Tecnológico de Monterrey

Integrative Activity

Members

Eugenio Guzmán Vargas | A01621084

Diego Alberto Carrillo Castro | A01645484

Braulio Fernando Antero Díaz | A01645356

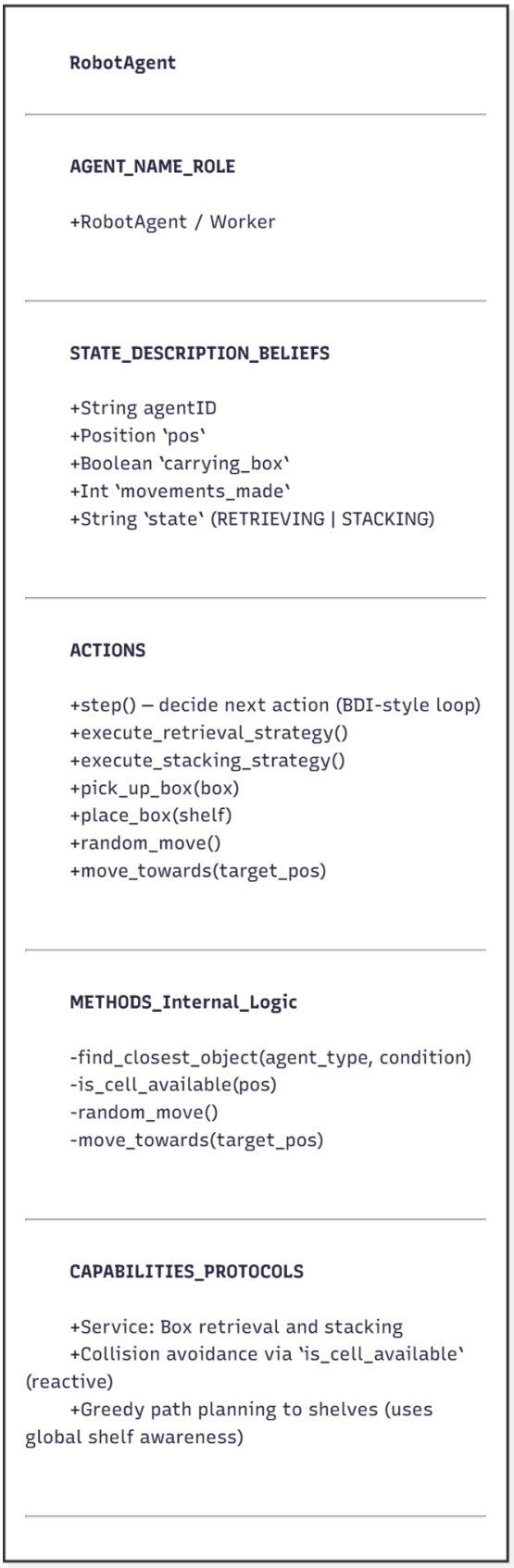
Diego Núñez García | A01644371

Victor Manuel Laureano Vega | A01638979

Modeling of Multi-Agent Systems with Computer Graphics

24/11/2025

Agent Class Diagram:



Beliefs

agentID: Unique identifier

pos: Current grid position, basis for movement and perception.

carrying_box: Boolean flag indicating whether the robot is holding a box, determines behavior mode.

movements_made: Counter of total grid movements (used for metrics).

state: Mode string: "RETRIEVING" or "STACKING".

Actions

step(): Scheduler entry point, executes perceive → decide → act loop.

execute_retrieval_strategy(): If not carrying a box, check current cell for a box → pick up if present, otherwise roam randomly.

execute_stacking_strategy(): If carrying a box, place it if on a viable shelf, otherwise navigate toward nearest non-full shelf.

pick_up_box(box): Marks the robot as carrying a box and removes the Box agent from the grid.

place_box(shelf): Places box on shelf if cell is unobstructed if obstructed, search alternative shelves; if none available, perform random movement to avoid deadlock.

random_move(): Select a random adjacent valid (non-diagonal) free cell and move there.

move_towards(target_pos): Greedy movement toward a target cell attempts prioritized axis movement, falling back to alternate axis or random move if blocked.

Internal Logic

find_closest_object(agent_type, condition): Manhattan-distance search for nearest object of a given type that meets an optional condition.

is_cell_available(pos): Checks if a cell is within bounds and free of robots or obstacles.

random_move() (internal): Uses neighborhood filtering via is_cell_available increments movements_made.

move_towards() (internal): Builds candidate axis moves, validates with is_cell_available, increments movements_made; falls back to random move if no axis move is possible.

place_box() (internal checks): Verifies shelf cell is unobstructed, otherwise finds alternative shelves, or performs random movement.

Capabilities / Protocols

Box retrieval and stacking: Collect boxes and place them onto shelves until completion.

Collision avoidance: Local reactive checks using is_cell_available.

Greedy shelf routing: No global box knowledge, but global knowledge of shelf positions for shortest-path stacking.

Exploration: Random roaming to discover boxes via local perception.

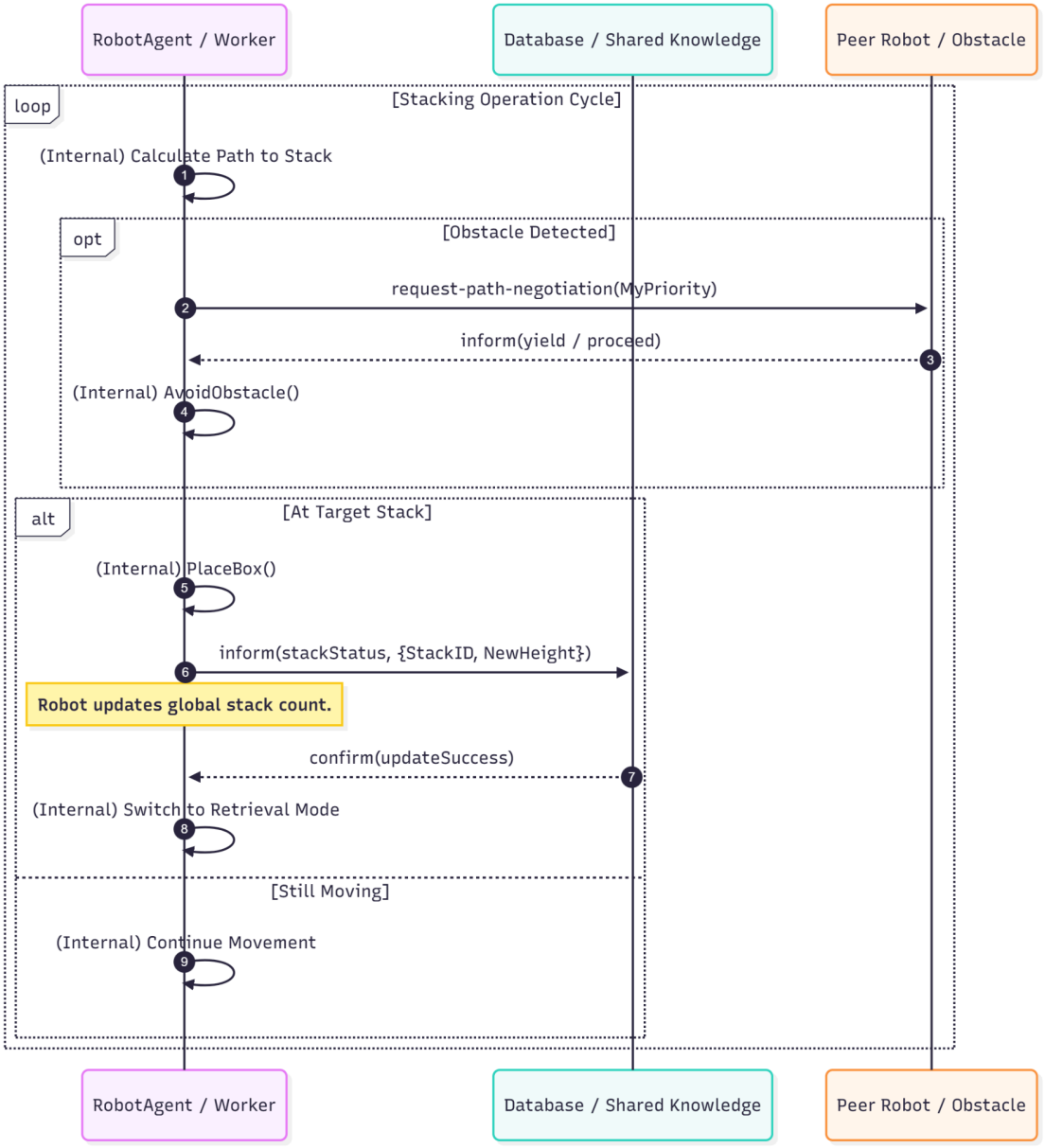
Interactions / Society

Environment: Operates within WarehouseModel providing MultiGrid, scheduler, and placement of Boxes, Shelves, and Obstacles.

Other agents: Picks up Box objects, places boxes onto Shelf objects, avoids collisions with other RobotAgents, and treats Obstacle objects as impassable.

Metrics: movements_made and shelf.stack_height tracked via model.datacollector or UI for performance monitoring.

AIP: Stacking And Reporting



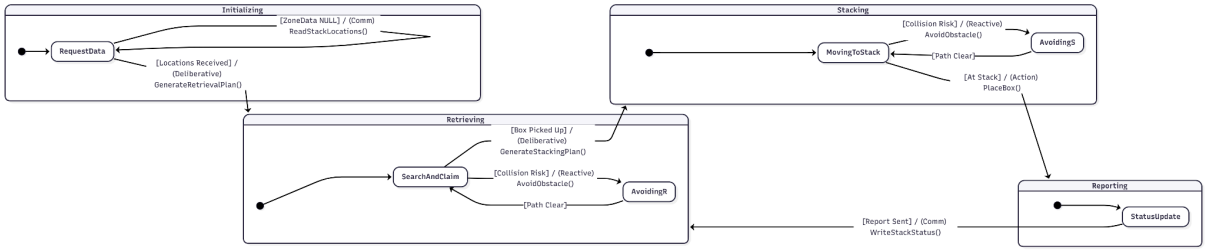
The detailed process of the agent shows 3 different loops, starting with a loop that entire process is contained within, meaning the robot continuously assesses its situation while in the stacking phase, for a conflict resolution such as finding an obstacle or crossing with another agent,it requests for a clear path in order to find another way, but in the case if the agents, the one that passes through is the one that is going first.

Finally with the decision logic (alternative paths), the robot checks its location using an alternative block:

- Scenario A: The robot places the box, it sends an inform message with the stackID, confirms the update and as a result the robot switches its internal mode back to retrieval mode.

- Scenario B: If the robot hasn't reached the stack yet, it simply triggers the (Internal) Continue Movement.

Cooperative Strategy Diagram



This diagram represents an overview of the warehouse system in the different states of the entire process. It visualizes the lifecycle of a robot "agent" that retrieves boxes, moves them to a stack and reports the status.

For initializing the primary goal is to get necessary data to start working, the robot enters the RequestData state and if the data is empty it triggers a communication action and loops back to try again.

Continuing with retrieving, the main goal is to find and pick up a box, the main state is SearchAndClaim, if an obstacle is detected, the robot temporarily switches to AvoidingR, once the path is clear, it returns to search for boxes.

Once its carrying a box, we have the stacking phase, where the main goal is to transport the box to the stack, the main state is MovingToStack, similar as retrieving, if it finds an obstacle, the agent has to find its way out to a clear path, once its done and the robot gets to a stack, it places the box and reports the status.

The final phase Reporting has the goal of updating the central system, when the agent enters the StatusUpdate, the systems loops back to Retrieving (not initializing).

Strategy to decrease the time spent and movements made

The code developed for the first part of the activity initializes the agents without them knowing the location of the boxes they are searching for; they only know the position of the shelves where they must place them.

The search for these boxes is performed randomly: the agents move throughout the area and, if they find a box, they take it to the nearest shelf.

Therefore, a first option to decrease the time it takes them to complete the task, and the total number of movements executed, could be to implement a more efficient search algorithm.

Two possible options to reduce randomness are: random walk with directional bias (that is, before changing direction, they follow a specific direction for a certain number of steps, which would help avoid immediately exploring the same areas) or a type of dynamic local memory (which could be implemented with a matrix of counters, representing something similar to a heat map, where each recently visited cell becomes less attractive for revisiting).

Territoriality could also be considered for task allocation, which would involve assigning an area to each agent to avoid duplicating search efforts for the same boxes.

Similarly, the communication protocols of the agents could be worked on, so that they can share areas already explored or, failing that, areas that are pending to be explored, again thinking about avoiding repetitive searches.