Tecnológico de Monterrey

Guadalajara campus

Evidence 2 - Final Delivery

Sebastián Alejandro Soria Piñuela - A01645849

Modeling of multi-agent systems with computers graphics

Group 301

December 4, 2025

1. Analysis of the Developed Solution

1.1 Why We Selected the Multi-Agent Model Used

We selected a multi-agent model because the behaviors required in our simulation, such as: mapping, detection, classification, and collection; should be separated into different autonomous responsibilities for a better result. A single, monolithic agent would not scale effectively as tomatoes spawn dynamically, collectors must make path decisions, and the drone must continuously update the state of the environment.

In the project we have:

- A static Map Agent manages global knowledge and supports all other agents with spatial context for navigation.

- A mobile Inspector Drone performs continuous scanning, enabling or disabling collector actions depending on detection results.

- Multiple Collector Robots independently harvest tomatoes but operate under constraints like capacity limits, A* navigation, and plant exclusivity.

This division aligns perfectly with MAS principles: distributed decision making, autonomy, cooperation, and environmental awareness. The complexity of tomato spawning, path allocation, plant restrictions, and varying tomato health states would be extremely inefficient to manage under one centralized model. Therefore, a MAS was the only architecture that allowed the simulation to remain flexible, scalable, and reactive in real time.

If we had used only reactive agents, the system would lack any explicit planning layer, and each unit would simply respond to local stimuli without considering global efficiency. In our scenario, with tomatoes spawning dynamically and multiple collectors sharing the same space, this would greatly increase execution time and could even prevent the system from ever reaching a stable state. On the other hand, using only BDI agents for every role would introduce unnecessary cognitive overhead and make each agent more vulnerable to abrupt changes in the environment, such as collisions or sudden congestion. By combining a static map agent, a scanning drone with decision capabilities, and reactive collectors, the final architecture balances planning, adaptability, and execution speed.

1.2 Variables Considered in the Decision

- Environmental Variables:
    - Number of plants and their grid positions.
    - Obstacles and restricted tiles affecting A* navigation.
    - Tomato spawn frequency and probability of defectiveness.
- Agent Variables:
    - Capacity limits of collectors
    - Collector distances relative to their assigned plant.
    - Plant exclusivity, restricting one collector per plant.
    - Shared global map updated and distributed by the Map Agent.
    - Continuous detection coverage from the Inspector Drone.
- Simulation Variables
    - Real-time updates when tomatoes appear or when tasks are completed.
    - Navigation costs are determined by the layout and agents' positions.
    - Communication timing between inspectors and collectors.

These variables interact in ways that heavily influence system performance, task allocation, and the emergence of cooperative behavior.

1.3 Interaction of Variables and Simulation Outcomes

- **Tomato Spawning + Drone Scanning:** Collectors cannot act independently; the drone acts as the gatekeeper. This ensures orderly workflow and prevents premature collection.
- **A* Pathfinding + Map Agent Context:** When the environment becomes more populated or multiple robots converge near the same plant, path costs increase. Collectors must re-route, reducing collisions and improving flow.

- **Capacity Limit + Drop Logic:** When collectors fill up, they abandon ongoing targets and deliver tomatoes to their drop zone. This introduces temporary delays but increases global throughput.

- **Plant Exclusivity Rule:** Only one collector may approach a plant at a time. This reduces overlap and conflict but introduces queuing behaviors when spawn density increases.

Altogether, these interactions create a complex pattern of emergent cooperation, where agents adapt their actions based on system state, resource availability, and other agents' positions.

1.4 Why We Selected the Graphic Design

We selected this graphic design because it makes the system look easy to understand for both the agents and the observer. The use of cubic plant models and bright, color-coded agents: white robots for healthy collectors, a red robot for defective collectors, and a drone for the inspector; creates immediate visual distinction between roles, reducing confusion during simultaneous interactions. Tomatoes also use distinct colors to communicate health states instantly after detection, allowing us to monitor classification and collection patterns in real time. The simplicity of geometric shapes ensures that the simulation remains understandable even during caotic scenarios involving multiple agents navigating the grid. Overall, this minimalist visual style enhances clarity, reduces visual noise, and provides a functional and efficient interface for analyzing the behavior and interactions of the multi-agent system.

1.5 Advantages of the Final Solution

The final solution has several advantages that contribute to its robustness and scalability. The clear specialization of roles prevents agents from becoming overloaded, ensuring that each component focuses on a specific task and behaves predictably. Dynamic tomato spawning introduces continuous variability, providing a realistic challenge for the inspector and collectors while demonstrating the adaptability of the system. The integration of A* navigation with the shared map provided by the Map Agent allows for efficient and adaptive movement across the grid, even as new tomatoes appear or obstacles influence path costs. Having multiple collectors enables parallel harvesting, significantly reducing task backlog and improving throughput. Additionally, constraint-based rules, such as collector capacity limits and plant exclusivity, create realistic logistical behaviors and prevent conflicts. Overall, these elements make the system modular, scalable, and easy to extend for future development.

1.6 Disadvantages and Limitations

Despite its strengths, the system also presents several limitations. The Inspector Drone acts as the main bottleneck: if it fails to detect tomatoes quickly enough, the collectors are forced to remain idle, reducing overall efficiency. Navigation conflicts may still arise under high density tomato spawning, particularly when multiple collectors attempt to approach nearby plants, despite the exclusivity rule. Capacity constraints can lead to congestion near drop zones when collectors reach their limit and must unload simultaneously. Furthermore, the Map Agent is static and cannot respond to unexpected dynamic events beyond the initial environmental configuration, limiting adaptability. Finally, tomato classification relies on probability rather than real perception, meaning the simulation does not fully capture the complexity of visual detection in real agricultural systems. These constraints highlight areas where the system can be improved.

1.7 Modifications for Improvement

- Add multiple inspector drones to reduce detection bottlenecks.
- Implement localized avoidance and negotiation between collectors to enhance path realism.
- Introduce dynamic map updates for temporary obstacles.
- Replace probability-based classification with image-based classification models.
- Implement a priority-based queue for collectors to reduce plant contention.

2. Reflection on My Learning Process

At the beginning of the block, my expectations included learning how to build a Unity environment or simulation, understanding what a multi-agent system is and how it works, and creating a functioning simulation capable of detecting and managing anomalies in crops. As stated in our initial expectations document, my personal goal was to strengthen my ability to debug system performance and to better understand how autonomous behaviors, navigation, and internal agent coordination could be implemented in practice.

Throughout the project, my understanding of MAS improved significantly. I moved from viewing agents as isolated components to recognizing the importance of coordination, communication, and the dynamics that emerge when multiple autonomous entities share the same environment. Working with a static Map Agent, a mobile Inspector Drone, and several

Collector Robots allowed me to experience firsthand how clearly defined responsibilities and architectural decisions shape the overall system behavior.

In the same way, from a technical perspective, I gained practical experience in Unity through grid construction, object instantiation, collision management, A* pathfinding, and communication between agents. Observing how the system reacted to dynamically spawned tomatoes helped me understand how computational models interact and how complex system level behaviors can emerge from relatively simple local rules.

Also, on the personal and collaborative side, this project strengthened my debugging skills, attention to detail, and ability to systematically test performance under different scenarios. It also highlighted my area of opportunity regarding documentation and technical presentation, encouraging me to improve how I structure and communicate results. Comparing my initial expectations with what I achieved, I can confidently say that I developed not only stronger technical abilities but also a deeper capacity for systemic thinking.