Implementing a Multi-Agent Planning System for Martian Exploration

Laia Barcenilla Mañá[1], Núria Cardona Vilar[1], Natalia Muñoz Moruno[1] and Helena Sánchez Ulloa[1]

[1] *Master in Artificial Intelligence, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.*

I. Introduction

The exploration of the Martian terrain poses significant challenges that overcome traditional control systems. In this context, multi-agent systems emerge as a robust solution enabling multiple autonomous entities to cooperate and collaborate to reach complex goals which could be impossible for a single agent.

The primary goal of the project is the design and implementation of a multi-agent system able to orchestrate a simulated exploration on Mars. Using the framework of CrewAI [1], the system coordinates several specialized crews for the generation of an integral operative plan that respects the strict constraints.

The **scientific goals** have been organized according to their operational priorities into three categories:

High Priority

- Measure radiation levels in sandy terrain at node N12, N86 and N1.
- Collect subsurface samples from rocky terrain near nodes N70, N102, and N152.
- Deploy seismic sensors at the rocky node N20.

Medium Priority

- Air dust sample collection during flight over N33 crater.
- Detect organic molecules in icy terrain at N59.
- Map CO_2 frost coverage in icy areas N53, N63 and N108.

Low Priority

- Capture panoramic images of crater terrain at nodes N5, N58, N121 and N150.
- Identification of thermal anomalies at icy nodes N56 and N112.

Moreover, the complexity of the planning resides in the limiting operations of the physical agents. The system must be able to deal with some **operational constraints** which have been defined as detailed below.

1. Rovers must enter a heat shelter during 20 minutes if the surface temperature is below -60°C.
2. Rovers cannot operate if node temperature is below -80°C.
3. Rover's energy consumption is different depending on the type of terrain: 5% in rocky, increased 10% in sandy, increased 20% in icy and increased 15% in crater terrains.
4. Rovers may not operate in terrain classified as radioactive.
5. Rovers may not operate in terrain classified as unstable.
6. Rovers and drones should recharge if energy drops below 30%.
7. Drones must return to base after 25 minutes of flight.
8. Drones are disabled if there is a dust storm.
9. Drones cannot operate if wind gusts are higher than 40 km/h.
10. Drone's energy consumption is increased 15% if wind is higher than 30 km/h.
11. Satellites must maintain communication with the base station at N30 or N84 every 5 hours if they have to identify thermal anomalies.
12. Satellites cannot communicate in nodes with communication loss.

Finally, some **known hazards** have also been specified in some nodes, which may happen or not (set them to true or false). The established hazards have been outlined below:

1. Nodes N4, N19, N128: unstable rocky terrain.
2. Nodes N51, N78, N118: radioactive terrain.
3. Node N33: frequent dust storms.

II. Methodology

a. Environment analysis

The Martian environment consists of a terrain map represented as a graph. The connectivity of the different terrains is determined by the edges. Accordingly, agents are only able to move between adjacent areas.
The environment can present different properties depending on the context. These properties have been described in earlier research [2, 3].

First of all, physical agents conducting the mission will only have partial information about the terrain. They will mainly rely on specific sensors, for instance thermal, to detect samples and obstacles, and an entire view of the planet state will not be available. However, the mission planning is done considering the full graph.

As a consequence, in the planning phase the environment is accessible and fully observable. This categorization is based on the assumption that the provided graph is a complete and updated representation, fully resembling the real conditions on Mars.

As the environment is presented as a graph, there is a fixed number of possible inputs and outputs that agents can find or produce. It is mainly limited by the graph nodes and edges, and the effects of the actions made by the other agents in the system. For instance, an agent moves between a finite set of possible nodes instead of continuous coordinates. As a consequence, even though the real world is known to be continuous, the model of the environment used in the task can be defined as discrete.

The environment has no uncertainty because all potential hazards, if considered, are defined from the start and assumed to happen (or not happen) with total probability as are then set to True or False. Accordingly, in case there are hazards all agents are aware of them from the beginning, and therefore the environment can be classified as deterministic. Nevertheless, a non-deterministic version could be easily achieved by introducing a random probability of encountering the described hazards. In fact, this extension was implemented during the development of this project, however discarded later on because this added layer of uncertainty caused unstable planning behavior on the agents, specially regarding the validation phase.

In the defined environment, all actions have an effect on the future. For instance, a rover's decision to move from a certain node to another implies to update its location and the consumption of battery, restricting its future ability to reach other areas and returning to a base. Moreover, there is no obvious notion of independent episodes. That is because, in case a new route is planned from scratch, the environment would not be reset, and it would be interesting to remember the previous performance of the agents. Because of this, the environment is considered to be non-episodic.

Finally, the state of the terrain fully depends on agents. This means that the environment is static and remains unchanged unless an agent performs an action that modifies it.

b. Graph definition

The exploration environment is modeled as an undirected graph $G = (V, E)$, where V represents a set of locations (nodes) on the Martian surface and E represents the navigable paths (edges) between them.
Each node $v \in V$ has a certain amount of attributes that dictate the operational limits of the agents:

- **Terrain type:** Nodes are categorized as sandy, rocky, plain, sandy, icy or crater. This affects the movement cost (energy required) of rovers as well as determines if specific scientific tasks can be performed. In this case attribute type is a string that can take the values `{crater, sandy, rocky, plain, icy}`.
- **Is base:** $N30$ and $N84$ are assigned as base stations for refueling and data transmission. In this case attribute type is a boolean so can take the values `{true, false}`.
- **Communication loss:** Isolated nodes suffer from a communication loss. Those nodes are $N150$, $N151$, $N152$, $N153$ and $N154$. In this case attribute type is a boolean so can take the values `{true, false}`.
- **Dangerous places:** Some nodes were defined as too dangerous to operate there (there are hazards there), so that no agent may enter these coordinates.
  - **Radioactive:** Nodes $N51$, $N78$ and $N118$ are defined as radioactive. In this case attribute type is a boolean so can take the values $\{true, false\}$.
  - **Unstable:** Nodes $N4$, $N19$ and $N128$ are defined as unstable. In this case attribute type is a boolean so can take the values $\{true, false\}$.
  - **Dust storms:** Node $N33$ is defined as having dust storms. In this case attribute type is a boolean so can take the values $\{true, false\}$
- **Atmospheric conditions (wind & temperature):**
  - **Wind:** following Mars climate patterns, wind speeds normally range from 10 to 30 km/h in the different nodes. Drone's energy consumption is increased 15% if wind is higher than 30 km/h (nodes $N10$, $N34$, $N35$, $N74$, $N130$, $N145$) and drones cannot operate if wind gusts are higher than 40 km/h (nodes $N8, N50, N73$).
  - **Temperature:** temperatures in Mars vary from 20°C to -153°C [4], for this reason the different temperatures have been defined within this range. The type of terrain has also been taken into account when defining the temperatures, with icy regions being colder than sandy ones as a general rule. Temperature affects on rovers as they cannot operate if node temperature is below -80°C (nodes $N0, N56$) and they must enter a heat shelter during 20 min if the surface temperature is below -60°C (nodes $N1, N44, N55$).

Finally, the edges $e \in E$ contain a length attribute representing the physical distance between nodes. This value is used by the agents planning algorithms to calculate the most optimal paths. Figure 1 shows schematically the aspect of the final graph, with different color nodes representing the different types of terrain.

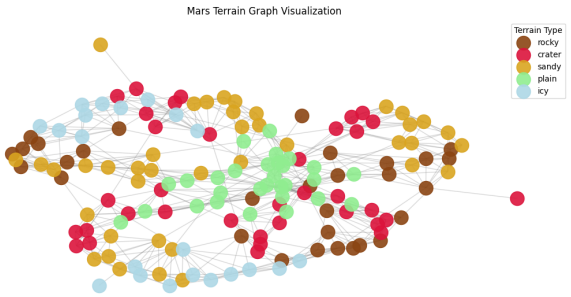

Figure 1: Mars terrain graph visualization.

c. Agents

1. Selection

According to the goals, several crews have been defined, each accomplishing its specific tasks. Below, each crew is described with the corresponding agents. Figure 2 presents an overview of the multi-agent system.

Mission Crew: receives the mission report and extracts the relevant structure.

- **Planner agent:** acts as the entry point for the system by receiving raw mission reports and feeding them directly into the `PlannerDivideTool` tool. Its primary role is to ensure data integrity.
- **Priority agent:** responsible for extracting scientific goals and categorizing them by type (rover, drone or satellite) using strict keyword matching. It maps each goal to its corresponding mission priority level to create a structured task list.
- **Hazard agent:** *(on the previous delivery there was a hazard and weather agent, when implementing the agent system we observed that only one agent was required for this task as weather are also part of the hazards and operational constraints)* it ensures that safety limitations are correctly assigned to the relevant assets (rover, drone or satellite).
- **Aggregator agent:** serves as the final integration layer by merging the outputs from the priority and hazard agents into a single, unified command structure.

Rover crew: plans and coordinates rover operations.
On the previous delivery only rover route planner agent and sample collector agent were defined. Now an extractor agent has been added in order to extract the important nodes so that the route planner agent had an easier to execute task.

- **Extractor agent:** specialized agent that identifies target node IDs from mission reports.
- **Route planner agent:** uses the `Multi_Rover_Route_Calculator` tool to compute different routes taking into account the hazards and operational constraints of the rovers and it takes the calculated routes and converts them into specific asset assignments
- **Sample collector agent:** Provides detailed instructions on how each sample should be collected, including specifications such as the sensors the physical agent performing the collection must use, the type of sample, and the collection procedure.

Drone crew: plans the drone flights and the collection of atmospheric samples.
On the previous delivery only drone flight planner agent and sample collector agent were defined. Now, similarly to rovers, an extractor agent has been added in order to facilitate the planner agent task.

- **Extractor agent:** specialized agent that identifies target node IDs from mission reports.
- **Flight planner agent:** ensures that flight paths are calculated by the tool `Batch_Flight_Calculator`.
- **Drone sampling agent:** Coordinates the sampling collection for drones, such as atmospheric samples or CO_2 coverage samples.

Satellite crew: integrates panoramic images and thermal anomalies at icy nodes.
On the previous delivery we only had the orbit planner agent and satellite planner, while developing the system we added communication loss extractor agent to process more easily the whole graph and extractor agent to produce target nodes requiring orbital imaging.

- **Communication loss extractor agent:** processes GraphML environment data to identify "communication loss zones." It maps restricted signal nodes to ensure that the mission planner avoids positioning satellites in areas where they cannot communicate with the base.
- **Extractor agent** responsible for filtering the mission priority report specifically for satellite tasks. Produce a Python list of target nodes requiring orbital imaging.
- **Satellite planner agent:** matches satellites to target nodes using a one-to-one assignment logic. It balances physical orbital locations against observation goals, ensuring all assets stay within communication windows and avoid hazardous "loss of signal" nodes.
- **Image capture agent:** converts mission goals into 5-step execution protocols. It makes instructions (including sensor calibration and timing) based on the specific imaging modality (thermal vs. panoramic) and the target terrain.

Validation crew: responsible of validating the output of the rover, drones and satellite crew (see if the different required nodes are correctly visited, the constraints are followed, hazards avoided...).
On the previous delivery this was defined as only one agent as part of the integration crew. As we go backward when an output is not good enough we did not want to integrate everything until we had the final correct output. For this reason this new crew has been created, to facilitate the flow of the whole system.

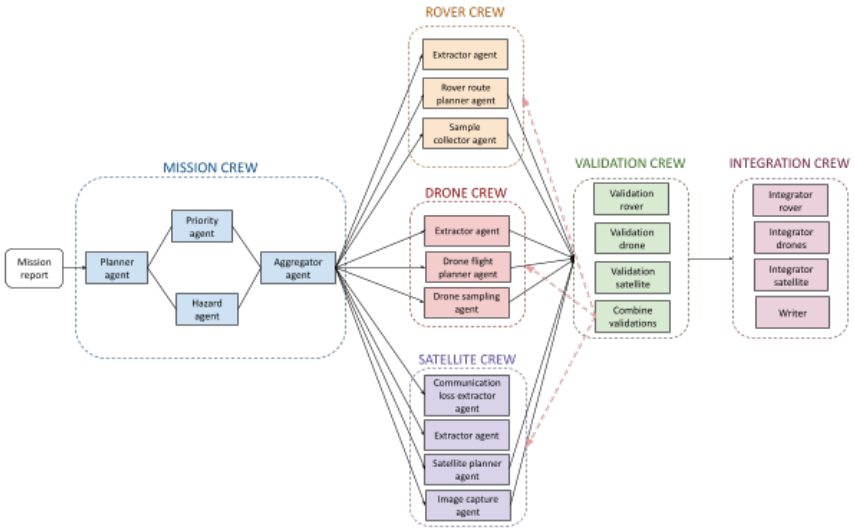

Figure 2: Overview of the multi-agent system.

- **Validation rover:** performs a strict binary check to ensure 100% of required rover mission nodes are present in the generated routes. Also outputs feedback e.g. Mission failed because N86 was not visited.
- **Validation drones:** performs a strict binary check to ensure 100% of required drones mission nodes are present in the generated routes.
- **Validation satellite:** cross-references imaging goals with satellite trajectories while strictly enforcing specific communication time-windows for thermal anomaly detection.
- **Combine validations:** synthesizes the individual feedback from the rover, drone, and satellite validators into a final, standardized "True/False" status json.

Integration crew: integrates different plans from the rover, drone and satellite crews into the final mission.
On the previous delivery only one agent was defined for that task. Now, we have more than one agent as a lot of outputs were obtained from the different crews and splitting the task into different specialized agents led to better results.

- **Integrator rover:** responsible for merging raw ground navigation trajectories with scientific sampling protocols from rovers.
- **Integrator drones:** responsible for merging raw aerial navigation trajectories with scientific sampling protocols from drones.
- **Integrator satellite:** responsible for merging raw orbital satellite positions with scientific sampling protocols from satellites.
- **Writer:** consolidates the specialized reports from the rover, drones and satellite integrators into a single, comprehensive mission report (final_mission_report.md).

2. Tasks and Tools

This section describes the tasks that the multi-agent system has been designed to conduct, together with the required tools to carry out each of them. Each task represents a precise objective in the Martian exploration framework, while the tools include algorithms that enable agents to perform an efficient and coordinated execution. All of this together provides a clear relationship between the objectives of the system and the resources employed to reach them. It is important to mention that, for the structure of the project, the tools have been saved in a specific directory together with the crew, to ensure easier management.

Mission Crew

Additional Information

This task, conducted by the **planner agent**, represents the input of the system. This task is carried out through the use of a tool called `Planner_Divide_Report`, whose objective is to divide the input markdown report into a structured format ready for processing. It divides the initial report into sections such as: scientific goals, operational constraints, mission priorities, and known hazards. The tool divides the text and transforms it into a Python list. Moreover, it also divides the mission priorities into a dictionary with the categories 'high', 'medium', and 'low'. The output of this task enables easier management of the requirements in further steps of the analysis.

Reporting Priority

This task which can be executed in an asynchronous manner to optimize the computational effort is conducted by the **priority agent**. Given the context extracted in the *Additional Information* task, it extracts the specific goals for each crew, returning a dictionary containing as key each crew. As a value, it returns another dictionary containing the priorities of the goals of each task, for instance containing for 'rover' the 'high', 'medium', and 'low' priority goals.

Reporting Hazard

The reporting hazards task is in charge of extracting and

isolating the operational constraints as well as the hazards affecting each individual crew. This task, executed by the **hazard agent**, similarly to the *Reporting Priority* task, can also be executed asynchronously. It returns a dictionary containing as keys each crew. Inside, each of the values corresponds to another dictionary that contains the hazards and operational constraints that affect the specific crew.

Reporting Aggregation
The last task of the mission crew corresponds to the aggregation of the results, which uses the context coming from the *Reporting Priority* and *Reporting Hazards* reports. This task is conducted by the **aggregator agent** and aims to generate a report in JSON format containing for each crew the goals scaled by priority, operational constraints, and hazards, generating a clear and structured report containing all the information provided by the previous agent.

Rover Crew
Final Nodes
This task is conducted by the **extractor agent**. It acts as a bridge between the mission crew and the rovers' operations. This agent is in charge of processing the priority report provided by the *Reporting Aggregation* task from the mission crew and isolate the nodes that require intervention. This task provides a Python list containing the node IDs where rovers need to perform their tasks.

Reporting Route
Subsequently, this task carried out by the **route planner agent** computes the routes for all the input rovers using a tool called `Multi_Rover_Route_Calculator`. For each of the provided rovers it computes the route from its initial position to each target node and back to the original position, computing a circular path. This tool takes into account the defined constraints such as the battery, the specified increased energy consumption depending on the terrain. It restricts terrains with the specified hazards (either true or false) considering unstable nodes, high radiation, extreme temperatures. It returns a dictionary having each rover as key, and the path that it should follow, together with further details of the route. Posteriorly, an optimal assignment of each rover to a route is conducted. The system assigns the objective nodes to rovers in an optimized way using the result obtained by the *Final Nodes* task. For each rover it filters the feasible routes and ensures that each rover receives at least one node if possible. For the assignment rover-node it prioritizes the smaller instance and in case of tie, it will be assigned to the rover with higher remaining energy. As a result it returns a dictionary containing each rover and the routes that they will conduct.

Reporting Sampling
The last task of this crew is conducted by the **sample collector agent** and conducted in an asynchronous manner. It aims to provide a document specifying the procedures standardized to ten steps for the different samplings and the sensor deployment.

Drone Crew
Final Nodes
Similar to the *Final Nodes* task from the rover crew, the **extractor agent** from the drone crew is in charge of processing the priority report from the *Reporting Aggregation* task. The goal of this task is to extract a list containing the nodes where the physical drones would need to act.

Reporting Route
After the extraction of the relevant nodes, this task computed by the **flight planner agent** computes the optimal routes for all the input drones. It relies on the use of the `Batch_Flight_Calculator` tool, which computes the flight from the initial location to one of the specified nodes (previously extracted) for multiple drones. From some information regarding the drones such as their location and their energy, for each objective node the tool assigns the closest drone to the destination and generates the complete route. This tool takes into account the operational constraints such as the maximum flight time, the wind's speed, and sand storms. Similarly to the situation before, the hazards can be activated or deactivated. For each one of the computed routes it checks the remaining battery of the drones. Finally it returns a JSON containing the valid routes for each input drone.

Generate sampling
Simultaneously the **sample collector agent** analyzes the goals for the drones and returns a detailed report containing the procedure of the different samplings to be conducted by the drone crew.

Satellite Crew
Communication Loss Extractor
This task is conducted by the **communication loss extractor agent**. It uses the `Communication_Loss_Detector` tool to scan the integrity of the graph communications. It identifies nodes where there is a loss of communication, a restriction that acts as a hard constraint for the satellites. This task returns a list containing the specific nodes where satellites cannot communicate.

Extractor
In parallel, similar to the *Final Nodes* task from the rover and drones crew, this **extractor agent** is responsible for extracting the nodes from the *Reporting Aggregation* report where satellites need to perform their tasks. It returns a Python list containing the node IDs where satellites act.

Planner
This task, executed by the **planner agent**, is responsible for assigning the target nodes to satellites in a conflict free manner. Given the report provided by *Reporting Hazard*, the input satellites, the nodes to be covered given by the *Extractor*, and the nodes to be avoided extracted by the *Communication Loss Extractor*, for each node it assigns it to one satellite and enforces all constraints to be taken into account. Moreover, it ensures a strict one-to-one assignment, meaning that each target node must be assigned to only one satellite. Moreover, it assigns a numeric value indicating when the satellite must communicate with base. As output, this task returns a dictionary containing the satellite, the location, and the target node, as well as the communication window.

Image Capture
Moreover, the **image capture agent** generates the required technical documentation for the operation. For each mission type (panoramic capture or thermal identification) it produces a five-step instruction set covering the procedure for each task.

Validation Crew
Validation Rover/Drone/Satellite
To guarantee the integrity of all levels, the system executes three validation tasks: *Validation Rovers*, *Validation Drones*, and *Validation Satellites*. They use as input the *Reporting Aggregation* report together with the route provided by each specific crew to compare if the requirements of the mission (goals) are exactly what the agents have computed. These tasks are conducted by the following agents: **validation rover agent**, **validation drone agent** and **validation satellite agent**.

While for rovers and drones it is centered on verifying that there is not any node missing to visit, for satellites it performs a more strict verification, checking if the satellite points to the correct object and computes the window communication times.

For each of them it returns a boolean verdict, that will be true if the mission has been completed with all nodes visited and without hitting hazards and constraints, and false otherwise. Moreover, in case of false, it also includes a feedback explanation that will specify the cause of failure.

Combine Validations
This task is executed by the **combine validations agent** and synthesizes the partial results achieved by the *Validation Rovers*, the *Validation Drones* and the *Validation Satellites* into a single JSON report that stores the verdicts of each crew.

Integration Crew
Integrator Rover/Drone/Satellite
The **integrator rover agent**, **integrator drone agent**, and **integrator satellite agent** are responsible for processing the routes for each individual crew, containing the coordinates, times, and other specifications, and the details on how each sampling should be performed. Taking all this information, the agent takes the spatial information with the operation. For instance, if a rover visits N70, the agent inserts in the report the procedures corresponding to that place. In a similar way, it ensures that the communication windows for satellites coincide with the protocols. For each of them, it provides a technical report detailing the logistics (the trajectory for drones and rovers and the location for satellites), together with an integrated scientific protocol and the required equipment together with the mission objectives.
Writer
Finally, the **writer agent** consists of the last task of the multi-agent system. This agent aggregates the three reports coming from the *Integrator Rover*, *Integrator Drone*, and *Integrator Satellite* tasks. In this task, the technical content is not altered, but restructured in a unique document called *final_mission_report.md*. The objective of this task is to deliver a final coherent plan where information of the rovers, drones, and satellites is presented integrally and in a structured manner.

3. Taxonomy

For this section the content from the previous delivery has been used, and new agents characteristics have been added.
In this section an explanation of the different agent properties and types can be found, different properties and types have been defined following the definitions from [5, 6, 7].

All agents in our multi agent system are going to be **autonomous**. This is because the system's whole purpose is to generate plans, and the executor agents (whose role would be only to follow given plans and do not require autonomy) will not be present. Therefore, it is essential for our agents to have full control over their own actions and internal state in order to create and manage the different plans. Regarding **mobility**, any of our agents will have this property as all of them will be executed on our local computers. Also, all of them should be **benevolent** and **truthful** as the information provided by them has to be correct. For this reason, when it comes to **character** they will have to be trustable and honest although those are subjective opinions. No **emotions** are required to be exhibited by them, as they are just doing a planning to explore Mars.

The agents will **not** be **flexible** and nor **reactive**, since they will not adapt to changes in the environment, as it is static. This means, that each of the agents should achieve its goals with the information provided, without recomputing the plans in case of changes. Moreover, as they will only act in a specific time they will **not** have **temporal continuity** and therefore they will **not learn**.

Mission crew: planner agent
As stated before, the planner agent acts as the entry point for the system by receiving raw mission report and creating a .json which will be send to the priority agent and hazard agent. For this reason the properties that have to fullfill include:

1. **Proactiveness:** since we want the agent to generate the structured task assigment to the priority agent and hazard agent without asking nor waiting for requests.

2. **Social ability:** as the agent needs to communicate and coordinate the splitted information to the priority and hazard agents. These agents will have to trust the information provided by the planner. Regarding security issues, information will be provided without encryption, as the agents will only receive the information related to their main goal.

3. **Rationality:** as the agent must respect constraints such as avoiding unstable or reactive zones, while trying to achieve its goals.

4. **Reasoning capabilites:** the agent should build the structured task assigment, based on its ability to infer based on the knowledge and information provided. It is going to have rule-based reasoning as if it detects hazard -> the information will be assigned to the hazard agent.

5. **Autonomy:** for its autonomy the agent will have access to the information from the .md, and urls given.

This agent is a **collaborative** one, as it coordinates the system and assigns information to other agents.

Mission crew: priority agent
As explained before, this agent will receive the summary from the planner and order the different priorities for each crew. For this reason it has to be:

1. **Proactive:** as it will have to make recommendations without our specific user's request.

2. **Social ability:** because it must communicate effectively with the planner agent (receive) and the aggregator agent (send the priority report).

This agent is not going to be **rational** and is not going to have **reasoning capabilities** as it receives a summary from the planner, extracts relevant information. It does not reason about consequences, balance trade-offs, or adapt its behavior based on context.
This agent would be classified as an **informative** as its main role is to process and distribute information.

Mission crew: hazard agent
This agent is in charge of detecting hazards and constraints, determining which nodes to avoid. Due to its capabilities, it needs to present the following properties:

1. **Social ability:** as it will communicate the hazards and constraints to the aggregator agent.

2. **Rationality:** as it should ensure the plan respects the constraints.

3. **Reasoning capabilities:** because it should infer the hazards and constraints. The type of reasoning it will follow is Rule-Based reasoning as it will process hazards defined in the initial mission data, applying if-then rules, such as if terrain is unstable, mark this node as forbidden.

Regarding **proactiveness**, it will not be proactive as it does not initiate actions, only takes into account possible hazards and constraints.

For all mentioned, the agent will be an **informative agent** as it detects hazards/constraints integrating the information from the planner.

Mission crew: aggregator agent This is the agent responsible for combining the reports of the priority, hazard, and weather agents and sending them to each crew. For this reason the properties that is going to have are:

1. **Social ability:** this is one of the most important properties of this specific agent. It must collaborate with the priority and hazard agents to gather all necessary information and, at the same time, it is responsible for coordinating the various crews by assigning and distributing tasks among them.

2. **Reasoning:** as the agent must process and synthesize data from multiple sources (priority, hazard, weather agents). It needs reasoning to integrate information and produce a structured report for each crew.

3. **Rationality:** the agent needs to make coherent and logical decisions when compiling information and ensuring that each report is accurate and consistent. For example, if conflicting data appear, it should select the most reliable one based on predefined criteria.

Note also that this agent will have really low **proactiveness** as it will act mainly when it has received all necessary data (it won't take independent initiative or pursue new goals beyond compiling and sending reports).
This agent will be an **informative** agent as it will be focused on collecting, filtering, and processing information. We have also though about classifying it as **collaborative**, but as its interaction with other agents are going to be passive (no negotiation between) we have discarded this option.

Rover crew: Extactor agent
As stated before this is a specialized agent that identifies target node IDs from mission reports. The properties that must fullfill are:

1. **Social ability:** as it will have to pass the information to the rover route planner.

2. **Rationality:** as it will have to ensure that proepr nodes are extracted.

3. **Reasoning capabilities:** as it will know the aspect of nodes "NXX".

4. **Proactiveness:** the agent should detect the different nodes once the .md is provided.

This agent is going to be an informative as it will integrate information from the priority.json into a easier to handle format (Python list).

Rover crew: Rover route planner agent
The Rover route planner agent will compute the optimal routes for each of the different rovers, taking into account its energy. This agent will have to present the following properties:

1. **Social ability:** as mentioned, the executor plan agents, in this case physical collector agents, are not going to be present. However, it should be expected that the Rover route planner agent communicates the routes to the different collector agents. Moreover the plans are send to the validation crew.

2. **Rationality:** the agents should ensure it achieves its goals (creating the optimal routes, taking into account the energy).
3. **Reasoning capabilities:** as it computes the optimal rover paths, inferred from graphs, information from the mission and energy computation. The agent needs to evaluate multiple routes and constraints.
4. **Proactiveness:** the agent should compute the routes once the information is provided, without requests. Moreover, the agent should ensure the optimal combination of routes.

This agent would be a **collaborative agent** as it would work with the physical collector rovers, ensuring route coordination.

Rover crew: Sample Collector Agent
The Sample Collector Agent will provide instructions on how the samples should be collected. This agent needs to present the following properties:

1. **Social ability:** it communicates the plans to the Plan integrator agent and hypothetically it would communicate the plans also to the physical collector agents.
2. **Rationality:** the agent should ensure it achieves its goals (creating the optimal sample collecting pipeline).
3. **Reasoning capabilities:** it generates specific collection procedures based on sample type.
4. **Proactiveness:** the agent should compute the collection pipelines once the information is provided, without requests.

As the previous one, this agent would be a **collaborative agent** as it would work with the physical collector rovers, ensuring correct sample collection.

Drone crew: Extactor agent
As stated before this is a specialized agent that identifies target node IDs from mission reports. The properties that must fullfill are:

1. **Social ability:** as it will have to pass the information to the rover route planner.
2. **Rationality:** as it will have to ensure that proepr nodes are extracted.
3. **Reasoning capabilities:** as it will know the aspect of nodes "NXX".
4. **Proactiveness:** the agent should detect the different nodes once the .md is provided.

This agent is going to be an informative as it will integrate information from the priority.json into a easier to handle format (Python list).

Drone crew: Drone flight planner agent
As stated before, this agent is going to compute the optimal routes for each drone, taking into account the report from the aggregator agent. The main characteristics that will have are:

1. **Social ability:** as similarly to the rover route planner agent, it will have to communicate with the hypothetical collector agents. Moreover, it will communicate the plans to the plan integrator agent.
2. **Rationality:** as it is going to make logical decisions to optimize routes under certain constraints.
3. **Reasoning:** as this agent will have to analyze multiple constraints (time, hazards, priorities...).
4. **Proactiveness:** as the agent will have to plan routes ahead and think about alternatives based on the constraints.

This agent would be a **collaborative agent** as it would hypothetically work with drone sampling agents, ensuring route coordination.
Drone crew: Drone Sampling Agent
The Drone Sampling Agent will provide instructions on how the drone samples should be collected. Similar to the sample Collector Agent, this agent needs to present the following properties:

1. **Social ability:** it communicates the plans to the Plan integrator agent and hypothetically it would also communicate the pipelines to the physical drones.
2. **Rationality:** the agent should ensure the creation of the optimal sample collecting pipelines.
3. **Reasoning capabilities:** it generates specific sample collection procedures based on sample types, such as atmospheric samples or CO_2 coverage samples.
4. **Proactiveness:** the agent should compute the collection pipelines once the information is provided, without requests.

As in the previous one, this agent would be a **collaborative agent** as it would work hypothetically with physical drones, ensuring correct sample collection.

Satellite crew: Communication loss extractor agent This agent is responsible for extracting the nodes with communication loss from the graph. For this reason this agent should be:

- **Rationality:** agent should ensure that the graph is properly read and nodes detected.
- **Reasoning capabilities:** it should know how to use the tool to extract the information from the graph.
- **Proactiveness:** the agent should start searching for the nodes once the graph is provided.

Satellite crew: Extractor agent
This agent identifies target node IDs from mission reports. There is a really similar agent for the rover and drone crew, so it will follow the same properties already previously defined there.

Satellite crew: Satellite planner agent
This agent is in charge of computing the optimal satellites that will have to go to a certain node. Between its properties we can find:

1. **Social ability:** the agent communicates plans to the validation crew.
2. **Rationality:** the agents should select the satellites that maximize the satellite coverage.
3. **Reasoning capabilities:** the agent needs to evaluate the satellite coverage and constraints.
4. **Proactiveness:** The agent should compute the orbital plans once the information is provided, without requests.

The Orbit planner agent will be a **collaborative agent** as it would send the orbits to the satellites.

Satellite crew: image capture agent
The image capture agent will provide instructions on how the different images from Mars must be captured. Similar to the Sample Collector Agent, this agent needs to present the following properties:

1. **Social ability:** it communicates the plans to the Plan integrator agent and hypothetically it would also communicate the pipelines to the physical satellites.
2. **Rationality:** the agent should ensure the creation of the optimal image collecting pipelines.
3. **Reasoning capabilities:** it generates specific image collection procedures.
4. **Proactiveness:** the agent should compute the collection pipelines once the information is provided, without requests.

As in the previous one, this agent would be a **collaborative agent** as it would work hypothetically with physical satellites, ensuring correct sample collection.

Validation crew: Validation rover/drone/satellite
The different agents from the validation crew aim to compare the results obtained with the ones requested by the user at the beggining. For this reason the characteristics that the three agents must present are:

1. **Social ability:** as they have to transmit correctly the information to the combine validation agent.
2. **Rationality:** as they have to properly compare the two input provided.
3. **Reasoning capabilities:** the agent needs to evaluate properly if the goals are met.
4. **Proactiveness:** as the agents should start working once the different crews had provided their plans.

It is **informative** because it must clearly report which user-defined nodes were successfully reached and provide a rationale for those that were not.

Validation crew: Combine validations
The Combine validations agent aims to integrate the different validation reports from each Rover, Drone and Satellite. For this reason, the agent should present:

1. **Social ability:** it interacts with other crews to collect validation plans.
2. **Rationality:** the agent needs to select consistent actions, by respecting the decisions made by the other agents.
3. **Reasoning capabilities:** the agent must integrate multiple plans.
4. **Proactiveness:** The agent should merge the different plans, once they are available.

The Combine validations agent will be a **facilitator** as it integrates the different outputs from the different crews, ensuring consistency between the different plans.

Integration crew: Integrator rover/drone/satellite and writer
The different agents from this crew are in charge of writing the final markdowns for the rovers, drones and satellites. Some of the properties they should present include:

1. **Social ability:** they interacts with other crews to collect the plans.
2. **Rationality:** they must ensure that actions align with system objectives.
3. **Reasoning capabilities:** they needs to check consistency.
4. **Proactiveness:** the agents should start integrating the plan once all the data is available.

The different agents from this crew are **facilitator** as they coordinate communication generating the different plans for the different crews.

d. Cooperation and coordination

1. Process definition

Mission crew
The MissionCrew is set to follow a sequential process, but it uses task properties to optimize execution through parallelism. The workflow begins with the `additional_information` task, which extracts data into `extract_md.json`. Once this is complete, the process triggers two different analysis tasks `reporting_priority` and `reporting_hazard` which are explicitly set to async_execution=True. This allows the priority and hazard agents to work in parallel.
The final stage is the `reporting_aggregation` task, which functions as a synchronization point, it requires

the context (outputs) from both asynchronous tasks before the aggregator agent can compile the final JSON report.

Rover crew
The RoverCrew is also set to follow a sequential process. It initiates with the `final_nodes` task (handled by the extractor), which feeds its output as context into the `reporting_route` task. During this second stage, the Route Planner utilizes the RoverPathfindingTool to calculate navigation paths, constrained by a max_iter=3 limit to ensure efficiency. Also, the `reporting_sampling` task is configured with async_execution=True. Because this task is listed last in the sequential crew but lacks a direct context dependency on the previous routing tasks, it is triggered to run asynchronously alongside the main flow. This allows the Sample Collector to begin preparing sampling protocols or analyzing regional data in the background while the primary navigation routes are being finalized and structured into the RouteOutput Pydantic model.

Drone crew
The DronesCrew also uses a sequential process. The workflow begins with the `final_nodes` task, where the Extractor agent processes the raw mission data. This leads into the reporting_route task, which is strictly dependent on the output of the first task. Here, the Flight Planner uses the `BatchDroneFlightTool` to calculate optimized flight paths, saving the results to routes_drone.json. Parallel to this main logical branch, the `generate_sampling` task is executed with async_execution=True. This allows the Sample Collector to operate independently and concurrently, generating the sampling strategies in sample_collector_drone.md without waiting for the flight routes to be finalized.

Satellite crew
The SatelliteCrew follows a sequential process. The execution starts with three independent tasks: `task_communication_loss_extractor` (using the CommunicationLossTool), `task_extractor`, and `task_image_capture`. These three are set to async_execution=False. The process finishes in the `task_planner`, which uses the Planner agent to synthesize information from both the communication loss analysis and the data extraction tasks via its context parameter. The final routes_satellite.json is provided.

Validation crew
The ValidationCrew follows a sequential process. Three different tasks (`task_validation_rover`, `task_validation_drone` and `task_validation_satellite`) are all configured with async_execution=True. This allows the Rover, Drone and Satellite validation agents to work in parallel without waiting for one another. Then, `task_combine_validations`, includes all three previous tasks in its context. It is the only task that is not asynchronous, this ensures that the Combine Validations agent only begins its work once the three parallel reports are finished, to then aggregate it into combine_validations.json file.

Integration crew
The IntegrationCrew is organized as a sequential process. The workflow starts with three tasks simultaneously: `task_integrator_rover`, `task_integrator_drone` and `task_integrator_satellite`. Because these are configured with async_execution=True. The process finishes with the `task_writer`, which acts as the overall aggregator. This task is strictly dependent on the completion of the three asynchronous tasks, as defined in its context. The Writer agent takes the integrated content and maps it to a structured FinalMissionReport Pydantic output, which will be further explained in the following section.

2. Pydantic Outputs

In order to maximize compatibility between agent outputs and ensure the correct formatting of the results, Pydantic outputs have been introduced. Pydantic is a python library that helps establish the structure of the data. In the context of the presented MAS, it allows to make sure that the output of a specific task or validation procedure will coincide with a custom structure defined a priori. In particular, five different Pydantic schemas have been used: one for the rover crew, two for the satellite crew, one to validate the results, and one for the integration crew.

The role of the schema inside the rover crew consists of guaranteeing that the results from the route planner agent are contained inside a valid JSON file with a ‘results’ dictionary. This ensures that the information returned by the agents is in the exact format necessary for subsequent tasks. Similarly, in the Satellite crew the two schemas determine how to express the satellite assingments (id, goal, location and communication window) and how to generate the task planning, respectively.

Regarding the schema used to validate the results, it allows to evaluate the overall results provided by each crew (rover, drone and satellite) as True (correct) or False (incorrect). The object that contains these boolean evaluations is later used to determine if it is necessary to replan, or that the flow can continue towards the integration step.

Finally, the last Pydantic schema introduced in the integration crew defines a more structured output than in the previous cases, in which the schemas usually served as an intermediate data validation. In the integration crew, the application of the schema serves to detail how the final report of the system will be. It forces the writer agent to include a title, a table of contents, individual sections for each crew and a final overall conclusion in the resulting md file. This approach guides the agent and prevents the omission of critical parts of the mission during the final report generation. Moreover, it standardizes the format of the report, so if it was to be

utilized in posterior tasks it would facilitate its usage.

3. Agent Interaction

The agent interaction in our system can be divided into four phases:

- **Phase 1: mission analysis.** The flow begins with the `Run Mission Analysis (Start Method)`. The agents involved in this phase are the planner, priority, hazard and aggregator agents.
- **Phase 2: parallel planning.** The flow splits into three concurrent "Crew Methods." Each crew operates under the defined constraints.
- **Phase 3: the validation loop.** The validation crew (rover, drone and satellite validators) performs a binary check. They verify if 100% of target nodes are reached by each corresponding crew. If failed, the router trigger sends the process back to the respective planning crews. This iterative feedback ensures that if a route is flagged as "unsafe" or "incomplete," the specific crew must recalculate. If passed, the flow proceeds to the final integration.
- **Phase 4: integration** Once the decision router receives a "True" status from the combine validations agent, it triggers Integrate plans. The agents involved are the integrator rover, drones, satellite and the writer.

The complete scheme of the flow can be seen in Figure 3. *As explained before, on the previous delivery the validation crew was defined as an agent inside the integration crew, this was modified so that no invalid data was merged into the final report.*

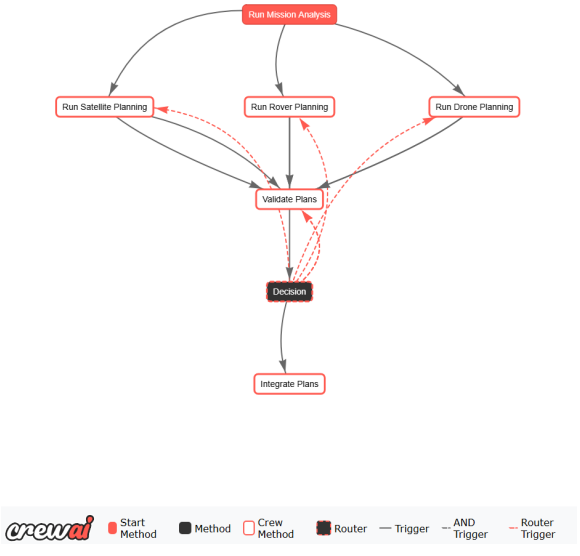

Figure 3: Final flow of the system created with CrewAI.

e. Execution

For the execution of the multi-agent system, the use of local models has been prioritized. Initially, the **Phi-4** model was employed using 14B of parameters. However, in the preliminary developmental phases, several limitations were found, such as their reasoning capacity. This model tended to hallucinate, specially in data extraction tasks and to interpret the graph hazards. This instability obliged us to design excessive directed prompts and to depend exclusively on tools, limiting the autonomous reasoning of the agents. Moreover, free cloud-based models, such as Gemini-2.5 and **Gemini-2.5-lite** (8B parameters), were also considered. However, the free versions offered neither significantly more parameters (Gemini-2.5-lite was not sufficient) nor sufficient free tokens (for the case of Gemini-2.5).

Given the necessity of finding a better trade-off between autonomy, precision of the results, and unrestricted token volume, we opted for using locally the **DeepSeek-R1:14b** model. This model is designed specially with reinforcement reasoning capacities, which demonstrated a substantial improvement in the comprehension of the context and logical decisions without inventing nodes or nonexistent restrictions. For the final execution, the model has been configured with a temperature of 0.1, maximizing the determinism and the reliability for the planning of the Martian exploration.

The experiments were conducted on a system equipped with an Intel Core i7 processor, 16 GB of DDR4 RAM, and an NVIDIA RTX 4060 Laptop GPU with 8 GB of dedicated VRAM.

III. Results

The different results obtained with the MAS previously described are presented in this section. In order to make the explanation more readable, some results have been abbreviated as [...]. The full outputs are available in the GitHub repository provided in Section VI. It should be noted that the results obtained from the hazard-enabled execution are not reported in this section but are available in the GitHub repository. These results are omitted here because hazards only affect rover and drone operations, and their routes are computed deterministically by a dedicated tool, leading to highly similar outputs across executions. The primary difference lies in the validation stage, where hazard constraints are explicitly considered. Furthermore, due to the increased complexity introduced by hazard handling and the known tendency of language models to hallucinate under such conditions, this study focuses on the hazard-free execution scenario to ensure clarity and reproducibility.

a. Mission crew

Additional Information

As it can be seen in the output provided below (Listing 1), the planner agent is able to properly extract the different sections of the input report through the usage of the provided tool.

Listing 1: extract md (JSON)

```
{"scientific goals":["Collect subsurface samples
    ↪ from **rocky** terrain near nodes **N70**
```

```
    ↪ and **N102**.","Capture panoramic images
    ↪ of **crater** terrain at nodes **N5**, **
    ↪ N58**, **N121** and **N150**.","[...]"],"
    ↪ operational constraints":["**Rovers** and
    ↪ **drones** should recharge if energy drops
    ↪  below 30%.","**Rovers** must enter a heat
    ↪  shelter during 20 minutes if the surface
    ↪ temperature is below -60C.","[...]"],"
    ↪ mission priorities":{"high":["Radiation
    ↪ measurement at **N12**, **N86** and **N1
    ↪ **.","[...]"],"medium":["Air sample
    ↪ collection at **N33**.","[...]"],"low":["
    ↪ Panoramic imaging at **N5**, **N58**, **
    ↪ N121** and **N150**.","[...]"]},"known
    ↪ hazards":["**Nodes N4, N19, N128**:
    ↪ unstable rocky terrain","[...]"]}
```


Reporting Priority and Hazards

Based on the information provided by the planner agent, the priority and hazard agents extract asynchronously the goals and risks affecting each crew, respectively. They are returned as shown in Listings 2 and 3. In each dictionary, a specific entry is created to detail the information about a certain crew. The priorities are divided into 'high', 'medium' and 'low', and the hazards are distinguish from operational constraints.

Listing 2: Priority report (JSON)


```
{"rover":{"high":["Measure radiation levels in **
    ↪ sandy** terrain at node **N12**, **N86**
    ↪ and **N1**.","[...]"],"medium":["Detect
    ↪ organic molecules in **icy** terrain at **
    ↪ N59**."],"low":[]},"drone":{"high":[],"
    ↪ medium":["Air **dust** sample collection
    ↪ during flight over **N33** crater
    ↪ .","[...]"],"low":[]},"satellite":{"high
    ↪ ":[],"medium":[],"low":["Capture panoramic
    ↪  images of **crater** terrain at nodes **
    ↪ N5**, **N58**, **N121** and **N150
    ↪ **.","[...]"]}}
```


Listing 3: Hazard report (JSON)


```
{"rover":{"operational constraints":["Rovers and
    ↪ drones should recharge if energy drops
    ↪ below 30%","Rovers must enter a heat
    ↪ shelter during 20 minutes if the surface
    ↪ temperature is below -60C.","[...]"],"
    ↪ hazards":{"Rovers may not operate in
    ↪ terrain classified as unstable.":["N4","
    ↪ N19","N128"],"[...]":[]}},"drone":{"
    ↪ operational constraints":["Rovers and
    ↪ drones should recharge if energy drops
    ↪ below 30%","[...]"],"hazards":{"Drones are
    ↪  disabled if there is a dust storm.":["N33
    ↪ "]}},"satellite":{"operational constraints
    ↪ ":["Satellites must maintain communication
    ↪  with the base station at N30 or N84 every
    ↪  5 hours if they have to identify thermal
    ↪ anomalies.","[...]"],"hazards":[]}}
```


Reporting Aggregation

The final result provided by the mission crew, which is created by the aggregator agent, consists of a JSON that includes a specific report for each crew, with the priorities, operational constraints and hazards. It is presented in Listing 4

Listing 4: Aggregation report (JSON)


```
{"rovers":{"priority":{"high":["Measure radiation
    ↪  levels in **sandy** terrain at node **N12
    ↪ **, **N86** and **N1**.","[...]"],"medium
    ↪ ":["Detect organic molecules in **icy**
    ↪ terrain at **N59**."],"low":[]},"
    ↪ operational constraints":["Rovers and
    ↪ drones should recharge if energy drops
    ↪ below 30%","Rovers must enter a heat
    ↪ shelter during 20 minutes if the surface
    ↪ temperature is below -60C.","[...]"],"
    ↪ hazards":{"Rovers may not operate in
    ↪ terrain classified as unstable.":["N4","
    ↪ N19","N128"],"[...]":[]}},"drones":{"
    ↪ priority":{"high":[],"medium":["Air **dust
    ↪ ** sample collection during flight over **
    ↪ N33** crater.","[...]"],"low":[]},"
    ↪ operational constraints":["Rovers and
    ↪ drones should recharge if energy drops
    ↪ below 30%","[...]"],"hazards":{"Drones are
    ↪  disabled if there is a dust storm.":["N33
    ↪ "]}},"satellites":{"priority":{"high":[],"
    ↪ medium":[],"low":["Capture panoramic
    ↪ images of **crater** terrain at nodes **N5
    ↪ **, **N58**, **N121** and **N150
    ↪ **."],"[...]"]},"operational constraints
    ↪ ":["Satellites must maintain communication
    ↪  with the base station at N30 or N84 every
    ↪  5 hours if they have to identify thermal
    ↪ anomalies.","[...]"],"hazards":[]}}
```


b. Rover crew

Reporting Route

The route planer agent is responsible for providing the rouges each rover would need to do. By the use of a tool, it is able to compute the optimal route for rovers and assign it to a specific input rover. Notice that the routes are circular, meaning that they start and end at the same position (see Listing 5).

Listing 5: Route report (JSON)


```
{"results": {
    "rover_0": [["N30","N28","N23","N20","N25","
        ↪ N30"],["N30","N28","N23","N18","N13","
        ↪ N11","N9","N4","N2","N1","N6","N8","
        ↪ N13","N18","N23","N28","N30"]],
    "rover_1": [["N84","N85","N86","N84"],["N84","
        ↪ N82","N77","N72","N67","N62","N60","
        ↪ N59","N62","N67","N72","N77","N82","
        ↪ N84"]],
    "rover_2": [["N84","N82","N77","N72","N70","
        ↪ N72","N77","N82","N84"]],
    "rover_3": [["N30","N28","N23","N22","N21","
        ↪ N18","N13","N12","N13","N18","N23","
        ↪ N28","N30"]],
    "rover_4": [["N84","N85","N88","N90","N95","
        ↪ N96","N101","N102","N101","N96","N95
        ↪ ","N94","N93","N88","N85","N84"]]}
}
```

Reporting Sampling
The sample collector agent is in charge of providing detailed report (ten-stage procedure) specifying the procedures to be conducted by rovers. The report is represented below (Listing 6).

Listing 6: Sample collector (MD)

```
# Rover Sampling Protocol Document

## 1. Measure Radiation Levels in Sandy Terrain
    ↪ at Nodes N12, N86, and N1

### Step-by-Step Procedure:

1. **Rover Navigation**: The rover will navigate
    ↪ to node N12 using pre-mission maps and GPS
    ↪  guidance.
2. **Terrain Assessment**: Upon arrival, the
    ↪ rover will assess the sandy terrain for
    ↪ any obstacles or uneven surfaces.
[...]

### Notes:
- Ensure all movements are logged for future
    ↪ reference.
- Monitor rover health throughout the operation.

---

## 2. Collect Subsurface Samples from Rocky
    ↪ Terrain Near Nodes N70 and N102

### Step-by-Step Procedure:

1. **Rover Navigation**: Traverse to node N70
    ↪ using pre-loaded terrain maps.
2. **Terrain Analysis**: Assess the rocky terrain
    ↪  for stability and potential hazards.
[...]

### Notes:
- Conduct regular equipment checks before each
    ↪ deployment.
- Be vigilant of any unexpected geological
    ↪ features.

---

## 3. Deploy Seismic Sensors at Rocky Node N20

### Step-by-Step Procedure:

1. **Rover Navigation**: Travel to node N20,
    ↪ ensuring adherence to predefined paths.
2. **Terrain Evaluation**: Assess the rocky
    ↪ terrain for optimal sensor placement.
[...]

### Notes:
- Ensure all sensors are functioning optimally
    ↪ before leaving the area.
- Monitor for any signs of equipment failure.

---

## 4. Detect Organic Molecules in Icy Terrain at
    ↪ Node N59

### Step-by-Step Procedure:

1. **Rover Navigation**: Navigate to node N59,
    ↪ avoiding any icy patches that could hinder
    ↪  movement.
2. **Terrain Inspection**: Inspect the icy
    ↪ terrain for signs of organic residue or
    ↪ discoloration.
[...]

### Notes:
- Maintain strict sterility throughout the
    ↪ sampling process.
- Document any unusual observations in the area.

---

# Final Report

## Samples Collected:

1. Radiation levels measured at nodes N12, N86,
    ↪ and N1.
2. Subsurface samples collected from rocky
    ↪ terrain near nodes N70 and N102.
3. Seismic sensors deployed at node N20.
4. Organic molecule detection scans conducted at
    ↪ node N59.

## Sensors Used:

- Radiation sensor module
- Subsurface drill/core sampler
- Seismic sensor array
- Spectrometer

## Observations:

- All operations completed within expected
    ↪ timeframes.
- No significant technical issues encountered
    ↪ during rover deployment.
```

c. Drone crew

Reporting Route
The flight planner agent returns a report detailing each input drone and the routes it is committed to do. The routes are computed by the use of tool and assigned to a unique drone. Similar to the routes computed for rovers, in drones they are also circular (see Listing 7).

Listing 7: Route report (JSON)

```
{"drone_0": [["N30", "N33", "N30"]],
"drone_2": [["N30", "N31", "N36", "N41", "N46", "
    ↪ N51", "N53", "N51", "N46", "N41", "N36", "
    ↪ N31", "N30"]],
"drone_1": [["N84", "N83", "N78", "N73", "N68", "
    ↪ N63", "N68", "N73", "N78", "N83", "N84"]],
"drone_3": [["N84", "N86", "N91", "N96", "N101",
    ↪ "N106", "N108", "N106", "N101", "N96", "
    ↪ N91", "N86", "N84"]]}
```

Generate Sampling

The sample collector agent is responsible for generating a detailed report (ten-stage procedure) with the specifications of the tasks to be conducted by drones. The report is depicted below (Listing 8).

Listing 8: Sample report (JSON)

```
# Drone Sampling Procedures

## Procedure 1: Air Dust Sample Collection during
    ↪ Flight over N33 Crater

### Step 1: Pre-Flight System Check
- Conduct a thorough pre-flight inspection of the
    ↪ drone, including battery levels,
    ↪ propeller function, and communication
    ↪ systems.

### Step 2: Take Off from Designated Launch Pad
- Initiate takeoff using remote control or
    ↪ autonomous mode, ensuring stable ascent.
[...]
---

## Procedure 2: Map CO2 Frost Coverage in Icy
    ↪ Areas (N53, N63, N108)

### Step 1: Pre-Flight System Check
- Conduct a detailed pre-flight inspection of the
    ↪ drone's systems, including camera and
    ↪ sensor functionality.

### Step 2: Take Off from Designated Launch Pad
- Initiate takeoff using remote control or
    ↪ autonomous mode, ensuring stable ascent.
[...]
```

d. Satellite crew

Planner

The first provided output (Listing 9), does not provide the correct goal nodes or the communication window, causing this result to be incorrect. The incorrect output is provided here as it comes from the same execution as all the other results. However, when running the same code individually for the satellite crew, a correct response was obtained (Listing 10). The correct execution shows each satellite together with its location, goal node, and communication window.

Listing 9: Route report, incorrect (JSON)

```
{"assignments":[{"id":"Satellite_0","goal":"","
    ↪ location":"N5","communication_window
    ↪ ":""},{"id":"Satellite_1","goal":"","
    ↪ location":"N58","communication_window
    ↪ ":""},{"id":"Satellite_2","goal":"","
    ↪ location":"N121","communication_window
    ↪ ":""},{"id":"Satellite_3","goal":"","
    ↪ location":"N150","communication_window
    ↪ ":""},{"id":"Satellite_4","goal":"","
    ↪ location":"N56","communication_window
    ↪ ":""},{"id":"Satellite_5","goal":"","
    ↪ location":"N112","communication_window
    ↪ ":""}]}
```

Listing 10: Route report, correct (JSON)

```
{"id":"Satellite_0","goal":"N5","location":"N5","
    ↪ communication_window":7}
{"id":"Satellite_1","goal":"N58","location":"N58
    ↪ ","communication_window":8}
{"id":"Satellite_4","goal":"N56","location":"N56
    ↪ ","communication_window":4}
{"id":"Satellite_2","goal":"N121","location":"
    ↪ N121","communication_window":3}
{"id":"Satellite_3","goal":"N150","location":"
    ↪ N150","communication_window":2}
{"id":"Satellite_5","goal":"N112","location":"
    ↪ N112","communication_window":5}
```

Image Capture

The image capture agent is responsible for generating a report detailing the operation. This obtained documentation is shown below (Listing 11). It is based on a five-step procedure for each of the tasks to be executed by the satellite crew (panoramic image capture and thermal anomaly detection).

Listing 11: Image capture report (MD)

```
\boxed{ \begin{aligned} &\text{Protocol 1:
    ↪ Panoramic Image Capture for Crater Terrain
    ↪ } \ &\text{Step 1: Sensor Initialization -
    ↪  Power on the satellite's camera and
    ↪ ensure all systems are operational.} \ &\
    ↪ text{Step 2: Positioning - Manually or
    ↪ automatically position the satellite over
    ↪ nodes N5, N58, N121, and N150, checking
    ↪ for communication signals.} \ [...] \ & \
    ↪ &\text{Protocol 2: Thermal Anomaly
    ↪ Detection for Icy Terrain} \ &\text{Step
    ↪ 1: Sensor Initialization - Activate the
    ↪ thermal imaging sensor and ensure
    ↪ calibration is complete.} [...] \end{
    ↪ aligned} }
```

e. Validation crew

Here, the output of validation rover agent (Listing 12), validation drone agent (Listing 13) and validation satellite agent (Listing 14) is provided. Those intermediate outputs check if the different routes provided by the previous crews are correct 'e.g. rover_ok', and if not say why with the 'e.g. feedback_rover'. Listing 15 combines them by only providing true or false for rovers, drones and satellites.

Listing 12: Rover verification feedback (JSON)

```
{
  "rover_ok": true,
  "feedback_rover": "All required mission nodes (
      ↪ N1, N12, N20, N59, N70, N86, N102) have
      ↪ been visited by the rovers. No hazardous
      ↪  nodes were traversed."
}
```

Listing 13: Drone verification feedback (JSON)

```
{
  "drone_ok": true,
```

```
  "feedback_drone": "All required nodes (N33, N53
      ↪ , N63, N108) are covered by the drone
      ↪ routes. Drone_0 covers N33, Drone_2
      ↪ covers N53, Drone_1 covers N63, and
      ↪ Drone_3 covers N108."
}
```

Listing 14: Satellite verification feedback (JSON)

```
{
  "satellite_ok": true,
  "feedback_satellite": "All satellite
      ↪ assignments match the required nodes
      ↪ from the priorities section. Thermal
      ↪ anomaly detection tasks at N56 and N112
      ↪ do not exceed communication window
      ↪ constraints."
}
```

Listing 15: All verification feedback (Run 3/3) (JSON)

```
{
  "rover_ok": true,
  "drone_ok": true,
  "satellite_ok": true
}
```

Note that for this run we firstly obtained that the SatelliteCrew had an error, so it was called again (with the router function) to solve this. The output for validation 1/3 can be found on Listing 16.

Listing 16: All verification feedback (Run 1/3) (JSON)

```
{
  "rover_ok": true,
  "drone_ok": true,
  "satellite_ok": false
}
```

f. Integration crew

Here, the output of integration rover agent (Listing 17), integration drone agent (Listing 18) and integration satellite agent (Listing 19) is provided. Clear sections for trajectory, protocols and instrumentation can be observed.

Listing 17: Rover integration report (Markdown Report)

```
# INTEGRATED MARS SURFACE OPERATIONS REPORT

## Section 1: Asset Trajectory Logs (by Rover ID)

### Rover_0:
- **Trajectory 1**: N30 -> N28 -> N23 -> N20 ->
    ↪ N25 -> N30
- **Trajectory 2**: N30 -> N28 -> N23 -> N18 ->
    ↪ N13 -> N11 -> N9 -> N4 -> N2 -> N1 -> N6
    ↪ -> N8 -> N13 -> N18 -> N23 -> N28 -> N30
[...]
---
## Section 2: Integrated Scientific Protocols (
    ↪ Mapping Nodes to Procedures)

### Procedure 1: Collect Subsurface Samples from
    ↪ Rocky Terrain near Nodes N70, N102, and
    ↪ N152
- **Assigned Rover**: Rover_2 (N70), Rover_4 (
    ↪ N102)
- **Objective**: Collect subsurface samples for
    ↪ scientific analysis.

### Procedure 2: Measure Radiation Levels in
    ↪ Sandy Terrain at Nodes N12, N86, and N1
[...]
---
## Section 3: Instrumentation & Hardware Summary

The following equipment is essential for
    ↪ executing the outlined procedures:

1. **Coring Drill**
2. **Ground-Penetrating Radar (GPR)**
[...]
---
## Executive Summary

This report integrates the spatial trajectories
    ↪ of five Martian rovers with their
    ↪ respective scientific sampling protocols.
    ↪ The mission objectives include subsurface
    ↪ sample collection, radiation level
    ↪ measurement, seismic sensor deployment,
    ↪ and organic molecule detection.
[...]
```

Listing 18: Drone integration report (Markdown Report)

```
# INTEGRATED MARS AERIAL OPERATIONS REPORT

## Section 1: Flight Trajectory Logs (by Drone ID
    ↪ )

### Drone_0
- **Trajectory**: N30 -> N33 -> N30
- **Notes**: This drone's path is a triangular
    ↪ route centered around N30, with a focus on
    ↪  N33.
[...]
---

## Section 2: Aerial Scientific Protocols (
    ↪ Mapping Nodes to Procedures)

### Air Dust Sample Collection at N33
- **Assigned Drone**: Drone_0
- **Protocol**: Conducted over N33 Crater,
    ↪ involving pre-flight checks, takeoff,
    ↪ navigation, hover at 500m, dust sampling
    ↪ at multiple altitudes, data transmission,
    ↪ and return to launch pad.
[...]
---
## Section 3: Sensor & Instrumentation Overview

### Environmental Monitoring Requirements
- **Wind Speed**: Monitored to ensure safe drone
    ↪ operation.
- **Temperature**: Recorded for atmospheric data
    ↪ collection.
[...]
---
This report synthesizes the flight trajectories,
```

```
    ↪ scientific protocols, and instrumentation
    ↪ used during the mission, ensuring
    ↪ alignment with Mars aerial operations
    ↪ objectives.
```

Listing 19: Satellite integration report (Markdown Report)

```
# INTEGRATED MARS ORBITAL OPERATIONS REPORT

## Section 1: Orbital Window & Assignment Logs

| Satellite ID | Goal | Location | Communication
    ↪ Window |
| Satellite_0 | Crater Terrain Imaging| N5 |
    ↪ 12:00-13:00 |
| Satellite_1 | Crater Terrain Imaging| N58 |
    ↪ 14:00-15:00 |
| Satellite_2 | Crater Terrain Imaging| N121 |
    ↪ 16:00-17:00 |
[...]
## Section 2: Sensor Execution Protocols

### Protocol 1: Panoramic Image Capture for
    ↪ Crater Terrain
- **Assigned Satellites**: Satellite_0,
    ↪ Satellite_1, Satellite_2, Satellite_3
- **Execution Steps**:
  1. **Sensor Initialization**: Power on the
      ↪ satellite's camera and ensure all
      ↪ systems are operational.
  2. **Positioning**: Manually or automatically
      ↪ position the satellite over nodes N5,
      ↪ N58, N121, and N150, checking for
      ↪ communication signals.
[...]
## Section 3: Data Transmission & Validation
    ↪ Metrics

### Communication Window Feasibility
- **Protocol 1**:
  - Each satellite (Satellite_0 to Satellite_3)
      ↪ has a dedicated communication window of
      ↪ 1 hour per node.
  - The sequential capture and transmission
      ↪ process is logistically feasible within
      ↪ the allotted time frame.
[...]
### Conclusion
The integrated orbital operations plan ensures
    ↪ seamless execution of both panoramic
    ↪ imaging and thermal anomaly detection
    ↪ protocols. The communication windows are
    ↪ optimized for mission success, and the
    ↪ sequential capture and validation
    ↪ processes guarantee high-quality
    ↪ scientific outcomes.
```

g. Final output

Here, the output for the final report (final_mission_report.md) is provided. This output is clearly structured into five sections. Firslty, it presents the table of contents. Secondly, the operations that must be taken into account on the Mars surface (by the rovers). Thirdly, the aerial operations report (that drones will take into account). Then, the orbital integration report (for satellites). And lastly, the mission conclusions (saying that the mission is ready for execution!).

MARS JOINT OPERATIONS: FINAL MISSION CONFIGURATION

Table of Contents

INTEGRATED MARS SURFACE OPERATIONS REPORT

Section 1: Asset Trajectory Logs (by Rover ID)

Rover_0:

- **Trajectory 1**: N30 → N28 → N23 → N20 → N25 → N30
- **Trajectory 2**: N30 → N28 → N23 → N18 → N13 → N11 → N9 → N4 → N2 → N1 → N6 → N8 → N13 → N18 → N23 → N28 → N30

Rover_1:

- **Trajectory 1**: N84 → N85 → N86 → N84
- **Trajectory 2**: N84 → N82 → N77 → N72 → N67 → N62 → N60 → N59 → N62 → N67 → N72 → N77 → N82 → N84

Rover_2: N84 → N82 → N77 → N72 → N70 → N72 → N77 → N82 → N84
Rover_3: N30 → N28 → N23 → N22 → N21 → N18 → N13 → N12 → N13 → N18 → N23 → N28 → N30
Rover_4: N84 → N85 → N88 → N90 → N95 → N96 → N101 → N102 → N101 → N96 → N95 → N94 → N93 → N88 → N85 → N84

Section 2: Integrated Scientific Protocols

Procedure 1: Collect Subsurface Samples from Rocky Terrain near Nodes N70, N102, and N152

- **Assigned Rover**: Rover_2 (N70), Rover_4 (N102)
- **Objective**: Collect subsurface samples for scientific analysis.

Procedure 2: Measure Radiation Levels in Sandy Terrain at Nodes N12, N86, and N1

- **Assigned Rover:**: Rover_1 (N86), Rover_3 (N12)
- **Objective:** Measure radiation levels and collect soil samples.

Procedure 3: Deploy Seismic Sensors at Rocky Node N20

- **Assigned Rover:** Rover_0 (N20)
- **Objective:** Deploy seismic sensors for data collection.

Procedure 4: Detect Organic Molecules in Icy Terrain at Node N59

- **Note**: No rover trajectory includes node N59; this procedure cannot be mapped.

Section 3: Instrumentation & Hardware Summary

1. Coring Drill,
2. Ground-Penetrating Radar (GPR),
3. Radiation Detection Units,
4. Seismic Sensor Deployment Tools,
5. Organic Molecule Detection Units,
6. GPS Navigation System,
7. Sampling Containers (Sterile),
8. Anchoring Equipment.
9. Communication Devices for Data Transmission
10. Protective Shielding for Sensitive Equipment
11. Anti-Freeze Protection for Equipment

Executive Summary

This report integrates the spatial trajectories of five Martian rovers with their respective scientific sampling protocols. While most procedures are feasible, detecting organic molecules at Node N59 remains unachievable due to rover limitations.

INTEGRATED MARS AERIAL OPERATIONS REPORT

Section 1: Flight Trajectory Logs (by Drone ID)

Drone_0:

- **Trajectory:** N30 → N28 → N23 → N20 → N25 → N30
- **Assigned Mission:** High-resolution imaging of rocky terrain.

Drone_1:

- **Trajectory:** N84 → N85 → N86 → N84
- **Assigned Mission:** Environmental data collection in sandy regions.

Drone_2:

- **Trajectory:** N30 → N28 → N23 → N22 → N21 → N18 → N13 → N12 → N13 → N18 → N23 → N28 → N30
- **Assigned Mission:** Surveillance of seismic activity.

Drone_3:

- **Trajectory:**
- N84 → N85 → N88 → N90 → N95 → N96 → N101 → N102 → N101 → N96 → N95 → N94 → N93 → N88 → N85 → N84
- **Assigned Mission:** Thermal anomaly detection in icy terrains.

Section 2: Aerial Scientific Protocols (Mapping Nodes to Procedures)

Protocol 1: High-Resolution Imaging of Rocky Terrain

- **Assigned Drones:** Drone_0, Drone_2
- **Objective:** Capture detailed images for geological analysis.

Protocol 2: Environmental Data Collection in Sandy Regions

- **Assigned Drones:** Drone_1
- **Objective:** Monitor atmospheric conditions and surface stability.

Protocol 3: Surveillance of Seismic Activity

- **Assigned Drones:** Drone_2
- **Objective:** Track ground movements for predictive modeling.

Protocol 4: Thermal Anomaly Detection in Icy Terrains

- **Assigned Drones:** Drone_3
- **Objective:** Identify temperature irregularities indicative of geothermal activity.

Section 3: Sensor & Instrumentation Overview

Imaging Sensors:

1. High-resolution cameras for terrain mapping.
2. Thermal imaging sensors for anomaly detection.

Environmental Monitoring:

1. Atmospheric sensors for data collection.
2. Motion sensors for seismic surveillance.

Data Transmission:

1. Real-time communication with ground control for immediate feedback and adjustments.

INTEGRATED MARS ORBITAL OPERATIONS REPORT

Section 1: Orbital Window & Assignment Logs

Satellite	Mission	Node	Comm. Window
Satellite_0	Crater Terrain Imaging	N5	12:00-13:00
Satellite_1	Crater Terrain Imaging	N58	14:00-15:00
Satellite_2	Crater Terrain Imaging	N121	16:00-17:00
Satellite_3	Crater Terrain Imaging	N150	12:00-13:00
Satellite_4	Icy Terrain Thermal	N56	20:00-21:00
Satellite_5	Icy Terrain Thermal	N112	22:00-23:00

Section 2: Sensor Execution Protocols

Protocol 1: Panoramic Image Capture for Crater Terrain

- **Assigned Satellites:** Satellite_0, Satellite_1, Satellite_2, Satellite_3
- **Execution Steps:**
  1. Power on cameras and ensure operational readiness.
  2. Position over nodes N5, N58, N121, N150, checking communication signals.
  3. Capture high-resolution images in sequence, ensuring complete coverage.
  4. Validate data for clarity and completeness.
  5. Transmit to ground control, prioritizing critical nodes (N150).

Protocol 2: Thermal Anomaly Detection for Icy Terrain

- **Assigned Satellites:** Satellite_4, Satellite_5
- **Execution Steps:**
  1. Activate thermal imaging sensors and calibrate.
  2. Position over nodes N56 and N112, checking communication signals.
  3. Scan for temperature irregularities using thermal sensors.
  4. Analyze data for anomalies and validate findings.
  5. Transmit results to ground control with priority given to critical nodes (N112).

Section 3: Data Transmission & Validation Metrics

Communication Window Feasibility

- **Protocol 1:** Each satellite has a dedicated 1-hour window per node, ensuring timely data transmission.
- **Protocol 2:** Satellites have a dedicated 1-hour window per node, aligning with thermal anomaly detection processes.

Data Validation Metrics

- **Image Capture:** Achieve $\geq 90\%$ clarity for mission success.
- **Thermal Data:** Ensure anomaly detection accuracy through ground-based modeling.
- **Transmission Success Rate:** Maintain 100% data transmission to ground control, prioritizing critical nodes.

Conclusion

The integrated orbital operations plan ensures seamless execution of imaging and thermal anomaly detection protocols. Communication windows are optimized for success, with sequential capture and validation processes guaranteeing high-quality scientific outcomes.

Conclusion

The MARS JOINT OPERATIONS mission is fully operational across surface, aerial, and orbital platforms. All systems have been successfully integrated. The mission is ready for execution.

IV. Discussion

During the experimental phase of this project, more than 40 runs of the system have been performed. This high number of iterations has been necessary due to the variable behavior of the agents and the fact that they diverged. This instability can be understood as something analogous to accumulated probability: if each agent has a certain probability of succeeding in its task(s), the probability that the whole system of agents works without any error is their product. Even if the agents have relatively high probabilities (e.g., 70%), it is only necessary that one of them fails for the final result to be invalid.

In the initial executions, the most common errors found included hallucinations, format errors and ignoring the provided tools. Hallucinations correspond to cases in which, for instance, the agents invented geographical data or answered non-existing parts of the mission. Regarding the format errors, agents often included chunks of plain text (e.g. "Here are the routes for the rovers") or triple quotations around the JSON files (""""). These errors were often breaking posterior tasks, because they caused the file structure to be incorrect. Furthermore, they are also more likely to occur as task complexity increases, as observed in the hazard scenario. As shown in the repository, even after three replanning attempts, the system is still unable to generate a fully valid plan.

In order to solve the errors, the agent prompts were iteratively refined and individually tested. With these modifications of the the roles, the goals and the backstories some agents were able to understand and perform their tasks correctly most of the time. However, in some cases it was also necessary to introduce tools, explicitly define how to use them in the agent task definition and clearly mention that no other output apart from the tool results was valid. Otherwise, even though the agent correctly called the tool, it later applied incorrect modifications to the output. Another type of solution that was very helpful were Pydantic outputs, to restrict the format of the results.

After completing all the iterative modifications, we have now achieved that all crews can successfully work individually and produce the expected results. Still, when executing the whole system, some errors or inconsistencies may sometimes appear. If this happens, the validation crew is in charge of detecting them, and the router trigger contacts the crew(s) involved. While this mechanism alleviates some problems, perfect results are difficult to be obtained.

Another important aspect is that the system generates two different plans: one of them taking into account the hazards, and another one without considering them. As in the real world hazards can occur or not with a certain probability, agents have a contingency plan that they can use in case the hazards are found. Otherwise, the plan without hazards is enough.

V. Conclusion

Overall, the results obtained in this project indicate that multi agent systems could automatize or help in the development of plans to explore foreign planets like Mars. The implemented architecture with CrewAI has demonstrated that agents are extremely susceptible to the prompts they are given, and to the results and formats provided by previous agents in the flow. Yet, their refinement combined with the usage of custom tools and Pydantic outputs allows to guide the system and converge towards plans that are generally valid.

VI. Code specifications

The code used in this study is available at the following GitHub repository: `https://github.com/hsanchezulloa/IMAS`.

References

[1] J. Moura, “Crewai: Framework for orchestrating role-playing, autonomous ai agents,” https://github.com/joaomdmoura/crewai, 2024.

[2] S. H. Raza. (2025, Mar.) Environment types and types of agents: A comprehensive study in artificial intelligence. [Online]. Available: https://medium.com/@ai.mlresearcher/environment-types-and-types-of-agents-d8bdd9286115

[3] J. Pascual Fontanilles, “Lecture 2: Agent architectures,” Universitat Rovira i Virgili, 2025, lecture notes, Multi-Agent Systems.

[4] A. Cermak. (2025, Nov.) Mars: Facts. NASA Science. [Online]. Available: https://science.nasa.gov/mars/facts/

[5] J. P. Fontanilles, “Lecture 3: Agent properties and types,” Lecture notes, Multi-Agent Systems, Universitat Rovira i Virgili, 2025, accessed: 2026-01-11.

[6] J. M. Bradshaw, “An introduction to software agents,” in *Software Agents*, J. M. Bradshaw, Ed. Cambridge, MA: AAAI Press/The MIT Press, 1997, ch. 1, pp. 3–46.

[7] G. Rigau, “Advanced techniques in artificial intelligence (curso 2021–2022),” Lecture notes, Universidad del País Vasco (UPV/EHU), 2021, consultado el: 11-01-2026. [Online]. Available: https://adimen.ehu.eus/~rigau/teaching/EHU/TAIA/Apunts/ATAI.1.pdf