in watts) or a discrete set of values (e.g., on/off, discrete mode of the system being controlled) or a set of even more complex functional forms. For example, the state may be constrained to a specific value (i.e., assignment) or to a probability distribution over the state values.
- Command Timeline: A command timeline is, practically speaking, a discrete timeline that represents the information received (or may be received) by the system being controlled for the purpose of changing its behaviour. Conversely, a command timeline represents the information that is sent (or may be sent) by the control system for the purpose of changing the behaviour of the system being controlled. Similar to a state, the codomain of commands may be a continuous range or a discrete range or something more elaborate.
- Measurement Timeline: A measurement timeline is, practically speaking, a discrete timeline that represents the information sent (or may be sent) by the system being controlled for the purpose of providing observation into its behaviour. Conversely, a measurement timeline represents the information received (or may be received) by the control system for the purpose of observing the behaviour of the system being controlled. A measurement may be sensor or instrument readings or any other data received from the system. Similar to a state, the codomain of measurements may be a continuous range or a discrete range or something more elaborate.

Categories for Control System are:

- Command Timeline (as before)
- Measurement Timeline (as before)
- Actual Timeline An actual timeline represents data at control system interfaces of a system being controlled. A control system has access to only two actual timeline categories regarding the system being controlled: Measurement timeline and command timeline. The state of the real system being controlled is generally unobservable. Nonetheless, a simulation of the system being controlled can produce actual state timelines that are observable by testers.
- Intention Timeline An intention timeline represents what the control system is trying

to achieve regarding the system being controlled. An intention timeline may be elaborated into more detailed intention timelines.

- Estimation Timeline An estimation timeline represents a control systems computed or analysed information over time regarding the system being controlled based on the available actual timelines, intent timelines, and/or other estimation timelines. An estimation timeline may be further classified as prediction timeline and trend timeline:
- A prediction timeline is a type of estimation timeline that contains values for times in the future (relative to when predictions are made) based on information from actual timelines, intent timelines, and/or other estimation timelines, including prediction timelines, available at the time. In estimation theory, this represents an estimate at a time point greater than available actual data, hence, prediction.
- A trend timeline is a type of estimation timeline that contains values for times in the past (relative to when trends are evaluated) based on information from actual timelines, and/or other trend timelines available at the time. In estimation theory, this is equivalent to smoothing (an estimate at a time point less than available measurement data), but may also include filtering (an estimate at a time point at the end of available measurement data).

A timeline based system usually has to support basic functionalities as:

- represent linear, possibly grounded time points for events such as activity scheduled start times, etc.
- represent finite and infinite states and propagate their evolution
- represent and check depletable and non-depletable resources and their constraints
- represent check time constraints which involves variables
- represent and check functional parametric dependencies among usages and activity parameters

In order to facilitate the analysis of a plan, Timeline-based system should support:

- the simulation of a placement of an activity (and separately or simultaneously propagate and model);
- the ability to query if a specific placement of an activity will violate the constraints;
- the ability to query for which time placements a task will not violate constraints;

- the ability to run arbitrarily coded constraint models to check complex constraints and activities (e.g. maneuvers, power, mobility, thermal);
- delete an activity and check the constraints;
- move an activity and check the constraints.

Soft constraints can be introduced in order to express preferences (e.g. prefer more or less instances of certain activities, prefer certain timing relationships (absolute and relative), preferences on resource usage, and other characteristics of the produced plan.

In [27], the requirements for a planning/scheduling application framework have been identified as the following:

- A hierarchical representation of activities
- The ability to reason about temporal relationships between activities
- Reasoning about resource usage
- Reasoning about arbitrary state attributes
- Reasoning dependencies between various parameters of activities
- Flexible representation of time
- Support for a wide range of search algorithms
- A graphical user interface for viewing and manipulating plans and schedules interactively
- A modeling language that is capable of expressing the reasoning facilities listed above, and a parser that translates user models expressed in this language to the system's internal data structures

The most relevant application of Timelines in the context of space mission are reported and described in the following sections.

5.1.1. Advanced Planning and Scheduling Initiative

Advanced Planning and Scheduling Initiative (APSI) was a research project funded by ESA. Its goals were two: create a software framework capable of improving the cost effectiveness and flexibility of the development of planning support tools, and on the other hand, using Artificial Intelligence planning and scheduling technology for space mission planning application.

The resulting framework is the Timeline Representation Framework (TRF) [11], implemented as a three layers architecture:

- a time and parameters layer: a common represents the information of the timelines, temporal information, and parameter information;
- a component layer: a middle level in which the modeller can model the components;
- a domain layer: an upper level is meant to maintain a shared representation of the plan.

The planning domain is modelled as a set of concurrent threads (timelines) and the problem is to synthesize a set of decisions to obtain a desired behaviour and to synchronize the threads. The TRF has been used in three use cases at ESA within APSI. In MrSPOCK, planning application of the Mars Express mission, Science Operations and the Flight Dynamics teams iteratively refine a plan containing all activities for the mission [10].
The planning process starts with the Long Term Plan (three months) and it is gradually refined until a Short Term Plan (one week) is generated. This process continuously leads to weekly STPs, finalized every two days into executable plans.
The optimization used for MrSPOCK was based on Genetic Algorithms (GA). GA is a well-known and effective computational paradigm for function optimization inspired from the study of population genetics. GA is combined with a constructive heuristic procedure that instantiates the temporal plan which represents the output of MrSPOCK.
The second use case was supported by the INTErnational Gamma-Ray Astrophysics Laboratory (INTEGRAL) long term planning team of the Integral Science Operations Centre (ISOC) based at ESAC in Madrid, Spain [39]. The scope of the project was to optimize the acquisition schedule for scientific opportunities announced yearly. It uses a local search algorithm that combines different techniques as hill-climbing, tabu-search and simulated annealing. It is based on maintaining a flexible consistent schedule for a revolution period, while trying to determine the amount of time that can be allocated to observation in that period. The TRF is, in other words, used to manage basic scheduling constraints and the search algorithm handles optimization criteria and more specific constraints.
The third use case was the XMM-Newton Mission APSI Scheduler (XMAS), supported by the XMM Newton long term science planning team. Due to similarities with the INTEGRAL scheduler, it has been implemented on the same set of tools [45].
The capabilities identified in the introduction of this chapter are all supported within the TRF.

5.1.2. Automated Scheduling/Planning ENvironment

ASPEN (Automated Scheduling/Planning ENvironment) ([27], [14]) is a modular, reconfigurable application framework (class library that provides the functionality of the components found in prototypical instances of a particular application domain) which is capable of supporting a wide variety of planning and scheduling applications.
ASPEN provides reusable software components such as:

- expressive constraint modeling language (to allow the definition of an application domain)
- constraint management system
- temporal reasoning for temporal constraints
- a GUI for plans/schedules (use in mixed initiative system, solving process is interactive human/machine)

In particular ASPEN is based on timeline representation capability, including states (finite and infinite), resources (depletable and non-depletable) and a constraint engine (e.g. simple temporal network, temporal constraint network, parameter constraint network).
Known applications are:

- Modified Antarctic Mapping Mission [43]
- Earth Observing One Mission [16], [17]
- Orbital Express Mission [18]

The corresponding embedded version, CASPER, for on-board use cases has been applied in the following missions:

- Earth Observing One Mission [41]
- 3CS Mission [15]

The ASPEN framework is composed by [27]:

- an Activity Database: central data structure in ASPEN is an activity. It can maintain hierarchical relationships between activities.
- a Temporal Constraint Network: it is a graph data structure that represents temporal constraints between activities and/or other activities and the scheduling horizon, it implements the Simple Temporal Problem [20]. The temporal constraint Network can be queried to assess is the network is consistent or not,

- a set of Resources Timelines: a Resource timeline is used to reason about the usage of physical resources by activities. Capacity conflicts detected functionality is present. Implementation of timelines for both depletable and non-depletable resources is considered,
- a set of State Timelines: a State Timeline represents arbitrary attributes, or states, that can change over time. State constraints can be checked in order to retrieve conflicts. Legal state transition should be modelled here.
- a Parameter Dependency Network: Each activity has a number of parameters that are either user-defined or computed by the system (start time, end time, duration, resources it uses, state it changes/uses,...). It is possible to create dependencies between pairs of parameters. These dependencies are represented and maintained in a Parameter Dependency Network.
- a set of Planning and Scheduling Algorithms: mainly divided into constructive and repair based. A constructive algorithm will incrementally construct a valid schedule, while a repair based algorithm will generate a schedule, not guaranteed to be valid, using random or greedy techniques, at every iteration eventual conflicts are repaired by selecting a repair method. Which conflict needs to be resolved first, and which repair method has to be applied is typically decided through the use of heuristics.

5.1.3. Extensible Universal Remote Operations Planning Architecture

Extensible Universal Remote Operations Planning Architecture (EUROPA) is a class library and tool set for building and analysing planners within a Constraint-based Temporal Planning paradigm [24]. EUROPA offers capabilities in 3 key areas of problem solving:

- Representation: EUROPA allows a rich representation for actions, states, resources and constraints that allows concise declarative descriptions of problem domains and powerful expressions of plan structure. This representation is supported with a high-level object-oriented modeling language for describing problem domains and data structures for instantiating and manipulating problem instances.
- Reasoning: Algorithms enforce domain rules and propagate consequences as updates are made to the problem state. These algorithms are based on logical inference and constraint-processing.
- Search: EUROPA provides a framework for integrating heuristics into a basic search algorithm and for developing new search algorithms.

EUROPA's main input modeling language is New Domain Definition Language (NDDL) (pronounced 'noodle'), a domain description language for constraint-based planning and scheduling problems. (state and activity description similar to PDDL [29], [22]).
An overview of the EUROPA framework is provided in [24]. Modern versions of EUROPA integrate the grounded and flexible time perspectives at a basic level [3].
EUROPA is composed by:

- a deepest layer, which offers fundamental representation, reasoning, modeling and search capabilities that can be used to create planning and scheduling applications. It is referred to as EUROPA Kernel. Its sub-components are:
  - a Constraint Engine: it handles consistency on constraints network. It provides a propagation architecture, to integrate specialized forms of local and global constraint propagation.
  - a Plan Database: it adds higher levels of abstractions for tokens and objects and the interactions between them.
  - a Rules Engine: it provides implementation of domain rules
  - a Solvers Module: it includes a component-based architecture for Flaw Identification, Resolution and heuristics
  - Model Interpreter: it provides the functionalities to execute other EUROPA kernel sub-components through a run-time interpreter.
- A second layer, containing important extension modules that have been found to be useful in dealing with real life domains and are therefore bundled with the EUROPA distribution:
  - Temporal Network Module: specialized algorithms and data structures to support efficient propagation of temporal constraints, detecting and maintaining temporal consistency.
  - Resources management module: specialized algorithms and data structures to support metric resources.
  - NDDL/ANML modules: parsers for NDDL and ANML. It defines the mapping from the language to the data structures provided by the Model Interpreter and so to all the modules in the kernel.
  - Client API: The client API exposes the functionality of the EUROPA kernel and built-in modules in a concise and safe manner so that client applications

can be built in C++ or Java.

5.2. Description Languages for Planning

Another key element while dealing with generalization of planning and scheduling solutions is the adoption or implementation of a Domain Definition Language. A Domain Definition Language is a formal language used to describe the planning domain and the problem instance in a standardized way, in order to make it readable (or parsable) for a variety of solvers. A quick history of the evolutions of such languages is reported in this section.

5.2.1. STRIPS

Precursor of this language family, STRIPS (STanford Research Institute Problem Solver) is the language used in the homonymous solver developed by the Stanford university in 1971. The language (and the solver) exploit concepts of classical planning. The planner assumes complete knowledge about initial states, deterministic outcome of actions and outputs a linear plan (intended as a simple sequence of actions). The language implements proposition logic and predicate. An action in STRIPS will have preconditions and effects, as shown in algorithm 5.1.

Algorithm 5.1 STRIPS action example

```
1: (: action move
2:    :parameters (?p - payer ?l1 - location ?l2 - location)
3:    :precondition (and (at ?p ?l1) (border ?l1 ?l2) (not (guarder ?l2)))
4:    :effect (and (at ?p ?l2) (not (at ?p ?l1)))
```

5.2.2. Action Description Language

Developed by an IBM researcher in 1987, ADL is considered an advancement of STRIPS. Effects can be conditional, indirect effects can be implemented in the world and non-deterministic and concurrent execution of actions are added to the representation capabilities. ADL also introduces negative literals and disjunction with respect to STRIPS.

5.2.3. Planning Domain Definition Language

PDDL has been initially developed by Drew McDermott and his colleagues in 1998 (inspired by STRIPS and ADL among others), mainly to be make possible the 1998/2000

International Planning Competition (IPC). After that, it evolved with every competition. Hereafter are described the main changes in feature between versions:

- PDDL1.2 separates planning in domain description and problem description. The domain description consisted of a domain-name definition, definition of requirements (model-elements which the PDDL-model is actually using), definition of object-type hierarchy, definition of constant objects, definition of predicates (templates for logical facts), and also the definition of possible actions (operator-schemas with parameters, which should be grounded/instantiated during execution). Actions had parameters (variables that may be instantiated with objects), preconditions and effects. The effects of actions could be also conditional (when-effects). The problem description consisted of a problem-name definition, the definition of the related domain-name, the definition of all the possible objects (atoms in the logical universe), initial conditions (the initial state of the planning environment, a conjunction of true/false facts), and the definition of goal-states (a logical expression over facts that should be true/false in a goal-state of the planning environment).
- PDDL2.1 (2002) introduces numeric fluent (i.e. modeling of non-binary resources), plan metrics (quantitative evaluation of a plan to allow optimization) and durative/continuous actions (which could have variable, non-discrete length, conditions and effects).
- PDDL2.2 (2004) introduces derived predicates (model dependency of given facts from other facts, e.g. A reachable from B, B reachable from C $\rightarrow$ A reachable from C) and timed initial literals (to model exogenous events occurring at given time independently from plan execution)
- PDDL3.0 (2006) introduces state trajectory constraints (hard-constraints in form of modal-logic expressions, which should be true for the state-trajectory produced during the execution of a plan, which is a solution of the given planning problem) and preferences (soft-constraints in form of logical expressions, similar to hard-constraints, but their satisfaction wasn't necessary, although it could be incorporated into the plan-metric) to enable preference-based planning.
- PDDL3.1 (2008 - present) introduces object fluents (fluents don't have to be numerical but also "object type") Successor, variant and extension of PDDL are:
- PDDL2.1 (2002-2006) extension of PDDL2.1. It provides the ability to model the interaction between the agent's behaviour and changes that are initiated by the agent's environment. Processes run over time and have a continuous effect on nu-

meric values. They are initiated and terminated either by the direct action of the agent or by events triggered in the environment. This 3-part structure is referred to as the start-process-stop model. Actions and events, which are instantaneous, are restricted to the expression of discrete change.

- NDDL (2002) - New Domain Definition Language is the NASA implementation of PDDL it uses a variable/value representation (timelines/activities) rather than a propositional/first-order logic, there is no concept of states or actions, only of intervals (activities) and constraints between those activities. In this respect, models in NDDL look more like schemas for SAT (satisfiability) encodings of planning problems rather than PDDL models. Because of the mentioned differences planning and execution of plans (e.g. during critical space missions) may be more robust when using NDDL, but the correspondence to standard planning-problem representations other than PDDL may be much less intuitive than in case of PDDL.
- MAPL (2003) - Multi-Agent Planning Language is an extension of PDDL2.1, presenting big modification: non-propositional state-variable are introduced temporal model given with modal operators (before, after, etc.) is introduced (PDDL3.0 introduces it too) actions whose duration will be determined in runtime are introduced explicit plan synchronization is introduced, realized through speech act based communication among agents (this assumption may be artificial, since agents executing concurrent plans shouldn't necessarily communicate to be able to function in a multi-agent environment) events (exogenous and endogenous) are introduced to handle concurrency of actions
- OPT (2003-2005) - Ontology with Polymorphic Types, big extension of PDDL2.1, is an attempt to create a general-purpose notation for creating ontologies, defined as formalized conceptual frameworks for planning domains about which planning applications are to reason. It introduces lambda-expressions allowing for efficient type-inference. It also introduces extension such as data-structures, non-Boolean fluents, return-values for actions, links between actions, hierarchical action expansion, hierarchy of domain definitions
- PPDDL (2004-2006) - Probabilistic PDDL extends PDDL2.1 with probabilistic effects (discrete, general probability distributions over possible effects of an action), reward fluents (for incrementing or decrementing the total reward of a plan in the effects of the actions), goal rewards (for rewarding a state-trajectory, which incorporates at least one goal-state), and goal-achieved fluents (which were true, if the state-trajectory incorporated at least one goal-state). Eventually these changes al-

lowed PPDDL1.0 to realize Markov Decision Process (MDP) planning, where there may be uncertainty in the state-transitions, but the environment is fully observable for the planner/agent.

- APPL (2006) - Abstract Plan Preparation Language is a variant of NDDL, which is more abstract than most existing planning languages such as PDDL or NDDL. The goal of this language was to simplify the formal analysis and specification of planning problems that are intended for safety-critical applications (e.g. power management or automated rendezvous in future manned spacecraft). APPL's expressive power is much less than PDDL's (in hope of staying robust and formally verifiable).
- RDDL (2011) - Relational Dynamic influence Diagram Language is based on PPDDL1.0 and PDDL3.0, but practically it is a completely different language both syntactically and semantically. The introduction of partial observability is one of the most important changes in RDDL compared to PPDDL1.0. It allows efficient description of Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs) by representing everything (state-fluents, observations, actions,...) with variables. This way RDDL departs from PDDL significantly. Grounded RDDL corresponds to Dynamic Bayesian Networks (DBNs) similarly to PPDDL1.0, but RDDL is more expressive than PPDDL1.0.
- MA-PDDL (2012) - Multi Agent PDDL is a minimalistic, modular extension of PDDL3.1 that allows planning by and for multiple agents. The addition is compatible with all the features of PDDL3.1 and addresses most of the issues of MAPL. It adds the possibility to distinguish between the possibly different actions of different agents (i.e. different capabilities). Similarly, different agents may have different goals and/or metrics. The preconditions of actions now may directly refer to concurrent actions (e.g. the actions of other agents) and thus actions with interacting effects can be represented in a general, flexible way. Moreover, a simple mechanism for the inheritance and polymorphism of actions, goals and metrics was also introduced in MA-PDDL. Since PDDL3.1 assumes that the environment is deterministic and fully observable, the same holds for MA-PDDL, i.e. every agent can access the value of every state fluent at every time-instant and observe every previously executed action of each agent, and also the concurrent actions of agents unambiguously determine the next state of the environment. This was improved later by the addition of partial-observability and probabilistic effects (in form of two new modular requirements, the latter being inspired by PPDDL1.0, and both being compatible with all the previous features of the language, including multi-agent).

- PDDL4J (2018): provide a Planning Domain Description library for Java
- ANML (Action Notation Modeling Language) as alternative to PDDL, NDDL (EUROPA modeling language) and AML (ASPEN modeling language). It:
  - uses strong notions of actions and states (like PDDL and AML)
  - uses a variable/value model (like NDDL and AML)
  - supports rich temporal constraints (like NDDL and AML)
  - provide simple, convenient idioms for expressing the most common forms of action conditions, effects and resource usage

  It supports both generative and Hierarchical Task Network (HTN) planning models in a uniform framework and has a clear, well-defined syntax.

6 | IT Technological Opportunities

In this chapter technological opportunities for Mission Planning application deployment in the IT field are reported, specially as a response to the evidenced needs of the New Space Economy.
Cloud computing and OS-virtualization will be introduced, followed by the description of microservice architectures (with advantages, disadvantages and design principles) and REST API as messaging layer.

6.1. Cloud Computing and OS-level virtualization

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power. It relies on the sharing of resources and "pay as you go" pricing model, which typically promote system scalability and reduce barriers to entry for new entities.
Maintenance of a cloud environment is easier because the data is hosted on an outside server maintained by a provider without the need to invest in data center hardware.
Another big advantage of cloud computing is parallelization of processes. Automatic planning processes usually have a big time complexity. Parallelization can be a tool to reduce time complexity without changing operational complexity. This means faster responses from the system without increasing overall resource usage.
Virtualization has been used for decades to deploy applications on servers. With the success of the last decade of Dockers [1], the virtualization shifted towards OS-level virtualization.
OS-level virtualization is an execution method which isolates hardware and software resources within a single OS. This is done through the creation of containers, multiple isolated user-space instances. The usage of such methods eliminate the needs of virtual machines and their guest OS.
While in traditional virtualization, a program executed in the OS can see every hardware and software resource available, a program executed on a virtualized OS can see only resources allocated to it.

The usage of containers implies a faster start of the application and a increased efficiency in terms of resource usage.

6.2. Microservices architecture

> *From a simplified point of view, one could argue that mission planning is based on a collection of tasks, resources, constraints and algorithms; and the system shall by its nature be easily adaptable to any new operational requirement. [...] However, reality is a bit more complex, as the mission planning system in many cases is embedded in a complex ground segment. The changes to be performed can range from the adaption of the mission planning model and algorithms, to the mission planning internal interfaces and workflows, over the complete revision of established ground segment internal workflows and last but not least to the adaption of external interfaces from and to the customer.* [35]

As generalized planning and planning-as-a-service emerges as needs for New Space missions, it is necessary to provide modularity in software components. Furthermore, scalability is required to help new private companies to grow from in-orbit demonstrator to constellation made of hundreds of spacecrafts. In this context, microservices could fit well.
A microservice architecture consists of a collection of small, autonomous services. Each service is self-contained and implements a specific business capability within a bounded context, where the latter is a certain area of business identified by a domain of interest.
As opposed to monolithic applications, Microservices are small, independent and loosely coupled. Each service has a separated codebase. This means that a small team of developers is enough to write and maintain a service.
Furthermore, microservices are deployed independently: a team can change an existing service without the risk of breaking the entire application.
Data persistence is handled at the microservice layer, thus no separate data layer is usually used.
As the communication of a microservice happens through a well defined API, internal implementation details of each service is hidden from other services.
Main advantages of a Microservices architecture are:

- Agility: independent deployment makes it easier to manage bug fixes and feature releases.
- Small, focused team: small team size promotes agility and avoids efficiency loss due

to context switch [47].

- Small code base: by not sharing code or data storage, each microservice is easy to maintain and be improved.
- Mix of technologies: each microservice can use the technology that best fits.
- Fault isolation: the unavailability of a single microservice won't disrupt the entire application if faults handling is properly implemented.
- Scalability: each microservice can be scaled independently. Resources are allocated only when needed, thus increasing efficiency.
- Data isolation: data updates involve only one service, while in monolithic applications different parts may use those data, thus making alteration risky.

However microservices advantages comes with some drawbacks:

- Complexity: each microservice is simpler, but the entire system as a whole is more complex than a monolithic application
- Development and Testing: handling dependencies between services can be challenging both during implementation and testing, specially when the application is evolving quickly.
- Lack of governance: the decentralized approach has advantage, but the amount of different technologies may become hard to maintain. Sometimes standards have to be defined, having care of not limiting teams flexibility too much. A classic example of a standardized technology is logging.
- Network congestion and latency: interservice communications can increase latency at the point it becomes a problem. Overly chatty APIs should be avoided and asynchronous communication should be implemented when possible.
- Data integrity: since each microservice is responsible for its own data persistence, consistency can be challenging.
- Management: in order to have a maintainable microservice architecture, a proper DevOps culture must be in place. An example is being able to correlate multiple service logging caused by a single user operation
- Versioning: updates to a microservice must not break other microservices that depend on it. Backward and forward compatibility should be always provided.

In microservices, the goal of improving software delivery speed as functional scope grows

is realized through greater agility, higher composability, improved comprehensibility, independent service deployability, organizational alignment and polyglotism [36]. This principle well applies to mission control software within the context of New Space, as the need of scalability and reliability have to cohabit with the need of reduced time to market.
While designing a microservice architecture, it could appear easy to focus on which services are needed. But this is not enough, as microservices need to be harmonized in order to provide value to the user. In order to design a microservice architecture, a model can be built considering five parts.

Service

Defining scope and granularity of the microservices is needed to design a functioning architecture. By implementing well-designed APIs, it will be possible to guide the user through complex systems.
Solution The solution represents a macro view of the architecture. When designing microservices, decisions are made to address the required functionalities of the specific services. The logic is linking input to output. While designing the solution, the design should orchestrate the inputs and outputs of the different services.

Process and Tools
Choosing the right process and tools is crucial for producing good overall system behavior. Adopting standardized processes like DevOps, Agile or tools like Docker containers promote changes in the system as needed.

Organization What an organization produces is often a mirror of which people are involved in the work and how they communicate. But organization design is highly context-dependent. A hundreds of employees company will handle its organization differently from a ten people start-up. A good microservice architecture needs a good understanding of which implications change in the organization can bring to the product.

Culture
Culture is the most intangible, but maybe the most important, aspect of a microservice architecture design. It is possible to define culture as the values, beliefs and ideals shared by the worker of a company or organization. Culture is important because it defines the atomic decision that people within the system will make. As for organization, it is context-sensitive.

Also if difficult to surveying, culture shapes how the people will do their work, thus shaping the nature of the system.

6.3. REST API

As already evidenced in chapter 2, a Message Abstraction Layer should be provided to interact with different subsystem of the MOS. In order to do so, REST API can be the right tool.
An API is a set of definitions and protocols for building and integrating application software. It's sometimes referred to as a contract between an information provider and an information user—establishing the content required from the consumer (the call) and the content required by the producer (the response). It is possible to think to an API as the mediator between the users and the resources or web services they want to get.

A REST API (also known as RESTful API) is an API that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.
It should be noted that since REST is a set of architectural constraints and not a protocol or a standard, a REST API can be implemented in a variety of ways.
When a user makes a request via a REST API, it transfers a representation of the state of the resource to the requester (or endpoint). Representation of data is delivered via HTTP in a variety of formats: JSON, HTML, XLT, Python, PHP or plain text. JSON is generally the most popular format since it's language agnostic and easily readable by a human.
The HTTP most common request types used in APIs are:

- GET: it is used to read/retrieve data from a web server. It is idempotent and safe (Idempotency means that multiple requests will have the same outcome, safety means that server state is not altered by the request).
- PUT: it is used to modify data already existing on the server. It is idempotent but not safe.
- POST: it is used to send data to the server. Usually it creates a new resource if it does not already exist. It is neither idempotent nor safe.
- DELETE: it is used to remove a resource from the server. It is idempotent but not safe.

It is often reported a parallelism between these four HTTP methods and CRUD methods (Create, Read, Update and Delete). While this can be a good example to learn request usage, they should not be confused. Furthermore, specially for PUT request, the corre-

spondence is not 1-to-1.
Other HTTP methods exist (HEAD, OPTIONS, TRACE, PATCH) but since they are not relevant to the implementation of a RESTful API, they are not covered in this list.

The criteria which make an API RESTful are:

- a client-server architecture made of clients, servers and resources, with request managed via HTTP;
- stateless client-server communication (no client information is stored between GET request and each request is separated and unconnected);
- cacheable data that streamlines client-server interactions.

Though the REST API has these criteria to conform to, it is still considered easier to use than a prescribed protocol like SOAP (Simple Object Access Protocol), which has specific requirements like XML messaging, and built-in security and transaction compliance that make it slower and heavier.

REST API can be categorized in 4 Level [50]:

- Level 0: using HTTP as a transport system for remote interactions;
- Level 1: introduction of Resources. A resource is identified by an URI (Unique Resources Identifier). It can be assimilated to the notion of object identity.
- Level 2: introduction of HTTP Verbs used in the way they are intended in the HTTP.
- Level 3: usage of hypermedia controls, often referred to as HATEOAS (Hypertext As The Engine Of Application State). In hypermedia controls the resource contains also the possible actions that can be executed, guiding the user in the fruition of the product exposed by the API.

7 | Conclusions

The Mission Planning System is part of a bigger and more complex system, the Mission Operations System. Due to the needs of New Space mission underlined in the introduction, its implementation should be able to scale as the mission size and the related commercial activity grow.

A Microservices Architecture guarantees the possibility to rapidly iterate on the designed system, as required in order to implement automation [23]. Furthermore, OS-level virtualization and deployment on cloud guarantee the scalability of the solution from a single spacecraft, as a technological demonstrator, to constellation of hundreds of spacecrafts.

As this needs are shared by a large number of different entities and missions, the implementation of a generalized application remarks the logic of maximizing the return of investment, in line with the New Space shifting paradigms described in Chapter 1.

As described in Chapter 4, the generalization of a planning system requires a shared vision of the application design, and this thesis has evidenced as enabler the implementation of a timeline-based planning system. Furthermore, the adoption of a standardized description languages among different research and industry entities could favour the spread of innovative planning algorithms. The author of this work recognise that the adoption of a unique description language is far from the today scenario, due to the little coupling of industry practice with research development and trend. However, due to the need of maximization of economic return, space community could find in the operation research field a rich source of improvements.

Generalized multi-mission planning requires the existence of a product flexible enough to satisfy the different needs of the different missions. The introduction of such a service could not only improve return of investments, but also reduce the time to market of newborn missions, thus providing an advantage with respect to competitors.

Bibliography

[1] Docker official website. URL https://www.docker.com/.

[2] J. ALLEN. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26: 510–521, 01 2013. doi: 10.1016/B978-1-4832-1447-4.50033-X.

[3] J. Barreiro, M. E. Boyce, M. B. Do, J. D. Frank, M. Iatauro, T. Kichkaylo, P. H. Morris, J. C. Ong, E. Remolina, T. B. Smith, and D. E. Smith. Europa: A platform for ai planning, scheduling, constraint programming, and optimization. 2012.

[4] M. K. Ben-Larbi, K. Flores Pozo, T. Haylok, M. Choi, B. Grzesik, A. Haas, D. Krupke, H. Konstanski, V. Schaus, S. P. Fekete, C. Schurig, and E. Stoll. Towards the automated operations of large distributed satellite systems. part 1: Review and paradigm shifts. *Advances in Space Research*, 67(11):3598–3619, 2021. ISSN 0273-1177. doi: https://doi.org/10.1016/j.asr.2020.08.009. URL https://www.sciencedirect.com/science/article/pii/S0273117720305676. Satellite Constellations and Formation Flying.

[5] E. Bensana, G. (y, J. Agnese, N. Bataille, and D. Blumstein. Exact and inexact methods for the daily management of an earth observation satellite. *European Space Agency, (Special Publication) ESA SP*, 03 1999.

[6] J. Bidot, T. Vidal, P. Laborie, and J. Beck. A general framework for scheduling in a stochastic environment. pages 56–61, 01 2007.

[7] D. Bindschadler, C. Boyles, C. Carrion, and C. Delp. Mos 2.0: The next generation in mission operations systems. 04 2010. ISBN 978-1-62410-164-9. doi: 10.2514/6.2010-1953.

[8] S. Calderoni. The increasingly important role of european universities in newspace, 2020. URL https://newspaceglobal.com/node-705/.

[9] CCSDS 529.0-G-1. Mission Planning and Scheduling, Informational Report, Issue 1. Standard, Consultative Committee for Space Data Systems, Washington, DC, USA, June 2018.

[10] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi. Mrspock: Generating a planning system through a timeline representation framework. 01 2008.

[11] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi. Developing an end-to-end planning application from a timeline representation framework. In *IAAI*, 2009.

[12] P. Chaizy, T. Dimbylow, M. Hapgood, and P. Allan. Plan management system for space science mission systems. *Advances in Space Research*, 44(1):1–22, 2009. ISSN 0273-1177. doi: https://doi.org/10.1016/j.asr.2009.03.022. URL `https://www.sciencedirect.com/science/article/pii/S0273117709002099`.

[13] S. Chien. *A generalized timeline representation, services, and interface for automating space mission operations.* doi: 10.2514/6.2012-1275459. URL `https://arc.aiaa.org/doi/abs/10.2514/6.2012-1275459`.

[14] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran. Aspen - automated planning and scheduling for space mission operations. 12 2000.

[15] S. Chien, B. Engelhardt, R. Knight, G. Rabideau, R. Sherwood, E. Hansen, A. Ortiviz, C. Lohr, and S. Wichman. Onboard autonomy on the three corner sat mission. 01 2001. doi: 10.2514/6.2002-T3-39.

[16] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davis, D. Mandl, S. Frye, B. Trout, S. Shulman, and D. Boyer. Using autonomy flight software to improve science return on earth observing one. *Journal of Aerospace Computing, Information, and Communication*, 2(4):196–216, 2005. doi: 10.2514/1.12923. URL `https://doi.org/10.2514/1.12923`.

[17] S. Chien, D. Tran, G. Rabideau, S. Schaffer, D. Mandl, and S. Frye. Improving the operations of eo-1 via automated mission planning. 04 2010. ISBN 978-1-62410-164-9. doi: 10.2514/6.2010-2199.

[18] C. Chouinard, R. Knight, G. Jones, and D. Tran. Orbital express mission operations planning and resource management using aspen. 05 2008. doi: 10.1117/12.782454.

[19] S. Chung. *Timeline-based Mission Operations Architecture.* doi: 10.2514/6.2012-1269750. URL `https://arc.aiaa.org/doi/abs/10.2514/6.2012-1269750`.

[20] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1):61–95, 1991. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(91)90006-6. URL `https://www.sciencedirect.com/science/article/pii/0004370291900066`.

[21] B. Du, S. Li, Y. She, W. Li, H. Liao, and H. Wang. Area targets observation mission planning of agile satellite considering the drift angle constraint. *Journal of Astronomical Telescopes, Instruments, and Systems*, 4(4):047002, 2018.

[22] S. Edelkamp and J. Hoffmann. The deterministic part of IPC-4: an overview. *CoRR*, abs/1109.5663, 2011. URL `http://arxiv.org/abs/1109.5663`.

[23] P. Ferri, M. Pecchioli, and A. Pena. Introducing automation in ground segment operations. 05 2004. doi: 10.2514/6.2004-247-97.

[24] J. Frank and A. Jonsson. Constraint-based attribute and interval planning. *Constraints*, 8:339–364, 10 2003. doi: 10.1023/A:1025842019552.

[25] S. Fratini and A. Cesta. The apsi framework: A platform for timeline synthesis. 06 2012.

[26] S. Fratini, T. Nogueira, and N. Policella. Integrating modeling and knowledge representation for combined task, resource and path planning in robotics. 06 2017.

[27] A. Fukunaga, G. Rabideau, S. Chien, and D. Yan. Towards an application framework for automated planning and scheduling. In *1997 IEEE Aerospace Conference*, volume 1, pages 375–386 vol.1, 1997. doi: 10.1109/AERO.1997.574426.

[28] V. Gabrel and D. Vanderpooten. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. *European Journal of Operational Research*, 139:533–542, 06 2002. doi: 10.1016/S0377-2217(01)00188-6.

[29] A. E. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5):619–668, 2009. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2008.10.012. URL `https://www.sciencedirect.com/science/article/pii/S0004370208001847`. Advances in Automated Plan Generation.

[30] A. Globus, J. Crawford, J. Lohn, and R. Morris. Scheduling earth observing fleets using evolutionary algorithms: Problem description and approach. 12 2003.

[31] A. Golkar and A. Salado. Definition of new space—expert survey results and key technology trends. *IEEE Journal on Miniaturization for Air and Space Systems*, 2(1):2–9, 2021. doi: 10.1109/JMASS.2020.3045851.

[32] N. G. Hall and M. J. Magazine. Maximizing the value of a space mission. *European*

Journal of Operational Research, 78(2):224–241, 1994. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(94)90385-9. URL `https://www.sciencedirect.com/science/article/pii/0377221794903859`. Project Management and Scheduling.

[33] D. Kaslow. Planning and scheduling of earth observing satellites. pages 1 – 12, 04 2007. doi: 10.1109/AERO.2007.352958.

[34] D. Krupke, V. Schaus, A. Haas, M. Perk, J. Dippel, B. Grzesik, M. K. B. Larbi, E. Stoll, T. Haylock, H. Konstanski, K. F. Pozo, M. Choi, C. Schurig, and S. P. Fekete. Automated data retrieval from large-scale distributed satellite systems. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1789–1795, 2019. doi: 10.1109/COASE.2019.8843045.

[35] F. Mrowka, T. Göttfert, M. Wörle, B. Schättler, and F. Stathopoulos. The terrasar-x/tandem-x mission planning system: Realizing new customer visions by applying new upgrade strategies. 05 2016. doi: 10.2514/6.2016-2597.

[36] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen. *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media, Inc., 1st edition, 2016. ISBN 1491956259.

[37] T. Nogueira, S. Fratini, and K. Schilling. Autonomously controlling flexible timelines: From domain-independent planning to robust execution. In *2017 IEEE Aerospace Conference*, pages 1–15, 2017. doi: 10.1109/AERO.2017.7943603.

[38] S. Peng, H. Chen, J. Li, and N. Jing. A heuristic method for single satellite observation and transmission tasks planning. In *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 2925–2932, 2017. doi: 10.1109/FSKD.2017.8393247.

[39] C. Pralet and G. Verfaillie. Aims: A tool for long-term planning of the esa integral mission. 2009.

[40] G. Rabideau, R. Knight, S. Chien, A. Fukunaga, and A. Govindjee. Iterative repair planning for spacecraft operations in the aspen system. In *International Symposium on Artificial Intelligence Robotics and Automation in Space (ISAIRAS 1999)*, Noordwijk, The Netherlands, June 1999. URL `https://ai.jpl.nasa.gov/public/papers/search-isairas99.ps`.

[41] G. Rabideau, D. Tran, S. Chien, B. Cichy, R. Sherwood, D. Mandl, S. Frye, S. Shulman, J. Szwaczkowski, D. Boyer, and J. Van Gaasbeck. Mission operations of earth observing-1 with onboard autonomy. In *2nd IEEE International Conference on Space*

Mission Challenges for Information Technology (SMC-IT'06), pages 7 pp.–373, 2006. doi: 10.1109/SMC-IT.2006.48.

[42] K. Reinholtz. *Timeline as Unifying Concept for Spacecraft Operations*. doi: 10.2514/6.2012-1274906. URL https://arc.aiaa.org/doi/abs/10.2514/6.2012-1274906.

[43] B. D. Smith, B. E. Engelhardt, and D. H. Mutz. The radarsat-mamm automated mission planner. *AI Magazine*, 23(2):25, Jun. 2002. doi: 10.1609/aimag.v23i2.1638. URL https://ojs.aaai.org/index.php/aimagazine/article/view/1638.

[44] D. Smith, J. Frank, and W. Cushing. The anml language. 09 2008.

[45] R. Steel, M. Niezette, A. Cesta, S. Fratini, A. Oddi, G. Cortellessa, R. Rasconi, G. Verfaillie, C. Pralet, M. Lavagna, A. Brambilla, F. Castellini, A. Donati, and N. Policella. Advanced Planning and Scheduling Initiative- MrSpock Aims for Xmas. In H. Lacoste, editor, *ESA Special Publication*, volume 673 of *ESA Special Publication*, page 8, Sept. 2009.

[46] C. P. S. The CubeSat Program. Cubesat design specification rev. 14.1. https://www.cubesat.org/s/CDS-REV14_1-2022-02-09.pdf, 2 2022.

[47] A. Tregubov, B. Boehm, N. Rodchenko, and J. Lane. Impact of task switching and work interruptions on software development processes. pages 134–138, 07 2017. doi: 10.1145/3084100.3084116.

[48] M. Vasquez and J.-K. Hao. A "logic-constrained" knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. *Computational optimization and applications*, 20(2):137–157, 2001.

[49] T. Vidal and H. Fargier. Handling contingency in temporal constraint networks: From consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, 11(1):23 – 45, 1999. doi: 10.1080/095281399146607. URL https://www.scopus.com/inward/record.uri?eid=2-s2.00442314380&doi=10.1080%2f095281399146607&partnerID=40&md5=482115e5ae85059e0f2e0dc7467e46f0. Cited by: 193.

[50] J. Webber, S. Parastatidis, and I. Robinson. *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly, Beijing, 2010. ISBN 978-0-596-80582-1. URL https://www.safaribooksonline.com/library/view/rest-in-practice/9781449383312/.

[51] W. J. Wolfe and S. E. Sorensen. Three scheduling algorithms applied to the earth ob-

serving systems domain. *Management Science*, 46(1):148–166, 2000. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2634914.

[52] L. Xing and Y.-W. Chen. Mission planning of satellite ground station system based on the hybrid ant colony optimization. *Acta Automatica Sinica*, 34, 04 2008. doi: 10.3724/SP.J.1004.2008.00414.

[53] G. Zhang, X. Li, G. Hu, Z. Zhang, J. An, and W. Man. Mission planning issues of imaging satellites: Summary, discussion, and prospects. *International Journal of Aerospace Engineering*, 2021, 2021.

List of Figures

List of Tables

Acknowledgements

I would like to acknowledge Leanspace SAS for hosting the research period that motivated this thesis. The guidance of my supervisors Ezequiel Gonzalez and Stan Kaethler and the support of my colleagues have made this achievement possible.
To my family and my friends goes a huge thank you, for walking beside me on this path.