Evidence 1: Integrated Activity

María Guadalupe Soto Acosta A00228158

Jimena Díaz Franco A01403295

Ilan Gómez Guerrero A01621122

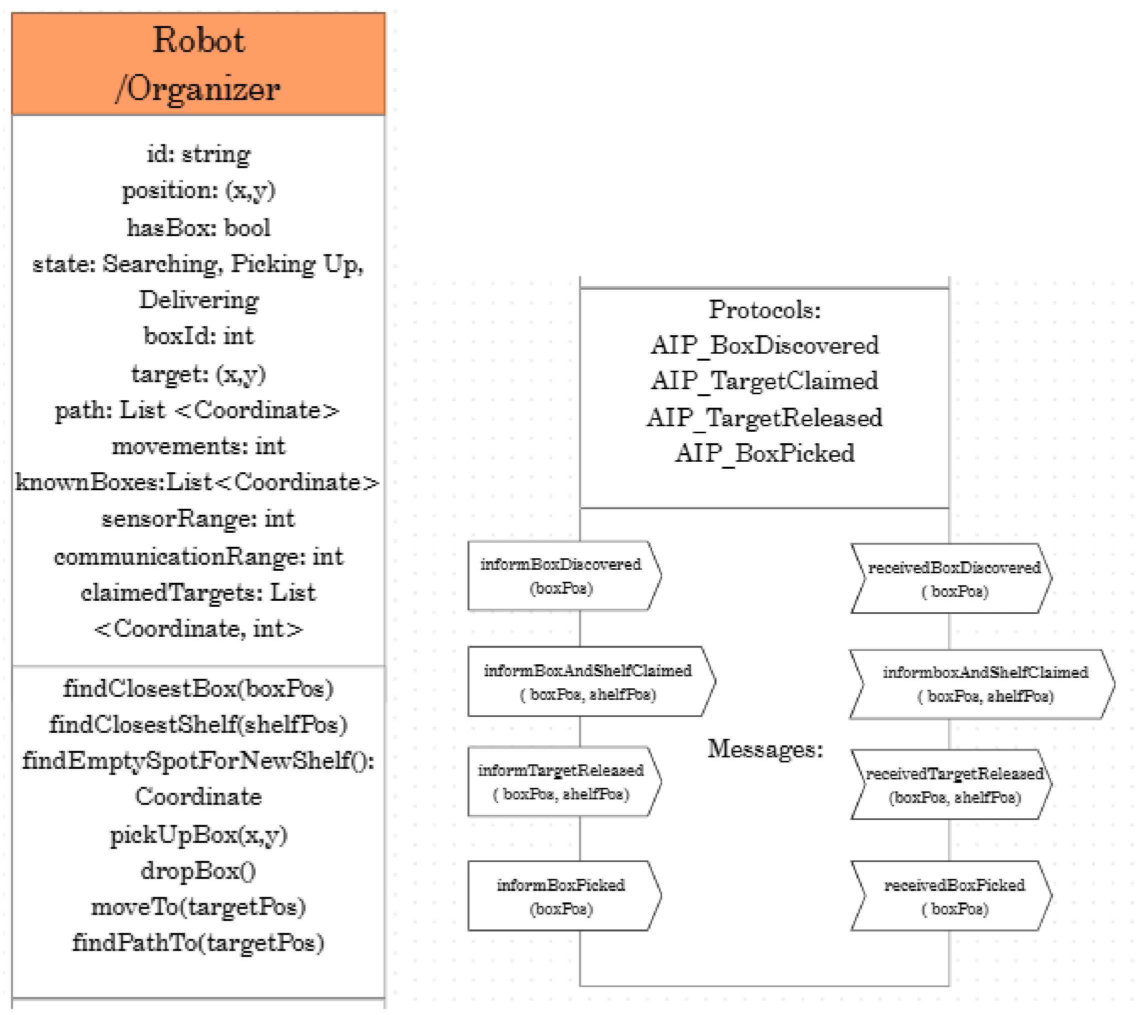
Francisco Raziel Andalón Aguayo A01640235

November 24th, 2025

Modeling of Multi-Agent Systems with Computer Graphics

Group 301

Agent Class Diagram

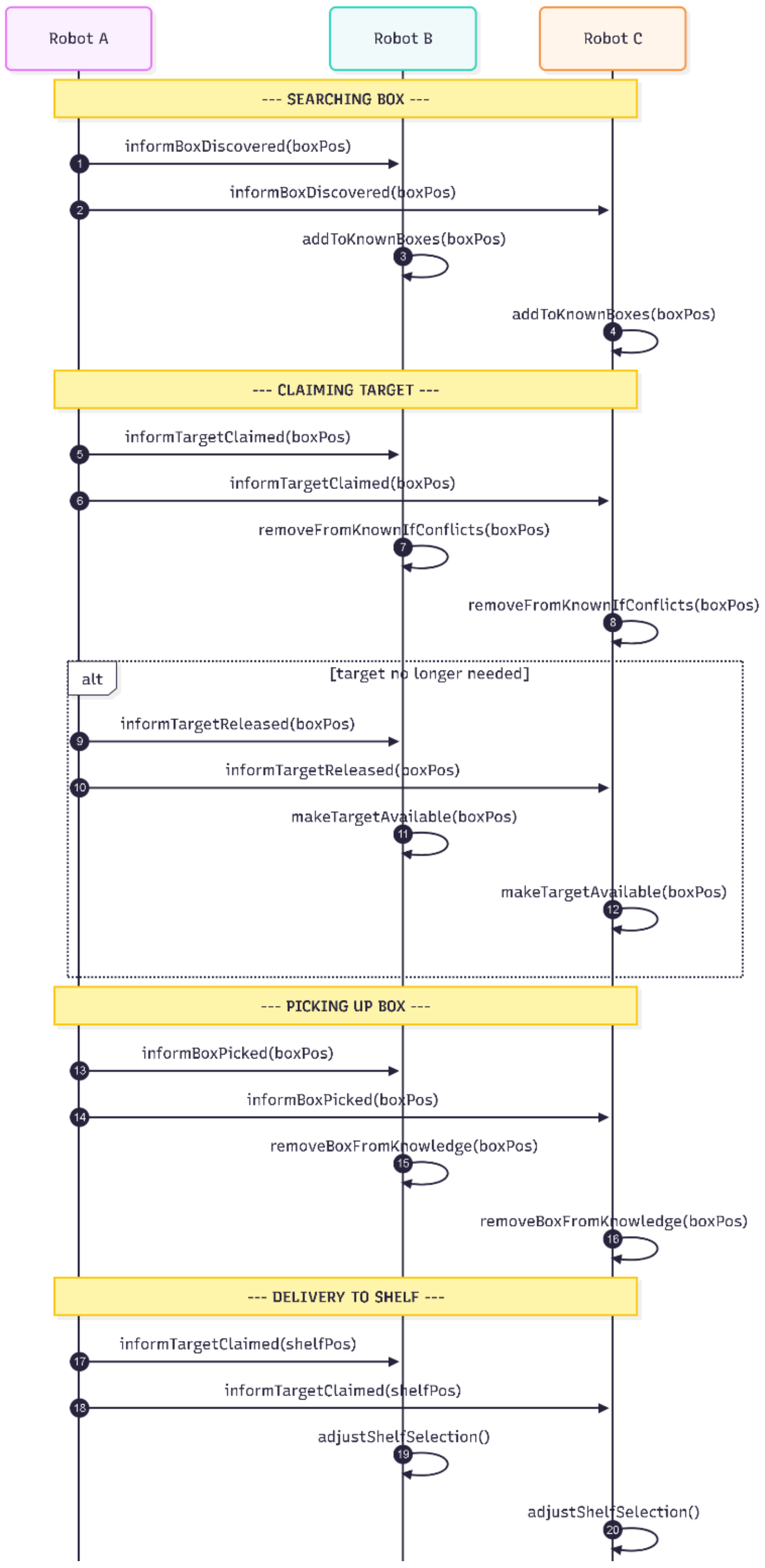

Robot /Organizer
id: string
position: (x,y)
hasBox: bool
state: Searching, Picking Up, Delivering
boxId: int
target: (x,y)
path: List <Coordinate>
movements: int
knownBoxes:List<Coordinate>
sensorRange: int
communicationRange: int
claimedTargets: List <Coordinate, int>
findClosestBox(boxPos)
findClosestShelf(shelfPos)
findEmptySpotForNewShelf(): Coordinate
pickUpBox(x,y)
dropBox()
moveTo(targetPos)
findPathTo(targetPos)
Protocols:
AIP_BoxDiscovered
AIP_TargetClaimed
AIP_TargetReleased
AIP_BoxPicked
Messages:
informBoxDiscovered (boxPos)
receivedBoxDiscovered (boxPos)
informBoxAndShelfClaimed (boxPos, shelfPos)
informboxAndShelfClaimed (boxPos, shelfPos)
informTargetReleased (boxPos, shelfPos)
receivedTargetReleased (boxPos, shelfPos)
informBoxPicked (boxPos)
receivedBoxPicked (boxPos)

The diagram describes how the Robots are built and what they can do. On the left, it lists the robot's attributes, like its current position, whether it has a box, and the state it is in (Searching, Picking up, or Delivering). It also keeps track of things like the box ID it carries, its target, the path it's following, how many movements it has made, the boxes it knows about, and its sensing and communication ranges. It even stores a list of claimed targets so it knows which locations are already taken by other robots.

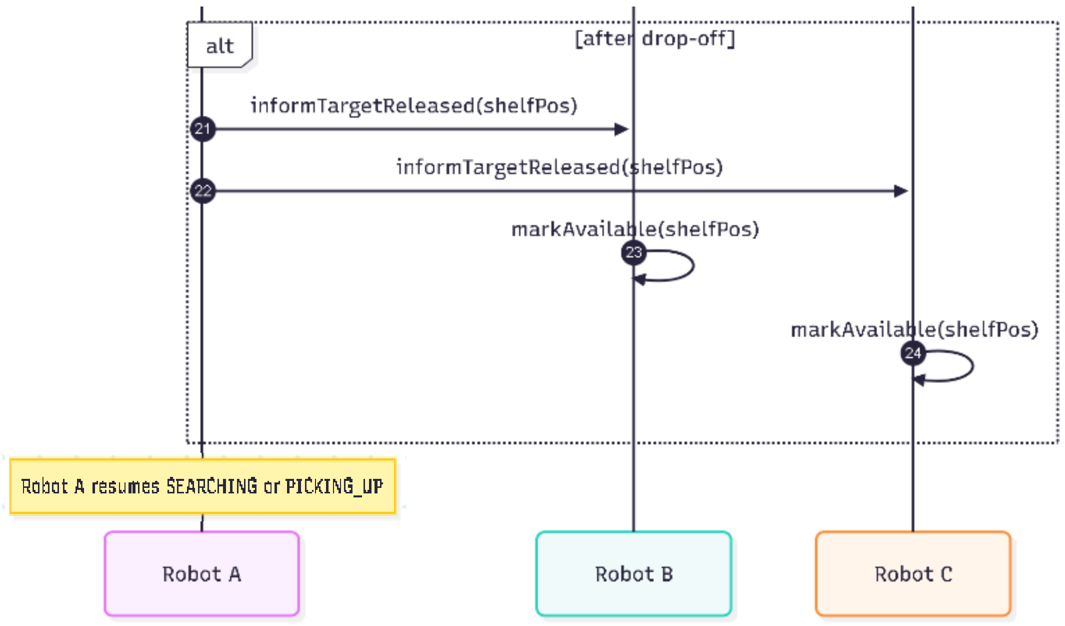
Below that, the diagram shows the robot's main actions, such as finding boxes or shelves (findClosestBox, findClosestShelf), identifying a spot for a new shelf (findEmptySpotForNewShelf), picking up and dropping boxes (pickUpBox, dropBox), moving toward a goal (moveTo), and calculating paths (findPathTo). These methods represent how the robot thinks and acts while organizing the warehouse.

On the right, the diagram includes the protocols the robot uses to communicate: *AIP_BoxDiscovered, AIP_TargetClaimed, AIP_TargetReleased, and AIP_BoxPicked*. Under them are the actual messages sent between robots, like *informBoxDiscovered, informBoxAndShelfClaimed, informTargetReleased, and informBoxPicked*, along with all their "received" versions. These messages let robots warn each other about new boxes, reserve spots, release them, and confirm when a box has been picked up.

Interaction Diagram (AIP)



Robot A
Robot B
Robot C
--- SEARCHING BOX ---
1 informBoxDiscovered(boxPos)
2 informBoxDiscovered(boxPos)
3 addToKnownBoxes(boxPos)
4 addToKnownBoxes(boxPos)
--- CLAIMING TARGET ---
5 informTargetClaimed(boxPos)
6 informTargetClaimed(boxPos)
7 removeFromKnownIfConflicts(boxPos)
8 removeFromKnownIfConflicts(boxPos)
alt
[target no longer needed]
9 informTargetReleased(boxPos)
10 informTargetReleased(boxPos)
11 makeTargetAvailable(boxPos)
12 makeTargetAvailable(boxPos)
--- PICKING UP BOX ---
13 informBoxPicked(boxPos)
14 informBoxPicked(boxPos)
15 removeBoxFromKnowledge(boxPos)
16 removeBoxFromKnowledge(boxPos)
--- DELIVERY TO SHELF ---
17 informTargetClaimed(shelfPos)
18 informTargetClaimed(shelfPos)
19 adjustShelfSelection()
20 adjustShelfSelection()


alt
[after drop-off]
informTargetReleased(shelfPos)
21
informTargetReleased(shelfPos)
22
markAvailable(shelfPos)
23
markAvailable(shelfPos)
24
Robot A resumes SEARCHING or PICKING_UP
Robot A
Robot B
Robot C

This interaction diagram shows how Robot A, Robot B, and Robot C coordinate during the box-organizing task. The diagram is divided into four interaction phases: *Searching Box, Claiming Target, Picking Up Box, and Delivery to Shelf.* In each phase, the robots exchange specific messages and also perform small internal updates that help them avoid conflicts.

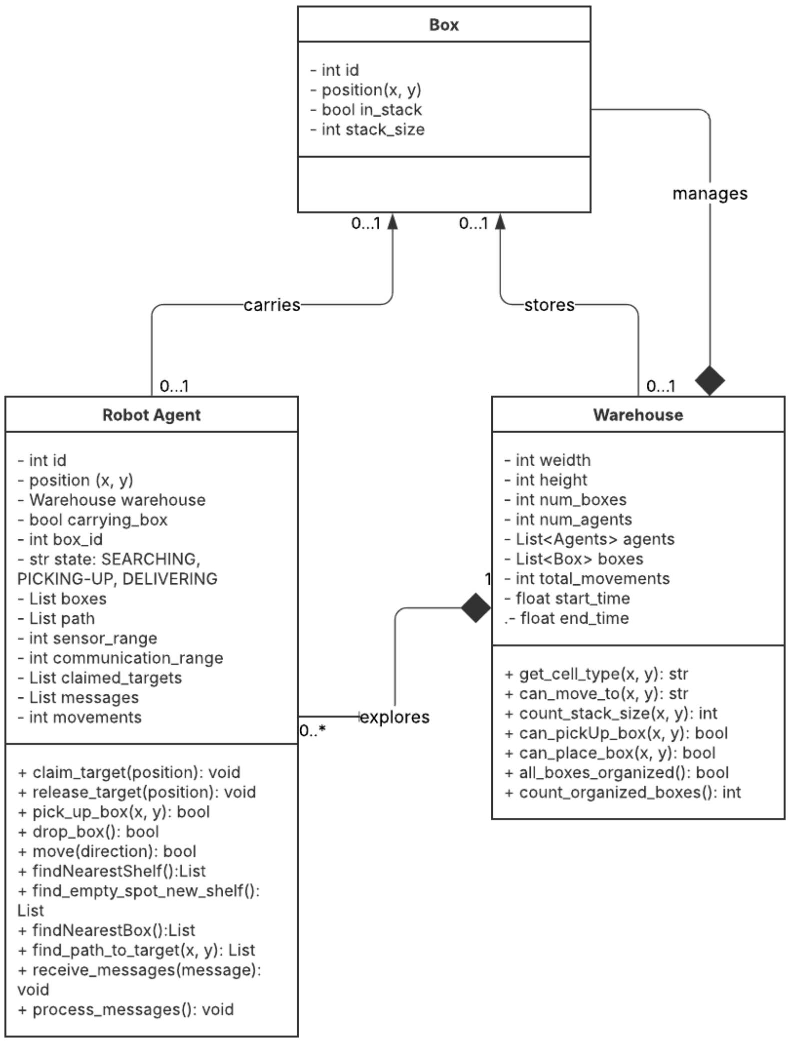
In the *Searching Box* section, Robot A discovers a box and sends the message *informBoxDiscovered(boxPos)* to Robots B and C. Both of them respond internally using *addToKnownBoxes(boxPos)*, so all robots share the same knowledge of the environment.

During *Claiming Target*, Robot A claims the box by sending *informTargetClaimed(boxPos)*. Robots B and C react with *removeFromKnownIfConflicts(boxPos)* so they don't attempt the same target. The alt block in this phase shows the case where Robot A releases the target, broadcasting *informTargetReleased(boxPos)*. Robots B and C then update their state with *makeTargetAvailable(boxPos).*

In *Picking Up Box,* Robot A picks up the box and informs the others using *informBoxPicked(boxPos)*. Robots B and C respond by cleaning their knowledge with *removeBoxFromKnowledge(boxPos).*

Finally, in *Delivery to Shelf,* Robot A chooses the closest shelf position and sends *informTargetClaimed(shelfPos).* Robots B and C adjust their choices using *adjustShelfSelection().* After Robot A finishes delivering, the alt block shows the release of the shelf position through *informTargetReleased(shelfPos)*, and the other robots mark it free with *markAvailable(shelfPos)*. When the robot ends to deliver, it either returns to searching if it didn't find another box during its path, or picks up another known box.

Standard Class Diagram



The standard class diagram outlines a system in which the central class, Box, acts as a basic unit within the warehouse environment. Box contains simple attributes such as an integer ID, a position represented by (x,y) coordinates, a boolean value in_stack, and an integer stack with its size. This class does not exist in isolation: it is linked to the class Warehouse, which stores boxes and maintains the physical structure where all interactions occur. Warehouse includes attributes such as width and height limits, the number of boxes and agents, as well as lists of agents and boxes. It also records the total number of movements performed in the system and stores timing values for start and end. The methods associated with Box and Warehouse allow the system to determine the type of a cell, convert (x,y) positions to strings, check whether an agent can move or interact with a specific coordinate, compute stack size, verify whether all boxes are organized, and count how many of them meet the organizational criteria.

Operating on top of this environment is the Robot Agent class, which explores the Warehouse and performs actions within it. Each agent contains an integer ID, its current (x,y) position, and a direct reference to the Warehouse it navigates. It also includes a boolean indicating whether it is currently carrying a box, the ID of that box if applicable, and a string describing its internal state, which may correspond to searching, picking up, or delivering. In addition, it maintains lists of boxes in its control, paths it must follow, sensor and communication ranges, claimed targets, received messages, and its movement counter.

The agent's operations reflect autonomous behavior inside the environment. Robot Agent can claim and release target positions, pick up and drop boxes, determine whether it can move in a given direction, locate the nearest stack, find nearby boxes, and compute paths to designated targets through methods that return lists of coordinates. It can also receive and process messages, enabling coordination with other agents. In general, the diagram presents a system where Box defines the fundamental structural unit, Warehouse maintains global state, and Robot Agent behavior that interacts with the stored boxes and the overall warehouse layout.

Analyze if there is a strategy that could decrease the time spent, as well as the number of movements made. What would it be like? Describe it.

For this, it first executes several times the Simulation inside the Unity environment. In these simulations it was able to notice the variability of the task. As the main idea is the random search trips of the robots until they can find all boxes, this can lead to many different scenarios to the ones that the random trips send them where the boxes are, or the trips send them to places without boxes, leaving with results that vary from really good times as the execution 4 with a time execution of 1:18 but also times like 2:13 of the number 3 in which the bots lost around 50 seconds searching the last box inside the warehouse.

Something that definitely will decrease the time of execution is a more plannified exploration, mainly establishing more an exploration phase for then after all the warehouse is explored in its totality. So if it was able to register a common between all the robots, so they divide the region and explore as they share their findings, after that we can find a more efficient way of taking and saving in the shelves all the boxes with that information.

Time Execution
1:53.72
2:05.55
2:13.22
1:18.16
1:52.40
1:26.14
1:48.29
1:36.35
1:43.02
Average
1:40.99