Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Guadalajara

<u>Modeling of Multi-Agent Systems with Computer Graphics</u>

Evidence 1 - Integrating Activity

Part 1. Multiagent Systems Documentation

Professor:

Iván Axel Dounce Nava

Students:

Sergio Eduardo Gutiérrez Torres | A01068505

Ana Karen Valencia Villaseñor | A01067717

Sebastian Cervera Maltos | A01068436

Alexei Delgado De Gante | A01637405

Emiliano Neaves Ortiz | A01568982

Daniela Rocha Muñoz | A00573664

Sunday, November 17, 2024

1. AGENT PROPERTY SPECIFICATIONS

Within this simulation, each agent type of Robot and Object is defined with specific properties that determine their behavior and state within the warehouse environment, which are explained below:

A. Robot Properties
   a. *movements*: Tracks the total number of movements made by the robot.
   b. *carrying*: Indicates the object currently being carried by the robot (if any).
   c. *orientation*: Tracks the robot's current facing direction ('north' {N}, 'east' {E}, 'south' {S}, 'west' {W}).
   d. *state*: Represents the robot's current state, defined in the *RobotState* enumeration; such as “IDLE”, “MOVING”, or “AVOIDING_COLLISION”.
   e. *priority*: A unique priority assigned to each robot, typically based on its ID.
   f. *path*: Stores the planned path for the robot to move to a target location.
   g. *reactive_goal & deliberative_goal*: Short-term and long-term goals for decision-making.
   h. *previous_positions*: A stack that records the robot's movement history to facilitate backward movement.
   i. *target*: The current target location or object the robot is interacting with.

B. Object Properties
   a. *stack_level*: Tracks the current stack level of the object.
   b. *sorted*: Indicates whether the object has been placed in its final position.
   c. *picked*: Flags if the object has been picked up by a robot.

C. Shared Model-Level Properties
   a. *reservation_table*: Tracks reserved grid positions to avoid collisions.
   b. *shared_intentions*: Stores collective planned positions for multi-agent coordination.
   c. *stackable_cells*: A cache to track stack levels at each grid cell.

2. UTILITY OR SUCCESS METRIC FOR EACH AGENT

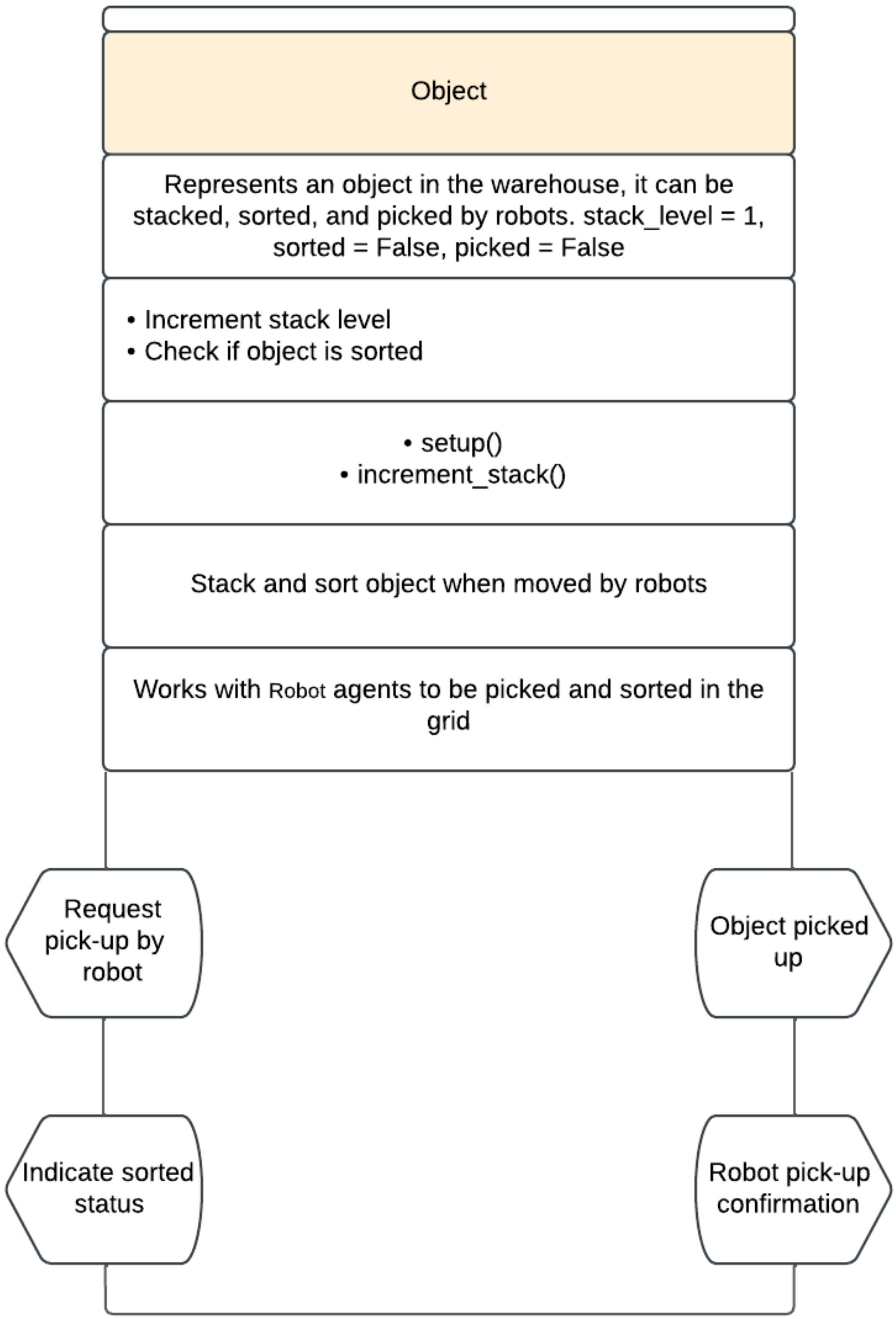
The success of each agent is going to be determined by its ability to achieve its goals effectively within the established limits of the simulation.

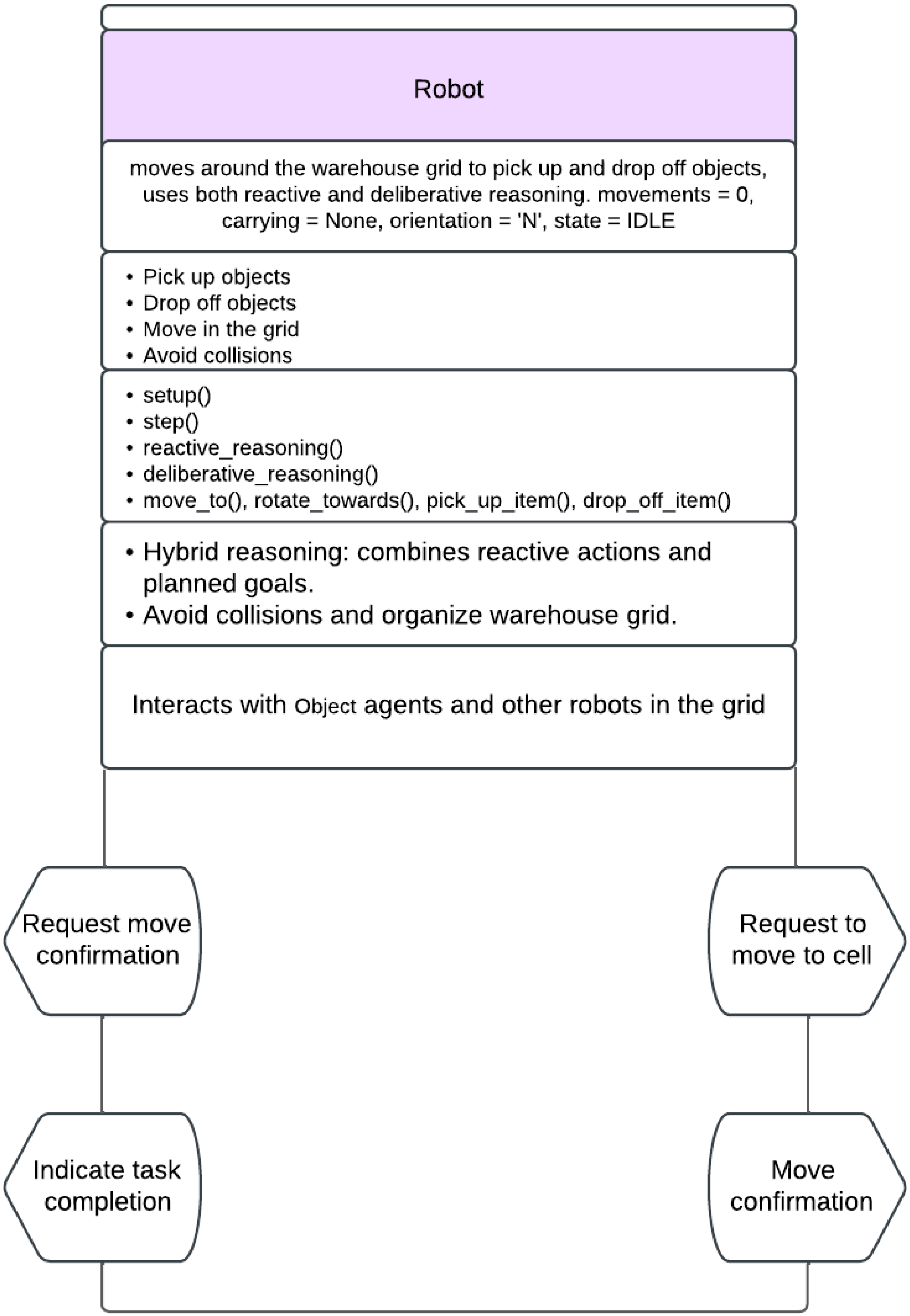
A. Robot Utility Metrics
   a. *Number of Objects Delivered*: Measures how many objects the robot successfully picks up and drops off at the correct location.
   b. *Minimized Movements*: Robots aim to achieve their objectives using the fewest possible movements.
   c. *Collision Avoidance Events*: A lower number of collision avoidance events indicates more efficient planning and navigation.
   d. *Path Efficiency*: The ratio of the shortest possible path to the actual path taken by the robot.

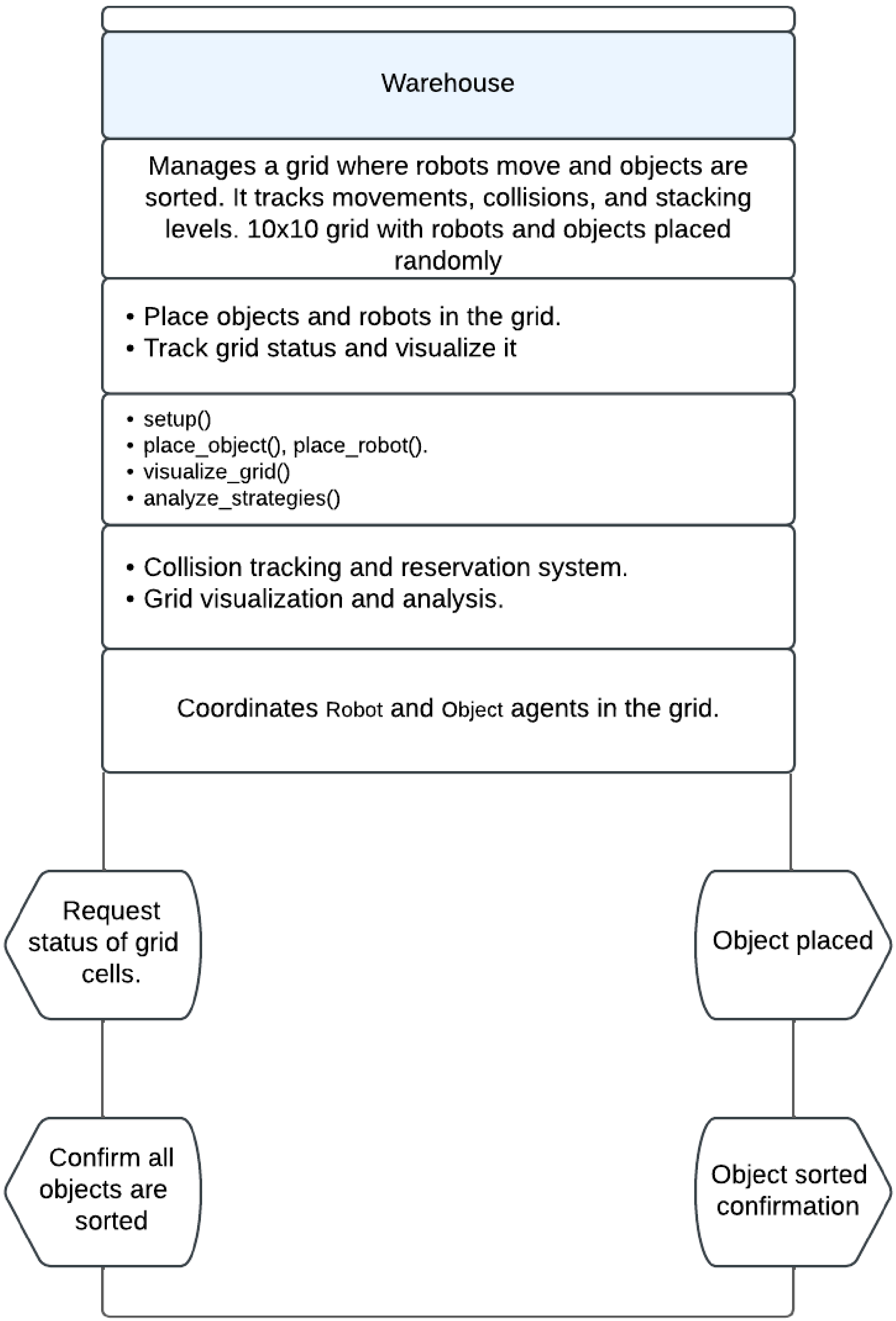
B. Object Utility Metrics
   a. *Sorting Status*: An object is considered "successful" when it is placed in its final sorted position.
   b. *Stacking Level*: Objects should contribute to forming complete stacks (up to the maximum stack level of 5).

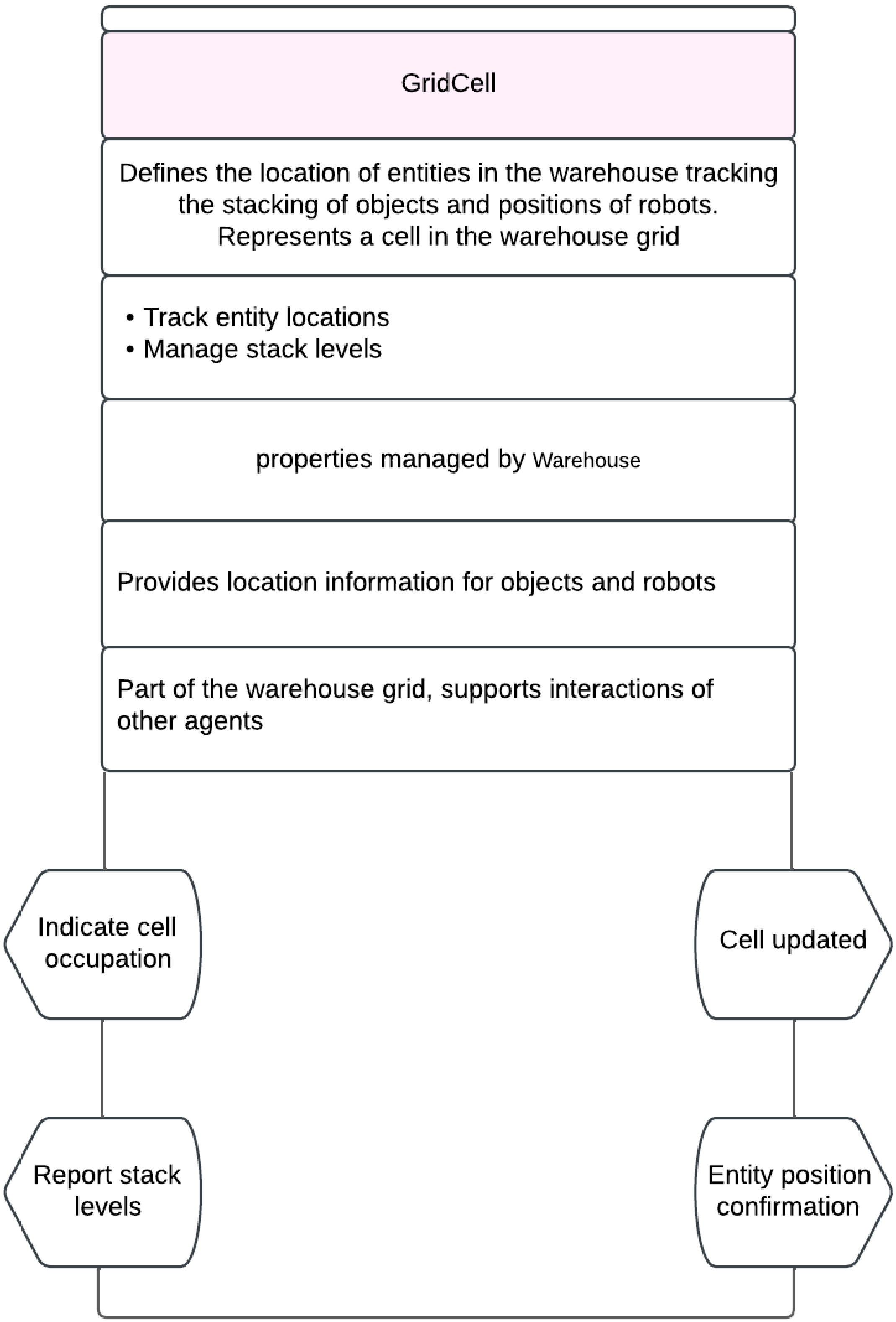
C. Global Success Metrics for the System
   a. *Time to Completion*: The total number of steps required to sort all objects.
   b. *Average Movements per Robot*: Indicates the overall efficiency of robot operations.
   c. *Collisions Avoided*: The number of instances where robots successfully avoided conflicts through reactive reasoning.

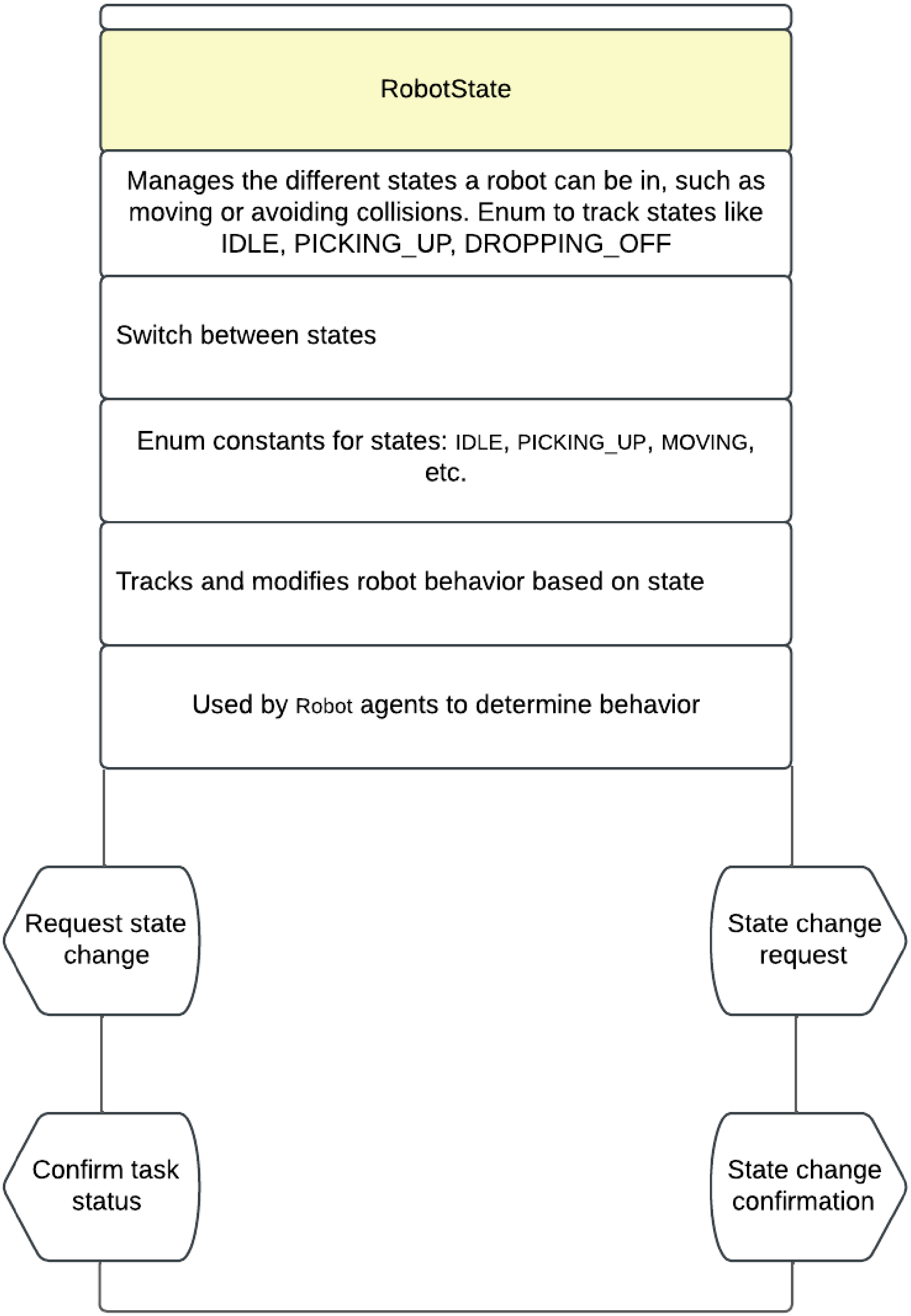
3. AGENT CLASS DIAGRAMS

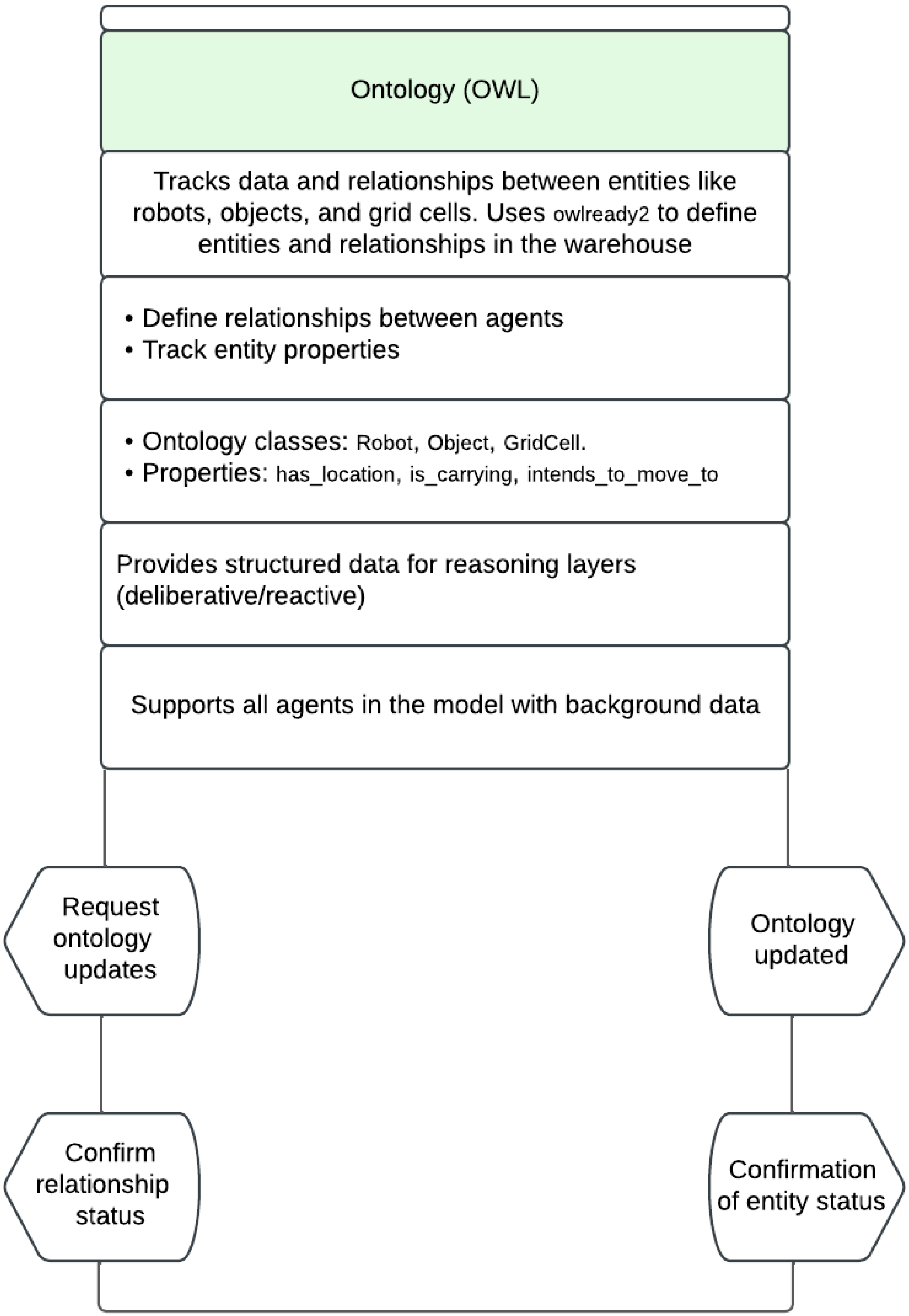



Robot
moves around the warehouse grid to pick up and drop off objects, uses both reactive and deliberative reasoning. movements = 0, carrying = None, orientation = 'N', state = IDLE
• Pick up objects
• Drop off objects
• Move in the grid
• Avoid collisions
• setup()
• step()
• reactive_reasoning()
• deliberative_reasoning()
• move_to(), rotate_towards(), pick_up_item(), drop_off_item()
• Hybrid reasoning: combines reactive actions and planned goals.
• Avoid collisions and organize warehouse grid.
Interacts with Object agents and other robots in the grid
Request move confirmation
Request to move to cell
Indicate task completion
Move confirmation


Warehouse
Manages a grid where robots move and objects are sorted. It tracks movements, collisions, and stacking levels. 10x10 grid with robots and objects placed randomly
• Place objects and robots in the grid.
• Track grid status and visualize it
• setup()
• place_object(), place_robot().
• visualize_grid()
• analyze_strategies()
• Collision tracking and reservation system.
• Grid visualization and analysis.
Coordinates Robot and Object agents in the grid.
Request status of grid cells.
Object placed
Confirm all objects are sorted
Object sorted confirmation


GridCell
Defines the location of entities in the warehouse tracking the stacking of objects and positions of robots. Represents a cell in the warehouse grid
• Track entity locations
• Manage stack levels
properties managed by Warehouse
Provides location information for objects and robots
Part of the warehouse grid, supports interactions of other agents
Indicate cell occupation
Report stack levels
Cell updated
Entity position confirmation


RobotState
Manages the different states a robot can be in, such as moving or avoiding collisions. Enum to track states like IDLE, PICKING_UP, DROPPING_OFF
Switch between states
Enum constants for states: IDLE, PICKING_UP, MOVING, etc.
Tracks and modifies robot behavior based on state
Used by Robot agents to determine behavior
Request state change
State change request
Confirm task status
State change confirmation


Ontology (OWL)
Tracks data and relationships between entities like robots, objects, and grid cells. Uses owlready2 to define entities and relationships in the warehouse
• Define relationships between agents
• Track entity properties
• Ontology classes: Robot, Object, GridCell.
• Properties: has_location, is_carrying, intends_to_move_to
Provides structured data for reasoning layers (deliberative/reactive)
Supports all agents in the model with background data
Request ontology updates
Ontology updated
Confirm relationship status
Confirmation of entity status

4. ANTHOLOGY CLASS DIAGRAMS

Entity
This is the main class, it is for all things in the warehouse, like robots and objects. Base class for everything in the warehouse
Connect to a cell in the grid
has_location: Links an Entity to a grid cell (GridCell)
To show where things are in the warehouse
Base class for Robot and Object

Robot
Robots move and pick up objects in the warehouse grid, a subclass of Entity, it is a robot in the warehouse
• Move around the grid. • Carry objects.
• is_carrying • intends_to_move_to
Move objects to the right places in the warehouse.
Works with Object and GridCell to keep things in order.

Object
Items that robots can carry and stack in the warehouse, subclass of Entity, it is a thing robots can move and stack
• Be picked up. • Be stacked.
is_stackable
Be moved and stacked by robots.
Part of the warehouse system with other objects.

GridCell
Shows positions in the grid where objects and robots can be, shows a spot in the warehouse grid
Hold the position of things in the grid
has_location
Keep track of where everything is in the warehouse
Part of the grid that helps robots and objects know where to go

5. CONCLUSION

This project helped us to develop and implement a simulation system for an automated warehouse, thanks to the use of agents and various computer components, which combines hybrid reasoning, ontologies and their complete integration into a unified agent model.

The modeled system uses a combination of interrelated components to simulate an automated warehouse, where robots and objects interact within a structured grid. The results obtained show the capacity of these components to coordinate complex tasks, resolve conflicts efficiently and promote collaboration between agents in a dynamic environment.

Among the alternative solutions that could improve agent efficiency, we could address 4 main points:

a. Better Use of Ontology: Where we could work on including a *semantic classification* that helps differentiate objects according to personal size or priority attributes; include *relationships to historical data* collected about routes used and status of stacking sites; or even *subdivide the ontology* to facilitate access to specific data and issues.

b. Better Pathfinding: Use *other types of algorithms* to plan agent routes such as *dynamic heuristics*, based on occupation and the state of the environment; or *based on graphs* such as Dijkstra (already explored in another training unit) that help to make routes more efficient. Also, an *obstacle prediction system* would help to anticipate conflicts and adjust routes before encountering other agents.

c. Better Ordering Strategy: Systems that help to establish a *prioritization of objects*, according to their importance or their stacking hierarchy; as well as implementing some form of *intelligent zoning* where specific spaces can be set up for different tasks such as collecting, sorting and stacking.

d. Better Conflict Resolution: And finally, systems that help us solve problems that may arise during practice, such as mechanisms for *prioritizing objects or agents*, which help us know if two agents get stuck, which of the two has a higher priority to continue along that path; escape or *secondary routes* so that robots can get out of bottlenecks without stopping; or *cooperative* *planning* where they can communicate their future plans, both on the route and at the site where they plan to stack, etc.

Finally, as a sort of final reflection, this integrative activity not only helped us to complement our understanding of multi-agent systems that we were learning during this first part of the course, but also offered us a better perspective on how all this knowledge is applicable and valuable in an environment of real problems; from logistics issues to the realization of automation and its robotics.

Likewise, we are aware that for the final challenge of the training unit (TU), improvements and a higher level of knowledge expansion will be necessary, given that the system will have to adapt to a more complex and elaborate scenario, which will also contain new types of agent, system and staff behaviors; So this first approach to a real-world implementation effectively contributed to our academic development in the subject.