INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY,
CAMPUS GUADALAJARA

Tecnológico de Monterrey

# Evidence 2 Intermediate Delivery

By

Diego Rodríguez Romero A01741413

Diego Alejandro Calvario Aceves, A01642806

Milan Alejandro De Alba Gómez A01637667

Gonzalo Calderón Gil Leyva A01740008

Modeling of Multi-Agent Systems with Computer Graphics, Gpo 301

Prof. Ivan Axel Dounce Nava

November 23rd 2024

Evidence 2

**Description of the problem**

Coordinated patrolling is a big challenge in a lot of aspects of security, special in urban areas. In that matter, patrolling is a coordinated effort between security personnel, cameras, and most recently, thanks to technological advances, drones. The effective coordination among these agents is necessary to detect suspicious activity in the area and solve any emergency. All that said, it is clear that an effective coordination and communication between all the actors in charge of the surveillance of an area is crucial, to ensure that we can use autonomous agents to control the drones and cameras, so they can interact with the environment and the security personnel to effectively answer to any threat, without solely relying only on humans.

This isn't a problem of a single area, it can be, in fact, applied to a lot of different areas of surveillance, for example a warehouse, a construction site, an agricultural zone or a factory. The solution of using drones as autonomous agents takes advantage of both the technology available and the multiagent systems theory.

**Agents Involved**

For the performed simulation there are 4 different agents involved, the drone, the cameras, the security personnel and the robbers (or threats). In a real world situation the drone and the cameras would be the only agents, because the security personnel and the possible threats would end up being human beings. But, talking about only about the simulation here is a short description of each agent:

-Drone: Its primary roles are patrolling, detecting and alerting of threats, even chasing them. Some of its key characteristics are that they can take off, fly and land, be controlled by security personnel, detect robbers and chase them.

-Camera: Its function is to detect movements and monitor the area. They are at a random location, have a detection radius, and can alert drones about suspicious activity.

-Security Personnel: They patrol, chase robbers and can control drones. Can move around, chase threats that the drones alert them off.

-Robbers: Move through the environment trying to avoid detection.

The agents have different relationships and interaction with each other, fo example:

-Cameras can detect and alert drones about robbers
-Drones can chase and track robbers
-Security personnel can take control of drones
-Security personnel can catch robbers
-Robbers try to evade detection by all other agents

**Agent Properties**

Even though each agent has a special set of properties that makes them unique, they also have common properties. For example, all agents have a position in the grid, they have a way to handle collisions, have detection ranges, have a height property (can be fixed, changing or random depending on the agent) and they all have different values that represent some sort of status. Now, talking about specific properties:

1.- Drone
Position:
target_height: Desired flying height (default: 5)
landing_station: Base coordinates
Status:
is_flying: Boolean for flight status
is_controlled_by_security: Boolean for security control (when security takes over control of the drone)
mission_completed: Boolean for mission status (if the number of steps has been reached and the drone has to return to the landing station)
Detection:
detection_range: Range for spotting robbers or other agents (3 units)
collision_threshold: Minimum distance to avoid collisions (1 unit)
Target tracking:
chasing_robber: Boolean for chase status
target_robber: Reference to tracked robber

2.- Cameras

Position:

height: Random height between 3-8 units (this is where the camera is going to be located throughout the whole simulation)

Detection:

detection_radius: Range for spotting movement (3 units)

last_detection_time: Timestamp of last detection

detected_agents: Set of detected agent IDs

3.- Security Personnel:

Status:

has_drone_control: Boolean for drone control

status chasing_robber: Boolean for chase status

alerted_to_robber: Boolean for alert status

Movement:

movement_speed: Speed of movement (1 unit)

collision_threshold: Minimum distance to avoid collisions (1 unit)

Target:

target_robber: Reference to tracked robber

4.- Robbers:

Status:

is_caught: Boolean for capture status

is_spotted: Boolean for detection status

Detection:

detection_range: Range for spotting threats (3 units)

collision_threshold: Minimum distance to avoid collisions (1 unit)

**Environment Properties**

The environment in this case is the place where our whole simulation is happening. We chose our environment to be a warehouse, even though it could be replicated to be any of the other places talked about in the problem description. The properties of our environment are:

Space and Grid:

grid_size: the size of the surveillance area (20 x 20 units in our case)

landing_station: fixed point for the drone to take off and land (10,10 for our simulation

visualization: We give every cell a value that represents what is going on in that cell, for example an empty space, the drone is there, a camera, etc.

Agent Management:

Agents Collections: The environment has a list of all the agents that exist in the specific simulation

Simulation Control:

steps: Control the current step number and state of the simulation

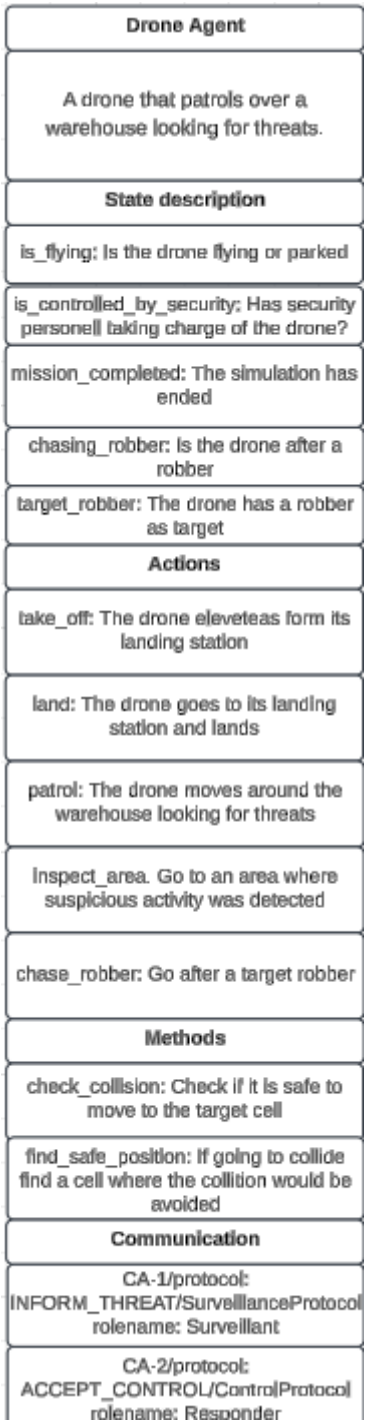
parameters: The values with which the simulation is initiated

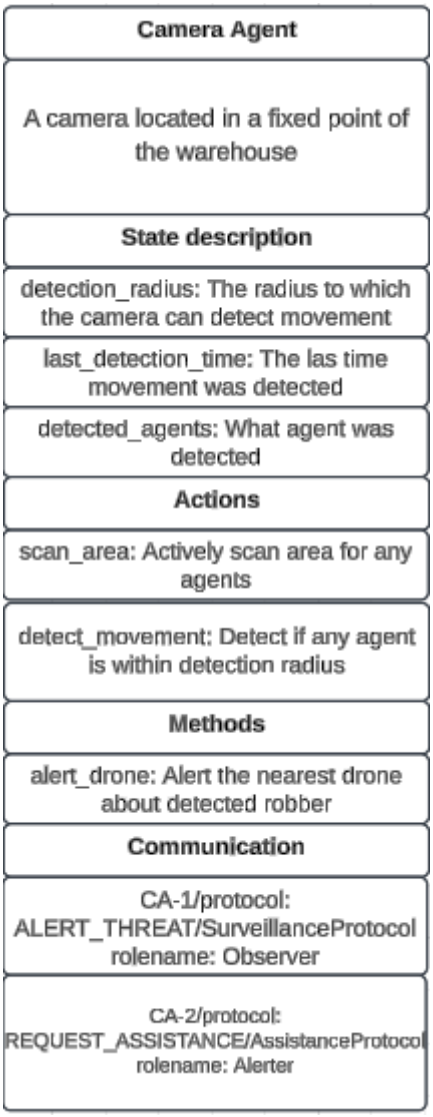
Visualization:

Includes all the plot configuration, grid display, etc. (only for our python simulation)

## Class Diagrams

Link a diagramas:Link



**Security Personnel Agent**

A security person that goes around the warehouse looking for threats.

**State description**

has_drone_control: Is it currently controlling a drone

chasing_robber: Is it chasing a robber

target_robber: The robber which it is chasinf if any

alerted_to_robber: Has it been alerted of a robber by a drone or camera

alerted_to_robber: Has it been alerted of a robber by a drone or camera

movement_speed: How fast is it moving at any given time

**Actions**

patrol: Patrol the area randomly while avoiding collisions

chase_robber: Chase and try to catch the robber with increased urgency

assess_threat: Assess if drone's area inspection reveals a threat

**Methods**

take_drone_control: Take control of a drone

release_drone_control: Release control of a drone

check_collision: Check if moving to new position would cause collision

**Communication**

CA-1/protocol: REQUEST_CONTROL/ControlProtocol
rolename: Controller

CA-2/protocol: COORDINATE_PURSUIT/PursuitProtocol
rolename: Pursuer

**Robber Agent**

A robber who tries to sneek in the warehouse and not be detected

**State description**

is_caught: Has it been caught by another agent

is_spotted: Has another agent detected it

**Actions**

move: Move while trying to avoid detection and collisions

evade_threats: Move away from detected threats

**Methods**

check_nearby_threats: Check for nearby drones and security personnel

check_collisions: Check if moving to new position would cause collision

**Communication**

CA-1/protocol: None
rolename: Target

**Utility/Success Measurement**

There are several ways that we could measure the utility or success of our Drone agent, that is the main part of our simulation, for example we could use detection effectiveness, its patrol performance or its pursuit success. We we will focus only on three metrics:

-Threat Detection Rate: That is the number of robbers successfully spotted by the drone over the total number of robbers in the simulation. The success criteria here is that higher is better, if we have 1 rubber and the drone spots it then we have a 100% success.

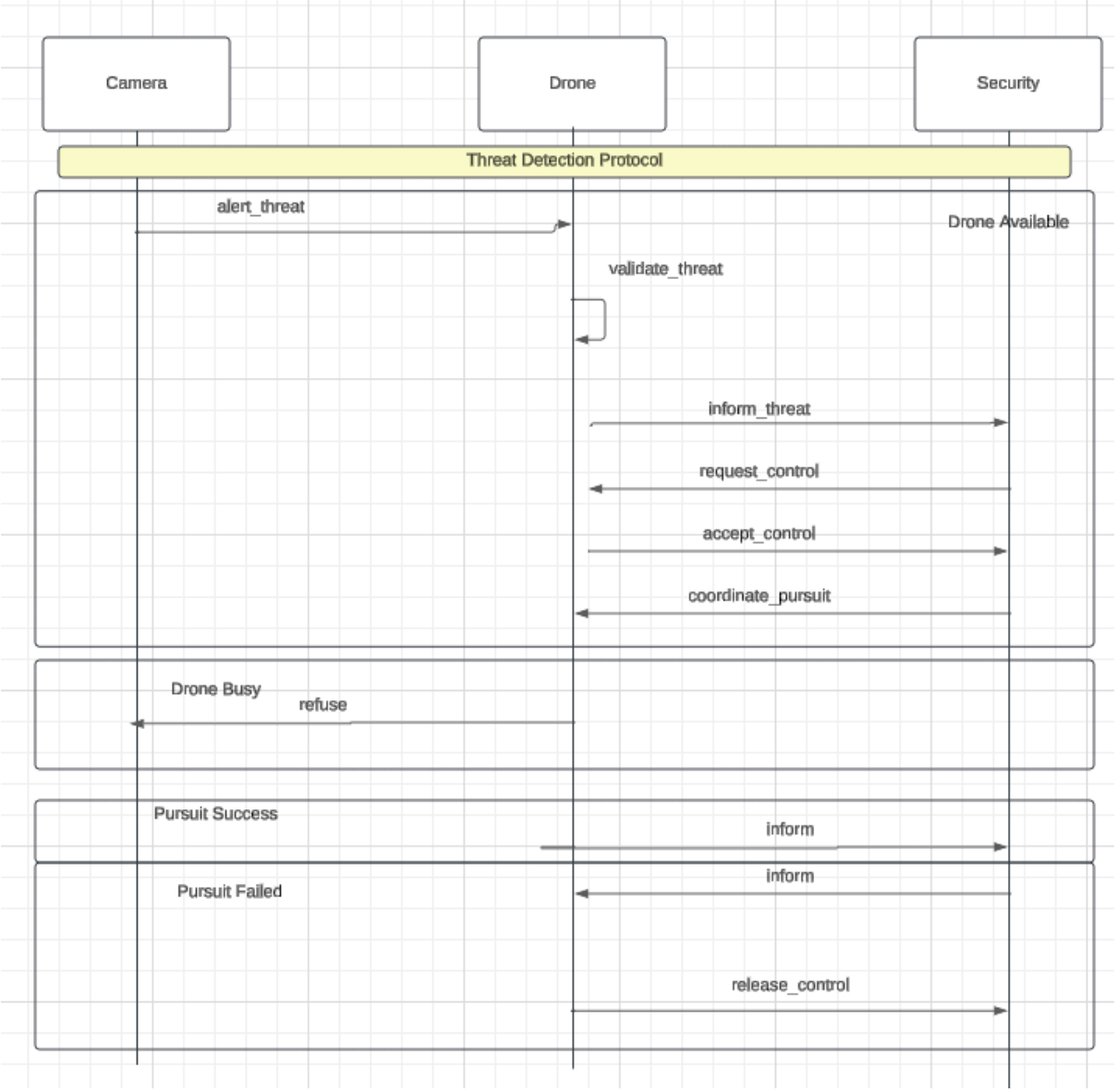
-Area Coverage: The percentage of the total grid that was surveilled by the drone during the patrolling. Here also, higher is better, our grid is 20x20 giving us 400 total cells, the higher number of cells we visit gives us more area.

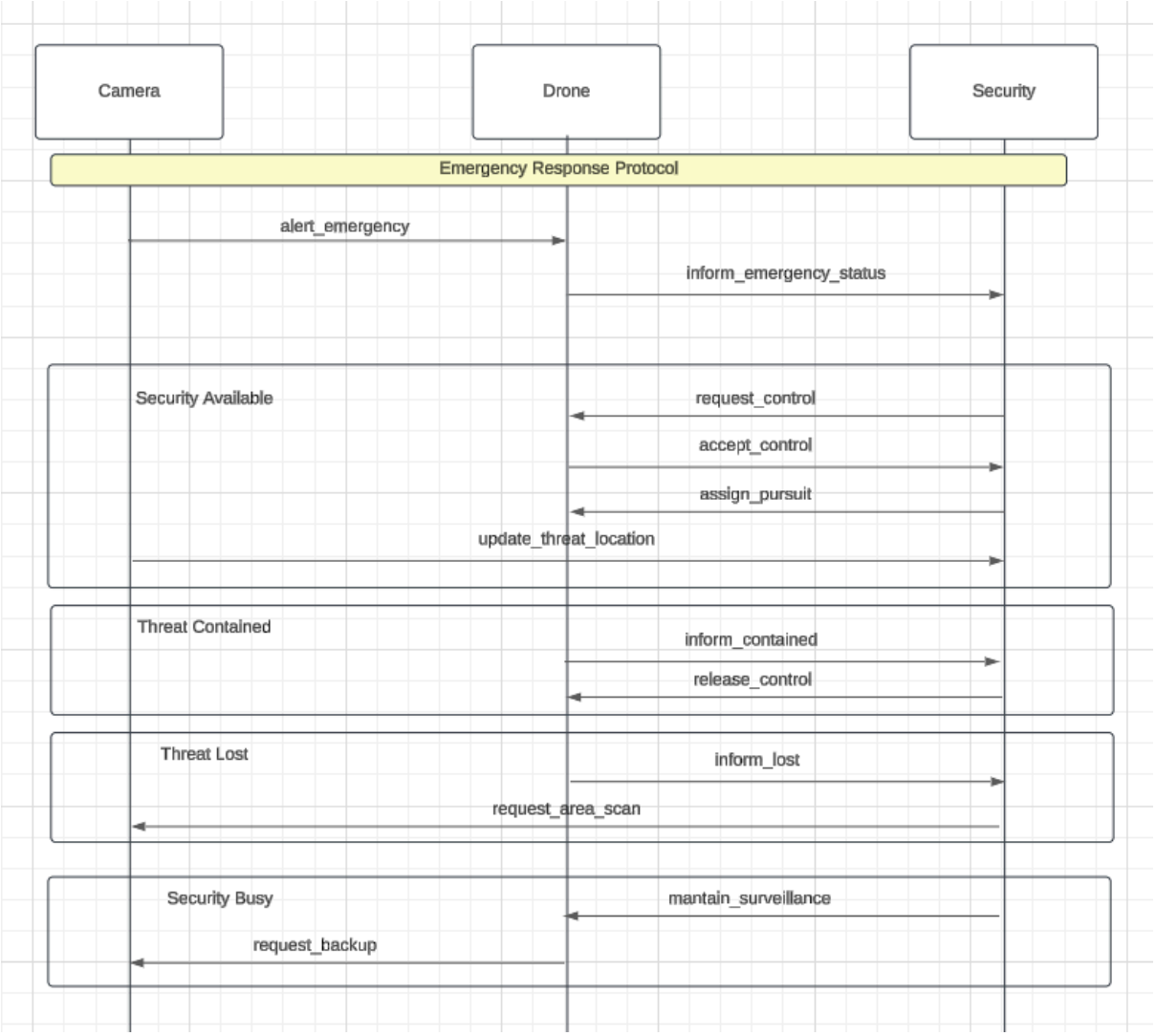
-Chase Success Rate: How often the drone tracked a spotted robber. Successfully chasing the robber means that the drone follows it until it is caught by security personnel.

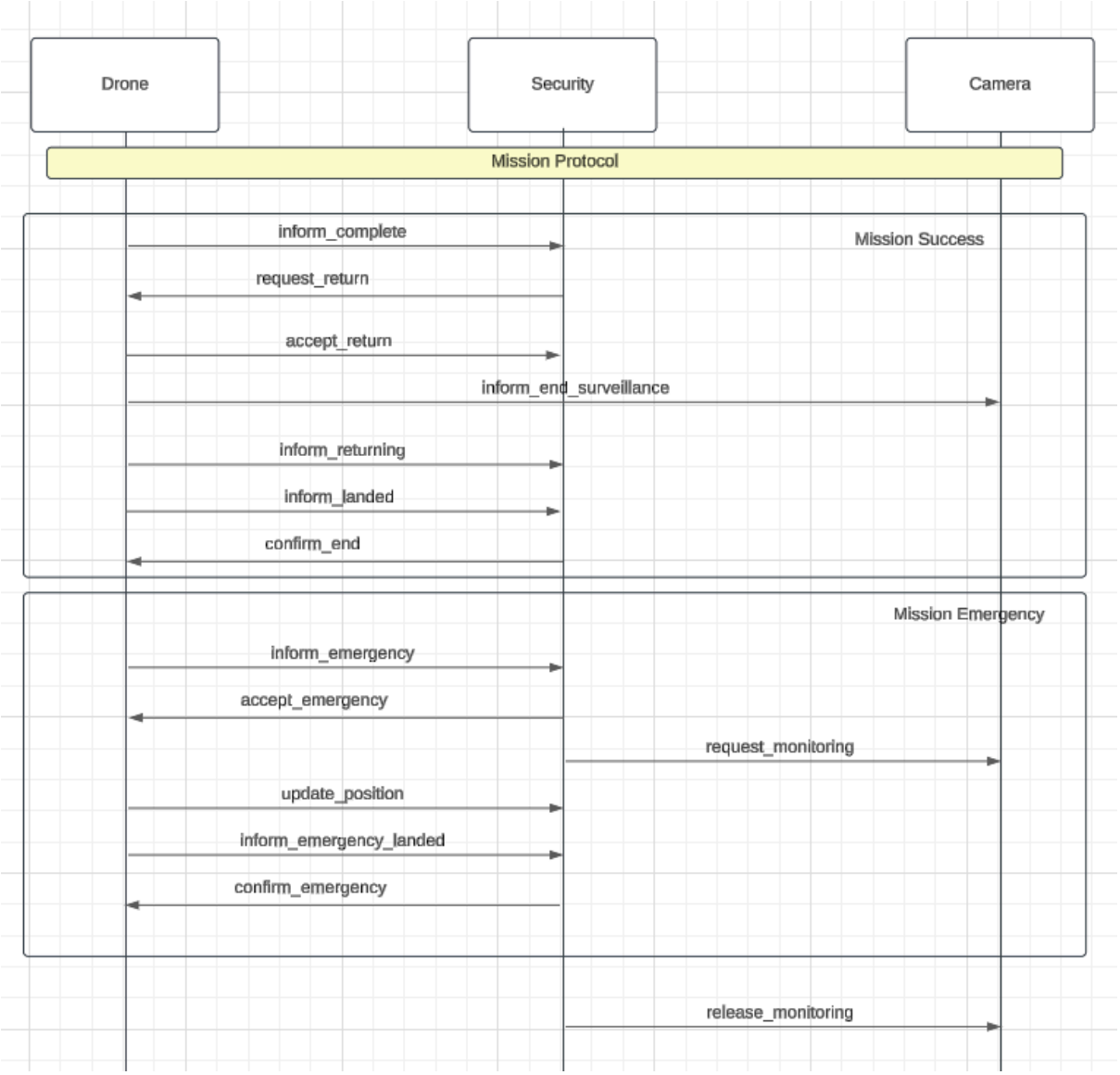
To combine these 2 metrics to define the overall success of the drone we can give them weights of 40%,30% and 30% respectively to get the success by the following formula Success = (0.4 * TDR) + (0.3 * AC) + (0.3 * CSR).

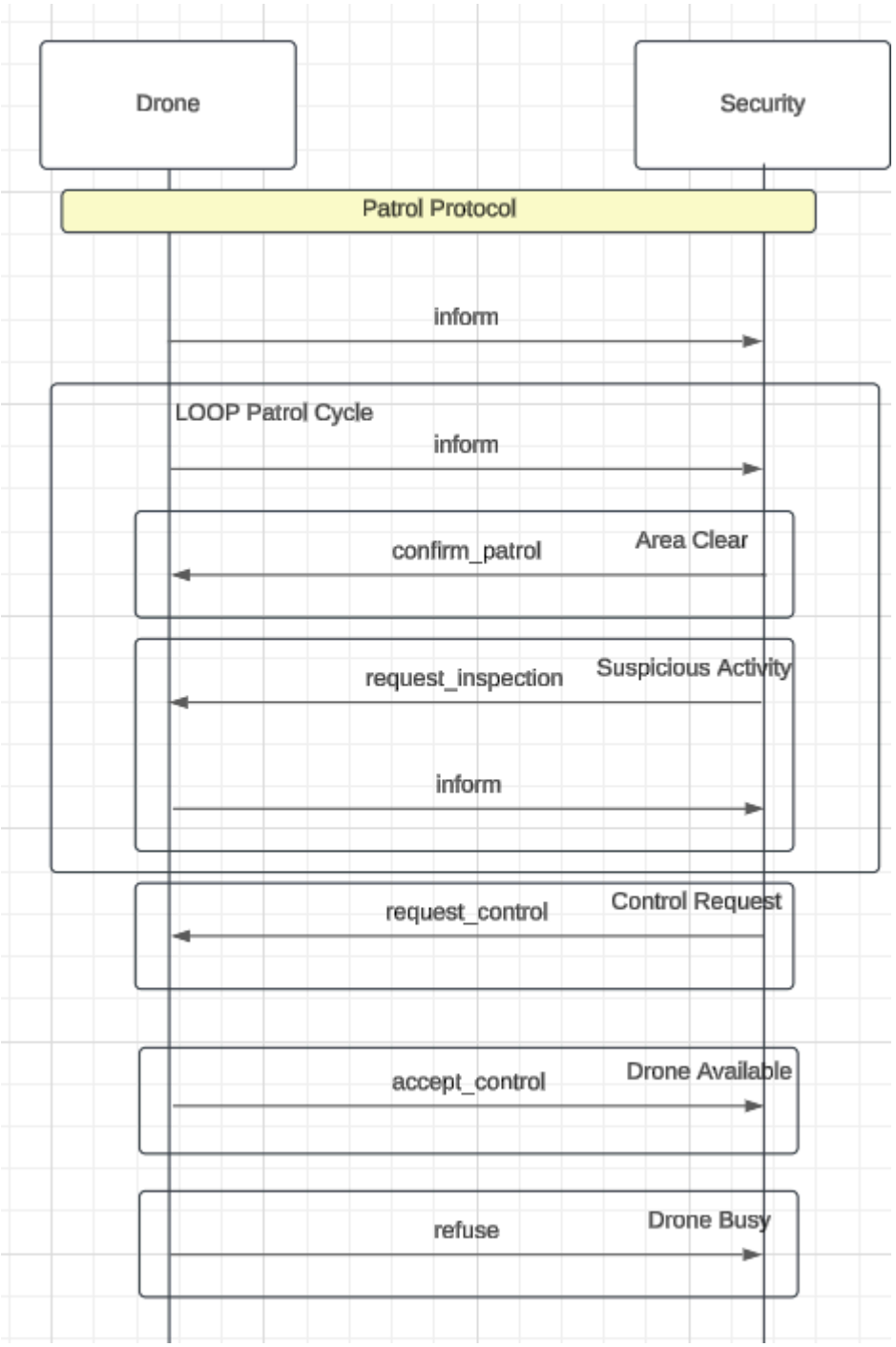
**Sequence diagrams of interaction protocols**

Link a los diagramas: Link




Camera
Drone
Security
Emergency Response Protocol
alert_emergency
inform_emergency_status
Security Available
request_control
accept_control
assign_pursuit
update_threat_location
Threat Contained
inform_contained
release_control
Threat Lost
inform_lost
request_area_scan
Security Busy
mantain_surveillance
request_backup

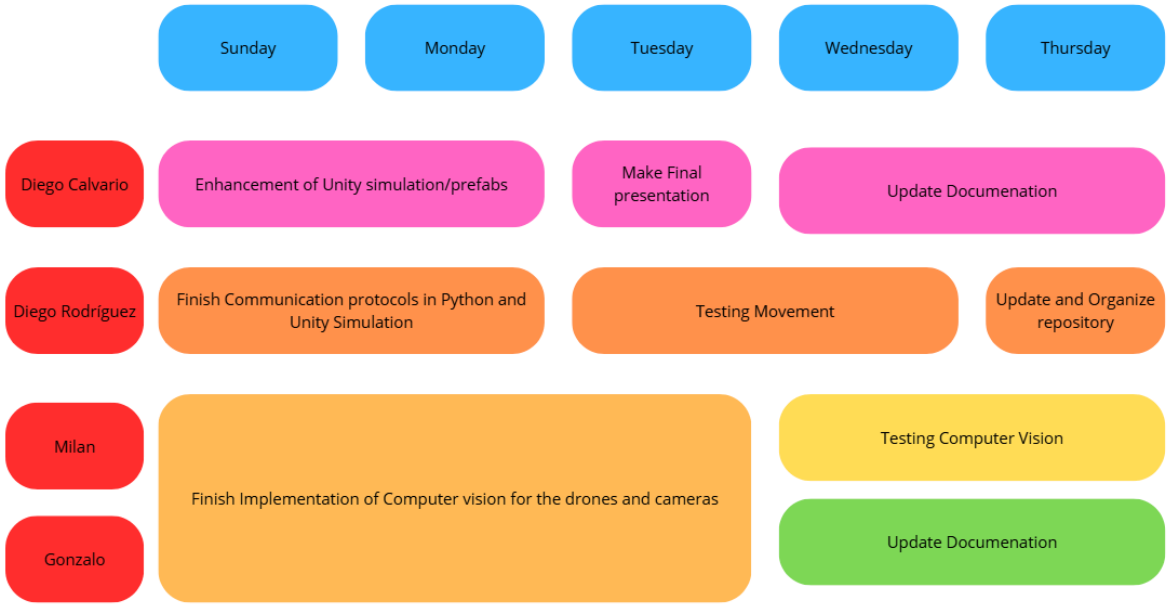

Drone
Security
Camera
Mission Protocol
inform_complete
Mission Success
request_return
accept_return
inform_end_surveillance
inform_returning
inform_landed
confirm_end
Mission Emergency
inform_emergency
accept_emergency
request_monitoring
update_position
inform_emergency_landed
confirm_emergency
release_monitoring


Drone
Security
Patrol Protocol
inform
LOOP Patrol Cycle
inform
confirm_patrol
Area Clear
request_inspection
Suspicious Activity
inform
request_control
Control Request
accept_control
Drone Available
refuse
Drone Busy

**Work Plan**

For the remaining week we have, until this moment, the following plan. Our biggest challenge left is implementing the Computer Vision side of the project, that's why we have half of the team working on that.

---




**Acquired Learning**

Through these 4 weeks we have acquired a lot of knowledge, both theoretical and practical. Talking exclusively to the one related to the challenge, all the Agent theory has been fundamental to the design and development of the agents, the architecture and communications protocols are also a crucial part for these projects that we learned in the classroom. On the practical side of the project, we have increased our previous knowledge about Unity, being able to work with prefabs and creation simulations there that are managed by python servers that we also learned to create. The Python and Unity scripts are the result of the theory acquired during the classes, and are also the base of our project from where other parts like the Computer Vision rely to work correctly.