Tecnológico de Monterrey

Evidence 2. Progress and Presentation of the Challenge

Review 2

Cesar Moran Macias | A01645209

Ana Paola Jiménez Sedano | A0164453

Ariana Guadalupe Rosales Villalobos | A01644773

Demmí Elizabeth Zepeda Rubio | A01709619

Luisa Merlo García | A01067715

Modeling of Multi-Agent Systems with Computer Graphics

Professor Iván Axel Dounce Nava

August 25th, 2025

Formal Challenge Proposal

Challenge Description

The challenge consists of designing and implementing a multi-agent system that enables Micro Aerial Vehicle (MAV) so that it can:

- Autonomously receive and interpret a mission
- Navigate to a designated area
- Identify a target person based on a description
- Land autonomously near the identified person within a 2 meter radius, without making content

The resultant agent is expected to successfully complete the mission as follows:

1) **Receive the mission:** The agent receives a text prompt with an instruction including a description of the person it's supposed to land next to and a close-by GPS location.
2) **Navigate autonomously:** The agent flies for at least 150 meters to the location of interest without human intervention.
3) **Search:** Once in the designed area the MAV initializes a search pattern, using its camera for scanning people. It must also analyze the collected data with the goal of identifying the person of interest described before.
4) **Perform final action:** Upon identifying the person of interest. The MAV must then perform an autonomous and safe landing 2 meters next to the target.

Type of Agents used & Responsibilities

The goal is that these drones are capable of performing tasks on their own with little to no intervention from the user and successfully completing the missions assigned. To achieve this, it is planned that each drone has **2 internal agents:**

1. **Agent responsible for computational vision:** This agent will perceive its environment through the MAV's camera to detect and recognize the characteristics of objects. It will then process this information to identify patterns and features that will be used to classify the object. The information, as well as the described characteristics of the person of interest, will be stored in its internal state, allowing its knowledge base to grow and enabling faster classification of the objects it observes. This will also facilitate the identification of the person of interest by comparing the description of the point of interest (such as clothing type and colors) with the actual person detected in the environment. By its characteristics and behavior, this agent can be classified as a model-based reflex agent because it makes decisions based on its processed percepts (the observed characteristics), its internal state, and its preexisting knowledge (such as its ontology and the description of the person of interest) in order to take action. This agent is also goal-oriented, since the actions it takes help it accomplish its main objective: finding the person.

2. **Agent responsible for arriving safely to the target:** To achieve this, it needs to be reactive (obstacle avoidance and flight safety) and proactive (following a search pattern to achieve the mission goal). Also, it needs to have an internal state where its position, search progress and detected obstacles are saved. This agent is

classified as goal oriented agent because it uses reactivity and proactivity to achieve its main goal, securely travel towards the target.

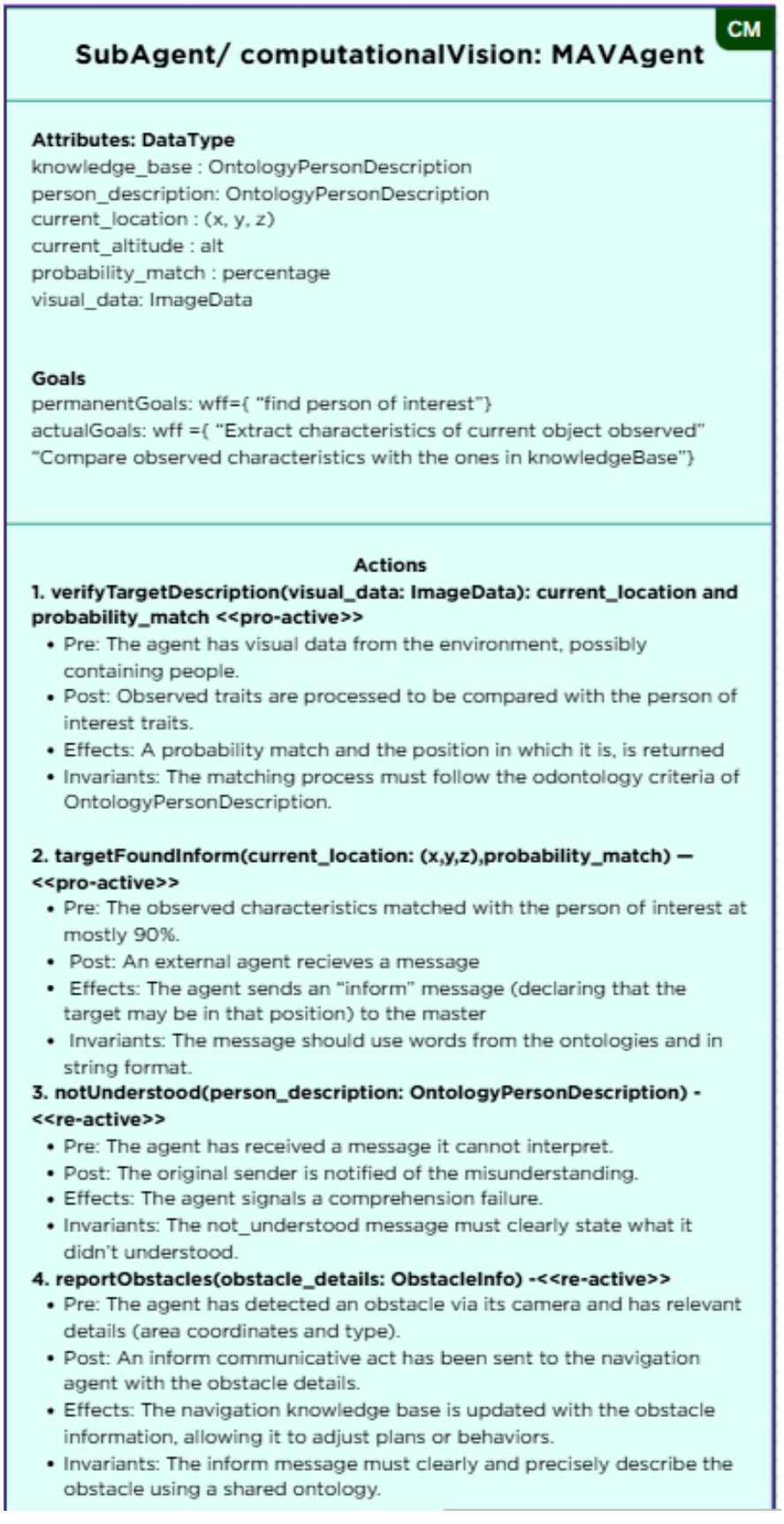
The **drones will be managed by an agent master, called SWARM master**, this agent will receive the mission definition in a text prompt and will communicate the contents of interest to the corresponding subagent. Also, it will delimitate the MAV's search areas (20 meter radius areas) and give landing permissions when the person of interest is found. This agent will be defined as a BDI agent because it will be the main agent that coordinates everyone and needs to make well thought decisions that consider a lot of data received from the other agents (it needs high social and coordinating abilities). Its beliefs would be the given mission, the drones statuses and their search areas and findings. The main desire would be finding the indicated person, but to achieve them, there would be more needed, such as dividing evenly the total area into sub search areas and giving landing orders. The intentions would be giving the directions to all the mav's to achieve the desires.

UML diagrams to explain how the simulation will work

We modified all our diagrams by making them more specific and in accordance with the specifications detailed in the article "Agent UML A Formalism for Specifying Multiagent Software Systems". The class diagram previously done in review 1 was separated into 3, one for describing its corresponding agent. Also, the protocol diagram was formatted in accordance to the message syntax communication between the agents.

Class Diagrams:

The following UML agent class diagram describes the agent in charge of computational vision:



Methods

1. interpretMission(person_description) → void

- Pre: Has recieved PersonOfInterest_description
- Post: Description is saved on knowledgeBase
- Effects: The MAV's knowledge base is updated with mission details.
- Invariants: The message is only in terms of the ontologies

2. scanAreaForPeople() → visual_data

- Pre: The MAV is in the designated search area.
- Post: The agent generates its percepts.
- Effects: The agent collects visual perception data about the current object.
- Invariants: The camera is configurated correctly (angle, altitude and color detection)

3. updateInternalState() → void

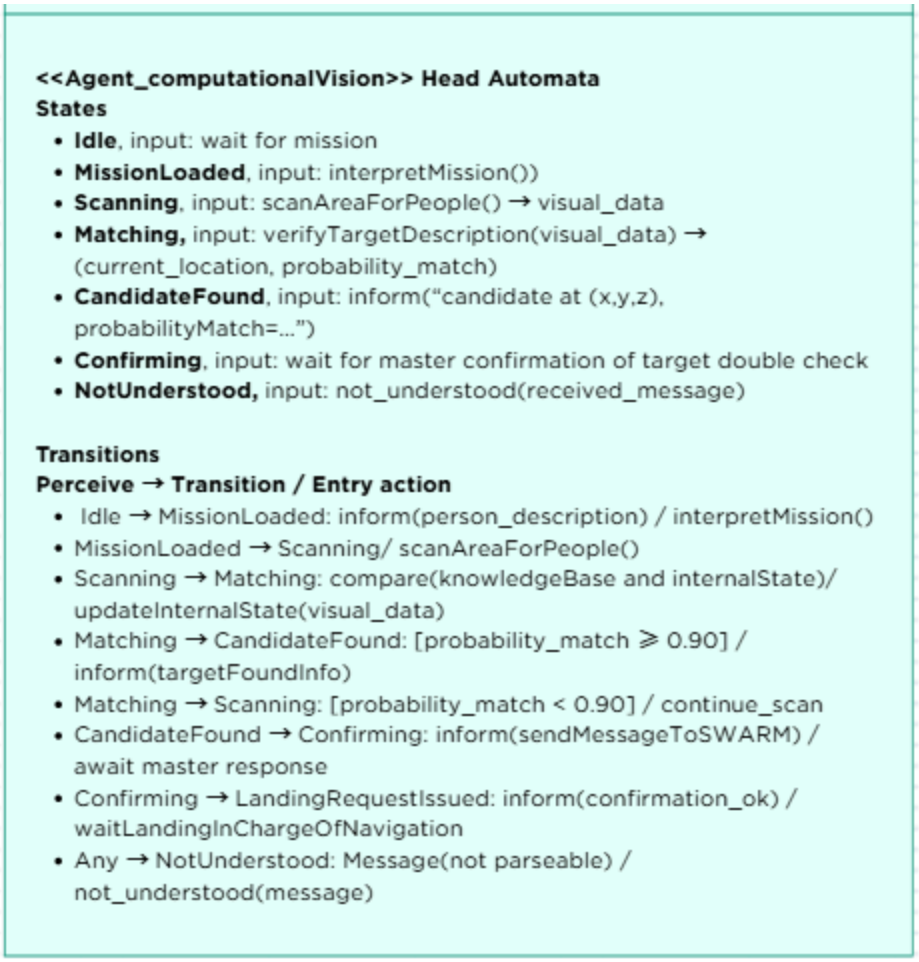
- Pre: The agent has received new perceptual inputs (traits observed) from the environment.
- Post: The agent's internal state incorporates the new perceptions.
- Effects: Internal state model is updated
- Invariants: The perceptual inputs need to be under the ontologyPersonDescription

Capabilities:

- Visual perception of the environment (substraction of traits from image)
- Receive the person of interest description from the SWARM master and save it in its knwoledge base in comprehensible terms
- Comparision of observed traits and desired traits
- Find the person of interest and report it to SWARM master
- Report obstacles to agent navigation

Constraints

1.- Society-name: PersonIdentificationProtocol
[constraint]: [confidence_score > 0.90]
2.-society-name: ObstacleReportingProtocol
[constraint]: [obstacle_type = 'obstructs_trajectory']
3.-society-name: SearchPersonOfInterestDescription
[constraint]: [gender = 'Feminine' AND clothing_top = 'Black blouse' AND clothing_bottom = 'White skirt']



Link:https://www.canva.com/design/DAGxOYVOrHc/w0ikltmwy05zv7GuBYoqlA/edit?utm_content=DAGxOYVOrHc&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

The following UML agent class diagram describes the agent in charge of navigation:

AV

<<Navigation_Agent>

Attributes: DataType
knowledgeBase : OntologyNavigation
current_location : (x, y, z)
target_location:(x,y,z)
Searchpattern:
current_altitude : alt
current_velocity: m/s
current_position: Coordinates
Search_State: Active, stopped, paused
flight_state: (Onroute, search, align, land)

Permanent goals: wff = {"Navigate until you find the target person", "Avoid obstacles reactively", "Update status and critical events", "Register current state", "Communicate actively with the system"
}

actualGoals: wff ={ "Sets the course when the target is found", "Ascend", "Search with a pattern strategy", "Request for alignation within the target", "Confirms the absence of obstacles", "Adjustments for descent", "Validates the safe distance between the drone and the target
"
}

Actions

StartSearch <Re-active> (area: Polygon, pattern: SearchPattern)

Pre: The target area is near the drone's arrival.
Post: A grid, spiral, or zigzag coverage plan is instantiated across the region.
Effects: The agent begins executing patterns and enters the "Search" stage.

Setcourse (gps_target) <Re-active>
Pre: The mission's destination has been announced by the swarm master.
Post: A path is created in the direction of the destination.
Effects: The agent updates the current state after switching to the "On Route" state.

Executes_PatternSearch(waypoint: Vector3) <Proactive>

Pre: The next waypoint is defined and a search pattern is in effect.
Post: The MAV collects visual data as it approaches the waypoint.
Effects: Perception-related new poses and frames are created, and the coverage percentage is modified.
Invariants: Vertical and lateral velocity cannot be higher than permitted.

reportObstacles(obstacle_details: ObstacleInfo)

Pre: Using sensors and a camera, the agent has identified an obstacle and has the necessary information (area coordinates, type, extent).
Post: The impediment specifics are given along with an informative communicative act.
Effects: Plans and actions can be modified thanks to the enhanced navigation knowledge base.

VerifyTarget_Description <pro-active> (visual_data: ImageData): probability_match and current_location

Pre: The agent has visual information from the surroundings, which may include human beings.
Post: The person-of-interest qualities are compared with the observed traits after processing.
Effects: The candidate's estimated position and a probability match are returned.
Invariants: The matching procedure needs to adhere to the OntologyPersonDescription criteria.

TargetFoundInform(current_location: (x,y,z), probability_match) <pro-active>

Pre: There was a chance of ⩾ 0.90 that the observed features matched the person of interest.
Post: A message is sent to an external agent.
Effects: The agent notifies the swar, master that the target might be in that location by sending a "inform" message.

startLandingSequence <re-active> (approved: LandingZone)

Pre: The landing has been confirmed by the Swarm master.
Post: We follow the protocol of landing Align → Validate → Descent.
Effects: The agent starts precise landing actions after exiting search mode.
Invariants: Checks and sequence order must be adhered to The Landing Protocol.

Replan_Landing(offset: Vector2) <re-active>

Pre: The master prevented or rejected a landing attempt.
Post: Using the given offset or limitations, a new landing zone is calculated. The result is a new *LandingPoint*.
Invariants: Unsafe zones that have already been rejected cannot be reused during the replan.

Methods

Compute_Route_Target(gps_target) → WaypointList
Pre: A destination for the mission has been determined.
Post: From the present stance to the target area, an efficient waypoint sequence is generated.
Effects: Speed and altitude profiles are used to build the navigation plan.
Invariants: Routes adhere to no-fly zones and geofences.

Generate_SearchPattern → WaypointList (area: Polygon, pattern: SearchPattern)

Pre: Near the search area, the MAV has arrived.
Post: Area-wide waypoints (grid, spiral, and zigzag) are created.
Effect: Using the ground sampling distance and field of view as parameters, a coverage plan is instantiated.

EvaluateHypothesis → MatchScore

Pre: For the current frame, a detection hypothesis is present.
Post: The person description is compared to a similarity score.
Effects: A single probability or score is created by combining matching features.
Invariants: weighting and thresholds.

ValidateLandingZone → SafetyCheck

Pre: There is a landing zone for candidates.
Post: For the following reasons (slope, obstacle density, and person distance), SafetyCheck returns OK/Reject.
Effects: The landing zone is either approved or denied for descent.

computeDescentProfile → DescentPlan (LandingZone)

Pre: SafetyCheck=OK is present in the landing zone.
Post: Hold points and a vertical speed profile are generated.
Effects: The restrictions are defined in the descent plan.

Execute_descent -> DescentStatus (Landingprotocol)

Pre: Alignment requirements are satisfied and safety is acceptable.
Post: Altitude is decreased as planned.
Effects: The MAV periodically performs safety checks as it approaches to the landing zone.

Handle_Abort → (reason: AbortReason)

Pre: There is a violation of a critical feature.
Post: Failure is reported once the MAV reaches a safe height and holds.
Effects: A fresh decision is awaited and current activities are posponed.

Capabilities :

- Visual perception of the surroundings.
- Get the SWARM master's description of a Person-of-Interest and enter it into the knowledge base.
- Determine a match score by comparing intended and observed qualities of the target.
- Notify the SWARM master of any candidate sightings and locate the Person-of-Interest.
- Identify, categorize, and report obstacles; then, adjust the navigation map appropriately.
- Create and follow routes to a GPS target; use search patterns.
- Perform a controlled descent after aligning over the authorized landing area.

Constraints

- society-name: ReplanProtocol [constraint]: [reused_blocked_landingzone = false]
- LandingProtocol [constraint]: [distance_to_person_m >= 1.0 AND distance_to_person_m <= 2.0]
- society-name: ObstacleReportingProtocol [constraint]: [obstacle_type = obstructs_trajectory]

<Agent_Navigation> Head Automaton

States

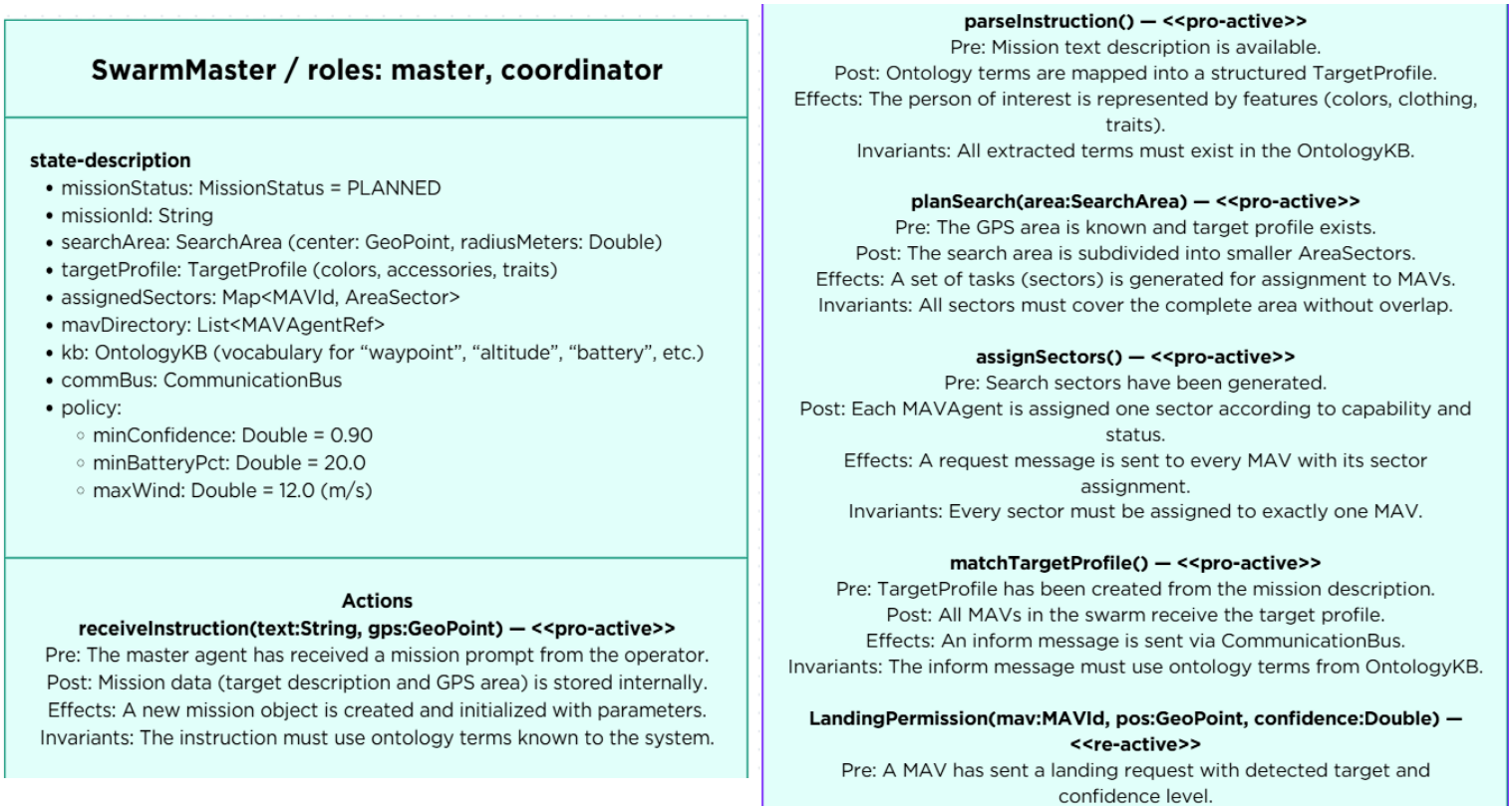
- MissionLoaded, input: interpretMission(person_description)
- Scanning, input: scanAreaForPeople() → visual_data
- Matching, input: verifyTargetDescription(visual_data) → (current_location, probability_match)
- CandidateFound, input: targetFoundInform(current_location, probability_match)
- Confirming, input: wait for master confirmation
- LandingRequestIssued, input: startLandingSequence(approved_lz)

Transitions

- Idle → MissionLoaded: receive(mission(person_description)) / interpretMission()
- Matching → TargetPersonFound: [probability_match ⩾ 0.90] / targetFoundInform(current_location, confidence)
- Scanning → Matching: [new_frame / updateState(visual_data)
- CandidateFound→ Confirming: inform(sendMessageToSWARM(candidate))

Link:https://www.canva.com/design/DAGwEI5qH10/5XCN5aNRyO9TP9JdMVwN0A/edit?utm_content=DAGwEI5qH10&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

The following diagram describes the Swarm master agent



Invariants: Only ontology-approved labels are used in the profile.

buildSearchArea(center:GeoPoint, radiusMeters:Double) → SearchArea
Pre: A GPS center and radius are defined.
Post: A SearchArea object is created for the mission.
Effects: The mission now has a geographic scope.
Invariants: The area geometry must be valid and expressed in mission CRS.

subdivideArea(area:SearchArea, n:Int) → List<AreaSector>
Pre: The SearchArea exists and n ⩾ 1.
Post: The area is split into n (or more) non-overlapping sectors.
Effects: Sector list becomes available for assignment to MAVs.
Invariants: Sectors must fully cover the area with no gaps or overlaps.

chooseMAV(sector:AreaSector) → MAVId
Pre: At least one MAV is available and the sector is ready to assign.
Post: One MAV is selected as the assignee.
Effects: Selection considers distance, battery, role, and load.
Invariants: The chosen MAV must meet minimum battery and role requirements.

assignSector(mav:MAVId, sector:AreaSector) → void
Pre: A MAV was selected for this sector.
Post: The MAV receives an assignment message for that sector.
Effects: The task map (MAV → sector) is updated.
Invariants: Each sector has exactly one active assignee.

broadcastTargetProfile(profile:TargetProfile) → void
Pre: TargetProfile is available.
Post: All MAVs receive an "inform" with the profile.
Effects: Shared understanding of the person of interest is established.

Post: The request is either approved or denied.
Effects: The master agent sends an inform message with the decision.
Invariants: Permission is only granted if confidence ⩾ 90% and safety conditions hold.

updateFromReport(msg:Message) — <<re-active>>
Pre: A MAV has sent a status report (battery, obstacles, progress).
Post: Mission state is updated with the new information.
Effects: The knowledge base is refreshed and tasks may be reassigned.
Invariants: Report content must comply with the shared ontology schema.

reassignTasks(failedMAV:MAVId) — <<re-active>>
Pre: A MAV has failed or reported low battery.
Post: Its sector is given to another available MAV.
Effects: A new request message is sent to the reassigned MAV.
Invariants: No sector remains unassigned while mission is in progress.

Methods

interpretMission(person_description:String, gps:GeoPoint) → void
Pre: The master has received the mission prompt (text + GPS).
Post: The description and GPS are stored in the mission record.
Effects: The knowledge base is updated with the mission fields.
Invariants: All terms must belong to the shared ontology.

mapTextToProfile(person_description:String) → TargetProfile
Pre: A valid person description is available.
Post: A structured TargetProfile is created (colors, clothing, traits).
Effects: Free text is translated into feature keys used by other agents.

Service Description

- Acts as a centralized coordinator in the swarm.
- Ensures consistency across knowledge bases.
- Provides each MAV with its specific sub-area to search.
- Broadcasts the landing permission when target is confirmed.
- Collects reports of obstacles, misunderstandings, and detected targets.
- Validates messages based on ontological structure and expected protocols.

Supported Protocols

- **MissionDistributionProtocol:**
Used to send the mission to each MAV agent (GPS + Person Description).

- **TargetConfirmationProtocol**
Used by MAVs to inform the master about potential matches.
- **ObstacleReportingProtocol**
- Used to notify the master about obstacles that interfere with flight.

- **TaskAssignmentProtocol**
- Used to assign or reassign search zones.

- **LandingPermissionProtocol**
- Used when a MAV requests authorization to land near the identified target.

recordReport(msg:Message) → void
Pre: A well-formed report message was received.
Post: Mission state is updated (battery, position, obstacles, progress).
Effects: Triggers checks for replanning or reassignment if needed.
Invariants: content fields match the shared message schema.

evaluateLanding(confidence:Double, safety:Bool) → Boolean
Pre: A MAV requested landing with a confidence estimate and safety flag.
Post: A decision (true/false) is returned.
Effects: Decision will be used to send permission or denial.
Invariants: Approve only if confidence ⩾ 0.90 and safety = true.

summarizeMission() → Report
Pre: Mission has progress data or has ended.
Post: A concise report is produced (assignments, findings, outcomes).
Effects: Final artifacts are stored for review.
Invariants: Report fields follow the mission reporting template.

Capabilities

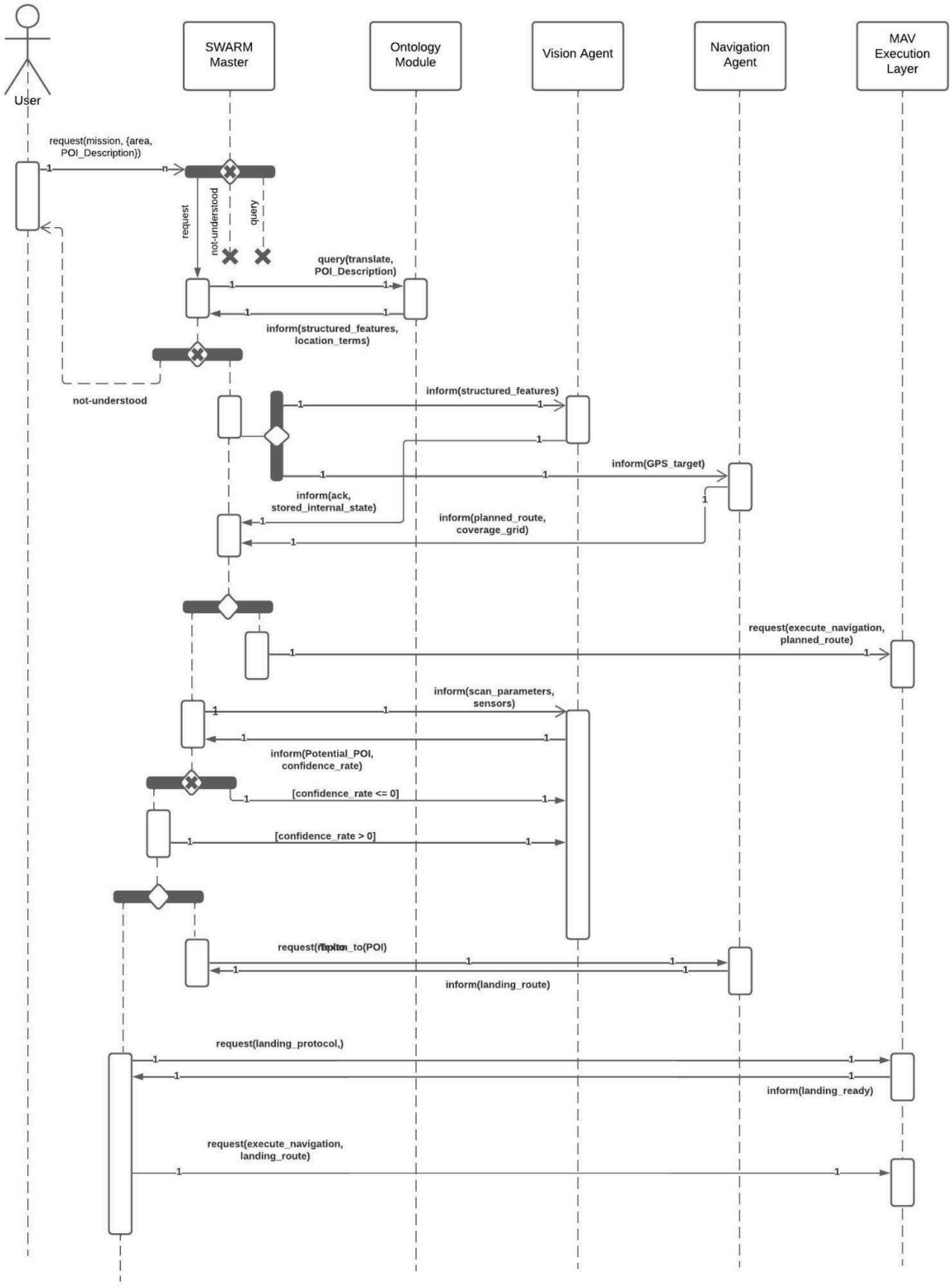
- Receive and interpret the mission (description of the person + GPS location).
- Decompose the mission into subgoals and assign them to each MAV agent.
- Coordinate communication between agents using shared ontologies.
- Maintain an internal map of agent roles, areas, and progress.
- Evaluate confirmation messages from agents and validate landing decisions.

Link:

https://www.canva.com/design/DAGxPRmXd5g/9k1Hfi9h1iusP9kraIrFfQ/edit?utm_content=DAGxPRmXd5g&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Sequence Diagram:

The following diagram is a Sequence UML diagram of the MAVs simulation: Link: https://lucid.app/lucidchart/b548548d-5ed9-4e3d-9252-73ad642df26e/edit?viewport_loc=-1275%2C-129%2C3975%2C1431%2Cjw3Kh2H_Hzd6&invitationId=inv_888c28c9-52a7-4f4e-a531-66e804fd6cf9


User
SWARM Master
Ontology Module
Vision Agent
Navigation Agent
MAV Execution Layer
request(mission, {area, POI_Description})
request
not-understood
query
query(translate, POI_Description)
inform(structured_features, location_terms)
not-understood
inform(structured_features)
inform(GPS_target)
inform(ack, stored_internal_state)
inform(planned_route, coverage_grid)
request(execute_navigation, planned_route)
inform(scan_parameters, sensors)
inform(Potential_POI, confidence_rate)
[confidence_rate <= 0]
[confidence_rate > 0]
request(navigate_to(POI)
inform(landing_route)
request(landing_protocol,)
inform(landing_ready)
request(execute_navigation, landing_route)

For the MAV agents (you can use any diagram or image as support):

1. **Short description of how the MAVs will receive an instruction and interpret it.**

As stated before, the ultimate mission of the MAVs is to help each other and work together in accomplishing the safe landing next to a person described by a text prompt that contains both, the description of the target person and an approximate search area with a GPS coordinate.

This input or instruction will be first received by the swarm master agent, which acts as the coordinator. The text description is mapped into a structures representation using the MAV's internal ontology and knowledge base, so that it can interpret terms of the description such as colors or clothing types and translate them into detectable features for the computer vision agent. There will be general ontologies that will be shared by all the agents: one with the terms necessary to describe the looks of a person (called ontologyPersonDescription) and another with terms necessary to describe a flight trajectory (called navigationOntology). These ontologies, along with the perceptions, will feed the knowledge bases of the corresponding agent. Meanwhile the GPS coordinate is interpreted by the navigation agent as the goal search area.
This structure and interpretation of the instruction becomes a set of goals and constraints, where the perception agent stores visual characteristics that define the person of interest, the navigation agent plans the route towards the designated GPS location, and the swarm master coordinates both and ensures the alignment of the goals for the accomplishment of the mission. This way the MAVs can transform the text prompt into executable actions and drive the mission autonomously.

2. **Short description of the main strategy for the MAVs to navigate to a specific GPS location.**

The main strategy for the MAVs to go to a GPS location is guided by the navigation agent, which works with goals, first, the MAV receives the GPS target and compares it with its current position from its sensors, then the navigation agent creates a flight plan with waypoints that move step by step until the MAV arrives at the destination. The plan follows a goal-based strategy, where each action, like moving forward, turning, or changing altitude, is chosen because it reduces the distance to the target, and at the same time, the agent uses a utility strategy, which means it also thinks about safety, battery use, obstacles, and flight stability, so in this way, the path is not fixed but can change in real time if conditions change.

The MAV uses an ontology, which is like a small dictionary of words and concepts it understands, for example, it knows what “GPS coordinate,” “waypoint,” “altitude,” or “battery level” mean, and this knowledge helps it interpret the mission instructions and transform them into actions that it can execute during the flight. Communication is also part of the strategy, when MAVs need to cooperate, they send messages with two main parts: the performative (type of message, like inform or request) and the content (the information). They can also include sender, receiver, ontology, or time, for example, a MAV can send information: sector completed or request: landing permission, this ensures all MAVs and the swarm master share information in the same way and avoid confusion.

The strategy includes constant monitoring and adaptation, the MAV checks its position, updates its flight plan, and sends or receives messages when needed, if there are problems like low

battery, obstacles, or bad weather, the MAV changes its actions but always keeps the same final goal which is reaching the GPS location safely and efficiently.

3. **Short description of the main strategy for the MAVs to identify specific characteristics from a person (assume you have an "imperfect" AI camera (that means, there is a small probability of wrong recognition), a view from above, and a certain flight altitude).**

The computational vision agent will receive a message from the swarm master informing the visual description of the person of interest. This message would be of type inform and it will pass the ontology (either one provided by NuclearSolutions or one form internet or one designed by us), and as content it will describe the characteristics of the person of interest using the classes, properties and terminology of the ontology.

One informal example of the content (in the implementation the message will be in a JSON) could be:

Person of interest has:

- Person: Yes
- Genre: Feminine
- Clothing top: Black blouse
- Clothing bottom: White skirt
- Hair: Long straight black hair
- Hand accessories: White bag
- Head accessories: Brown hat

This description will be saved on the agent's knowledge base. After that, the agent will start perceiving its environment through the camera. From what it detects at a specific moment, it will start recognizing the characteristics defined in the ontology such as if it is a person, genre,superior clothing, etc and it will save them on the agent's internal state. After having a complete description of the object in that moment, it will compare it with the characteristics in the knowledge base (these are the ones from the person of interest) and give a percentage of probability of that being the person of interest. If it isn't more than 90%, it will conclude that it isn't the target and continue looking. However, if it has a great coincidence, it will send a message (of type inform) to the SWARM master saying "Person of Interest found". After receiving it, the master will send a message of type request to another MAV saying "go to (coordinates) possible person of interest found" (to double check if it is truly the person of interest). If this drone confirms it as well through a message of type inform sent to the master that says "Person of Interest correct", the master will send a request message to the original MAV that says "stop looking, let agent navigation land". Then the master will send a request message to the navigation agent of the original MAV saying "Land safely".

4. **Short description of the main strategy for the MAVs to do the landing near the person.**

In order to correctly plan the MAV landing stage, it is essential that the previous protocols for identifying the target person have been successfully executed.

These protocols include:

- The prompt-style description of the target person that will be received by the swarm master to configure the drone's search.
- The search patterns executed by the navigation agent, previously designed to efficiently cover the area of interest.
- The continuous processing of the perception agent, which uses computer vision to scan the area and detect possible matches with the given description.
- The information collected by these agents is sent to the Swarm Master, who receives the results through communication protocols between agents, using ontologies and the ACL (Agent Communication Language).

The Swarm Master is responsible for:

- Confirming the identification of the target person by comparing the detected characteristics with the textual description that initiated the MAV mission.
- In the event of a positive match, issuing a request to the navigation agent to initiate the landing protocol.

The landing protocol involves:

- Adjusting the drone's dynamic parameters: speed, position, and altitude.
- Following safety measures previously defined as preconditions for landing.

In addition, agents need to be responsive to unforeseen changes or situations outside their standard configuration, using replanning strategies so that their operation is not limited by constant changes in the environment, this reinforces their autonomy.

The ontology that follows this protocol is defined by :

Landing_protocol:

Swarm agent -> Targetperson= 1 && confidence= high; request (Landing_protocol) to NavigationAgent

Landin_zone:coordinates

Altitude:int

Velocity:int

Safety_radius:int

ObstacleDensity:int

Confidence_threshold: int

Landing State

Current_location: Coordinates

Battery_level:int

Distance_person:int

Search_State: Active, stopped, paused

5. **Short description of the main strategy for the MAVs to communicate and collaborate to find a person.**

The strategy for the MAVs to communicate and collaborate to find a person centers around a SWARM master agent which will organize the mission; as it is mentioned in Review 1, this master agent designates to individual drones specific search areas to ensure coverage of the environment. Once the target person is identified, the master agent will give landing permissions.

Since each individual drone within the MAV system is equipped with two internal agents, they have different responsibilities for the mission success. One is the vision agent, tasked with detection and recognition of specific characteristics of the target person, like color, and is highly important for the search with the MAV scanning individuals using the camera and analyzing data collected. The other agent is responsible for a safe flight, making sure that the drone navigates through the environment without incident and also performs a safe landing, all of it according to the conditions given by the challenge situation.

The communication and collaboration occur in key steps:

- The SWARM master initiates the mission by assigning a mission message to MAVs, containing the search area coordinates and target person description.
- MAVs then autonomously navigate to their assigned areas and perform local searches using their vision agents.
- Upon identifying a potential target, a MAV sends a report of potential target message to the SWARM Master, including the pointed location and the characteristics of the potential target.
- The SWARM Master verifies the sighting and, once confirmed, issues an authorized landing message to the designated MAV, specifying the coordinates

and the required proximity (which will be 2m) with safety parameters of ‘no contact’.

Work Plan & Acquired Learning

Plan for next review

For the next iteration the main pending activities and their responsibles are the following:

Task	Responsible	Due Date	Estimated Effort
Finalize corrections of UML diagrams and interaction protocols.	Ariana & Luisa	28/08/2025	2 hours
Advancement on the code for the implementation of the visual agent	César	29/08/2025	3 hours
Advancement on the code for the implementation of the navigation agent	Ana	29/08/2025	3 hours
Advancement on the code for implementing the graphical solution	Demmí	29/08/2025	3 hours

Pending tasks

Task	Responsible	Dates	Effort
Begin the creation of the components in blender	Demmí	28/08/2025	2 hours
Join python with unity for simulations	Everyone	28/08/2025	1 hour

Investigation of next stage	Ana & Ariana	29/08/2025	1 hour
Documentation	César & Luisa	29/08/2025	2 hours
Correction of professor's notes	Luisa	30/08/2025	1 hour

Acquired Learning

As a team we understand the basic theory of MAS, especially the different types of agents and how they can communicate with each other using simple messages or speech acts.

We also practiced with the idea of agents having roles and responsibilities, which helps us plan the challenge in a more organized way. For the practical side, we modeled some elements in Blender like a basic classroom and a rocket, even if these models are simple, they help us learn how to build the environment that the agents will use later in the simulation, so this gave us experience in creating objects that can be reused when we design the scenario for the final project.