Tecnológico de Monterrey

Evidence 2. Review 2

Members

Eugenio Guzmán Vargas | A01621084

Diego Alberto Carrillo Castro | A01645484

Braulio Fernando Antero Díaz | A01645356

Diego Núñez García | A01644371

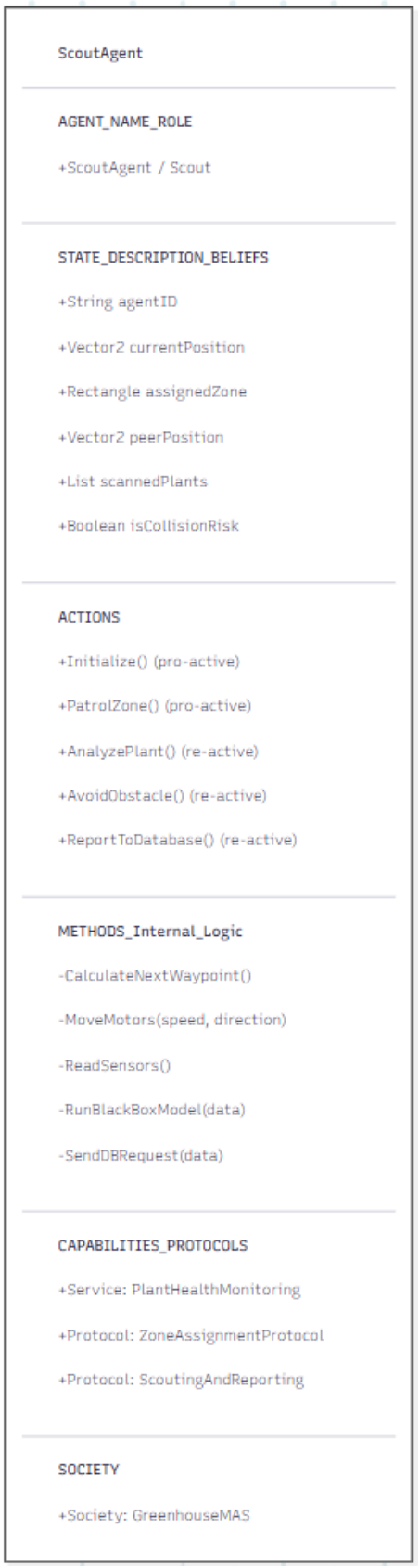
Victor Manuel Laureano Vega | A01638979

Modeling of Multi-Agent Systems with Computer Graphics

18/11/2025

Multi-Agent Systems:

Diagram 1: Agent class



The AGENT_NAME_ROLE section identifies the class as ScoutAgent and its role as Scout, establishing it as the single blueprint for the R1 and R2 agents.

The STATE_DESCRIPTION_BELIEFS section contains the agent's working memory. This includes key data for navigation (currentPosition), task execution (assignedZone), and coordination (peerPosition). A list of scannedPlants is included to ensure work is not repeated.

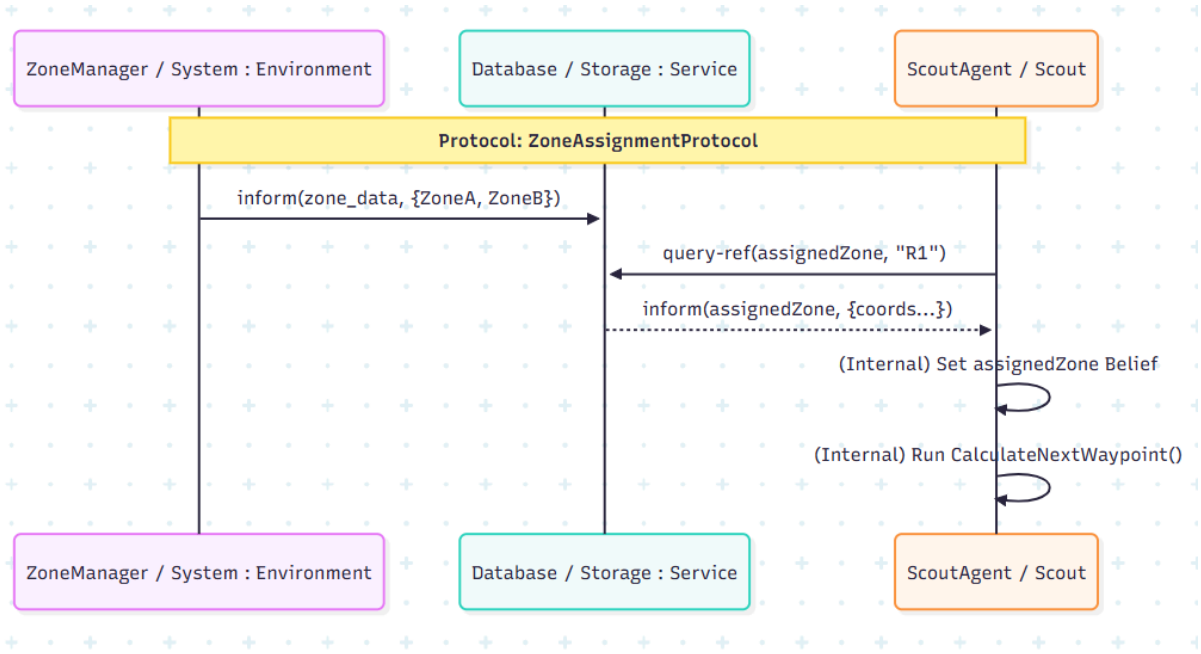
The ACTIONS section lists the agent's high-level capabilities. PatrolZone is the main pro-active behavior, while AnalyzePlant and AvoidObstacle are re-active behaviors triggered by sensor events. ReportToDatabase is the agent's method of communicating its findings to the system.

The METHODS_Internal_Logic section details the functions that implement the high-level ACTIONS. This includes MoveMotors for physical movement, RunBlackBoxModel to execute the plant analysis, and SendDBRequest to handle data transmission.

The CAPABILITIES_PROTOCOLS section defines the agent's understanding of system-wide rules. ZoneAssignmentProtocol is used at startup to get its assigned area. ScoutingAndReporting is the protocol for scouting the warehouse and correctly reading and writing to the database to avoid conflicts and share information.

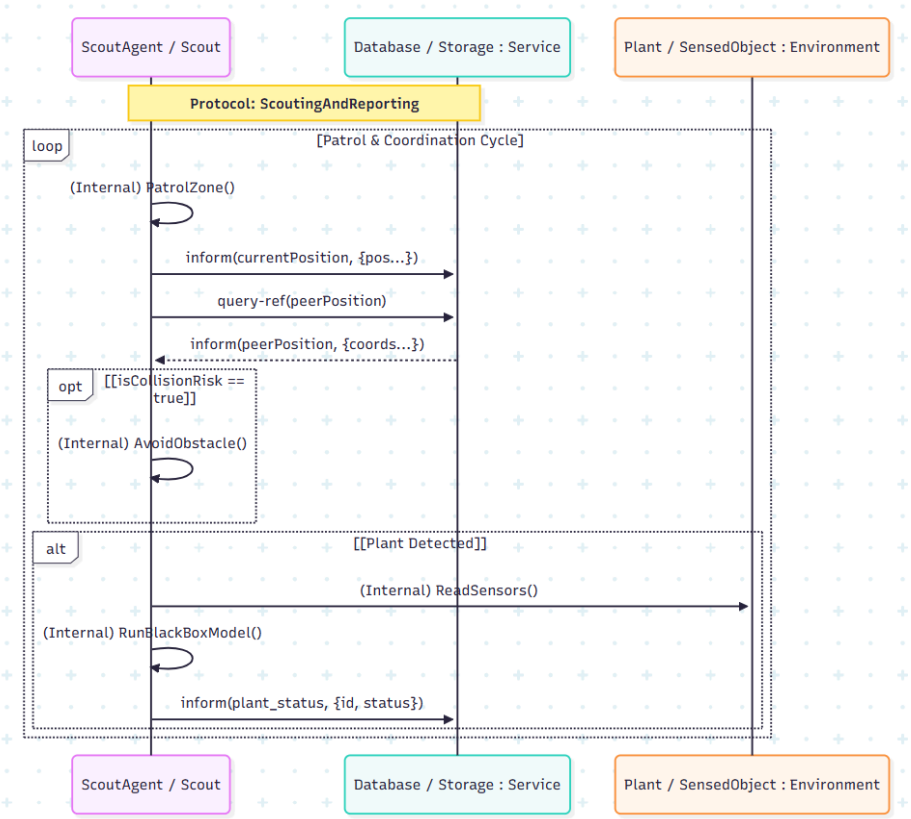
Finally, the SOCIETY compartment, GreenhouseMAS, identifies the Multi-Agent System to which this agent belongs.

Diagram 2: AIP: Zone Assignment Protocol



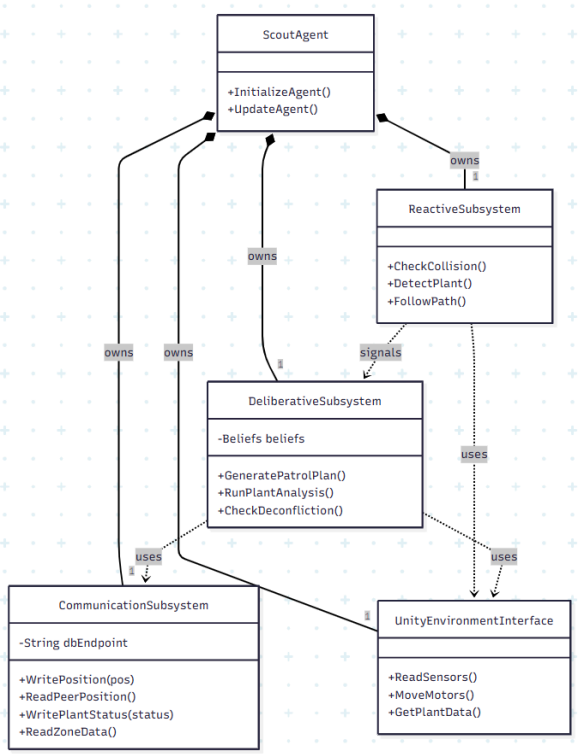
This diagram shows the interaction for the ZoneAssignmentProtocol. It visualizes how the Initialize() (pro-active) action in the agent is fulfilled by communicating with the database, which has been pre-filled by the ZoneManager (a non-agent system component).

Diagram 3: AIP: Scouting and reporting



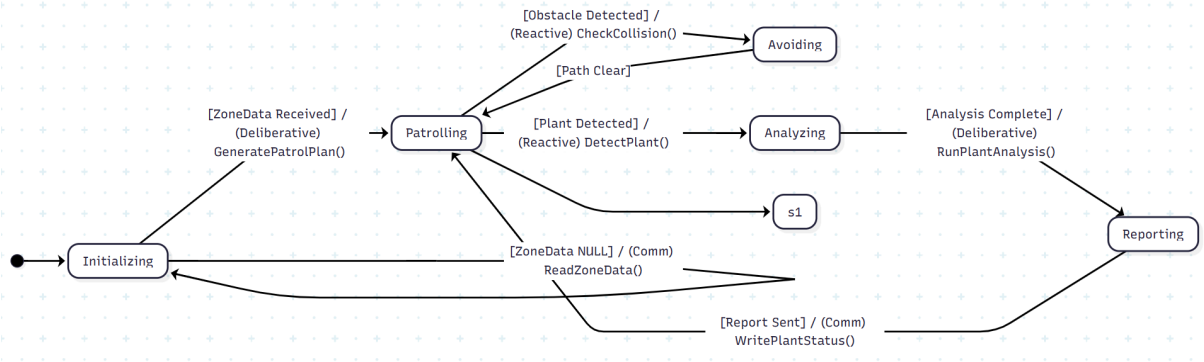
This diagram shows the ScoutAndReport protocol. It visualizes the agent's main operational loop, driven by its PatrolZone() action. It shows how it uses the database to both report its status and read its peer's status, triggering the AvoidObstacle() action if necessary. It also shows the AnalyzePlant() and ReportToDatabase() sequence.

Diagram 4: Agent subsystem class diagrams



This diagram shows the ScoutAgent internal architecture, where the main ScoutAgent class acts as a central controller. It owns and coordinates several specialized subsystems: the DeliberativeSubsystem handles thinking tasks like GeneratePatrolPlan() and RunPlantAnalysis(); the ReactiveSubsystem manages immediate responsive actions such as CheckCollision() and signaling the deliberative layer when a plant is detected; the CommunicationSubsystem serves as a dedicated network interface, abstracting all database interactions like WritePosition(); and the UnityEnvironmentInterface provides the low-level bridge to the game engine, handling ReadSensors() and MoveMotors()

Diagram 5: Agent Subsystem State Diagram



This diagram illustrates the agent's behavioral flow, showing how it shifts between states based on specific triggers from its subsystems. The agent begins in the Initializing state, where it repeatedly attempts to fetch its zone data. A [ZoneData Received] trigger from the CommunicationSubsystem confirms it has its assignment, which transitions the agent to the Patrolling state. Patrolling is the agent's default operational state, driven by its ReactiveSubsystem. This state can be interrupted by two triggers:

1. An [Obstacle Detected] trigger (e.g., from a sensor) immediately shifts the agent to the high-priority Avoiding state. It remains there until a [Path Clear] trigger allows it to resume patrolling.
2. A [Plant Detected] trigger (from the ReactiveSubsystem) transitions the agent to the Analyzing state, passing control to the DeliberativeSubsystem.

Once in the Analyzing state, an [Analysis Complete] trigger signifies the black-box model has finished. This moves the agent to the Reporting state. Finally, a [Report Sent] trigger from the CommunicationSubsystem confirms the data is in the database, and the agent transitions back to Patrolling to continue its cycle.

Computer Graphics:

How do we visualize the world

Our virtual environment will be the representation of a tomato greenhouse. Commonly, these have a simple, pattern-like distribution, where there are lines of crops, placed parallel to each other, with spaces between them where farmers walk. These are the paths our agents will be going through. For now, we will not take environmental elements(weather, sunlight, etc.) into consideration for our simulation, and we will focus on recreating the greenhouse. The terrain, although not purely plain, is not expected to contain obstacles or irregularities that the agents can not go over or around.

What virtual elements are key for our proposal

The greenhouse, the crop lines, the tomato plants, the terrain, and the agent itself (a small robot with a camera and other sensors).

How do we visualize the models of the agents and other relevant objects

The scout agent is a small robot that will be moving between the lines of the greenhouse to scan each plant. The Plant proposed initially will have one tomato in order to perform the appropriate tests, this plant will count with multiple leafs along the main tail of it and there are going to be plenty of them scattered around the map.

Schematics/Hand-Drawn sketches to support the description

Plant prototype

Proof models

Organization sketch