Tecnológico de Monterrey

Guadalajara campus

Evidence 2. Review 2

José Yael Varela García - A01645324

Faisal Alali- A01830963

Sebastian Certuche - A01644942

Sebastián Alejandro Soria Piñuela - A01645849

Danilo Paolo Tato Velázquez - A01644630

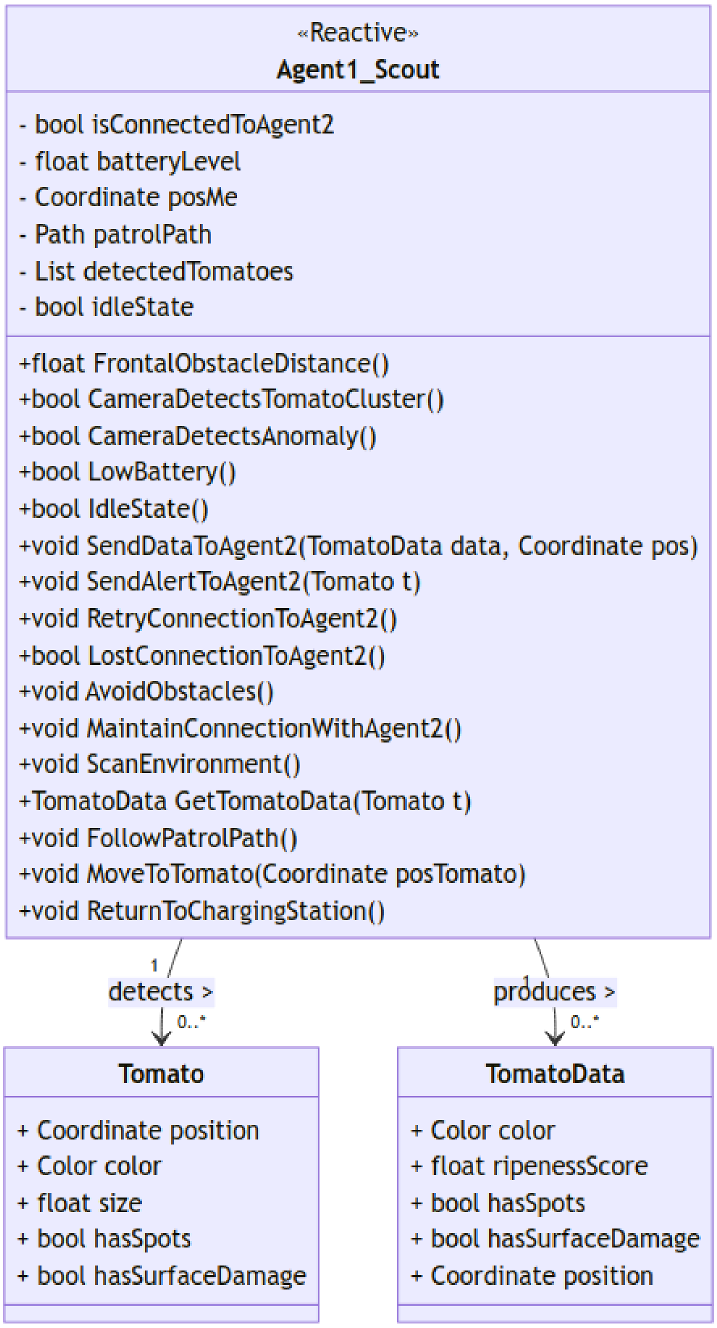
Modeling of multi-agent systems with computers graphics

Group 301

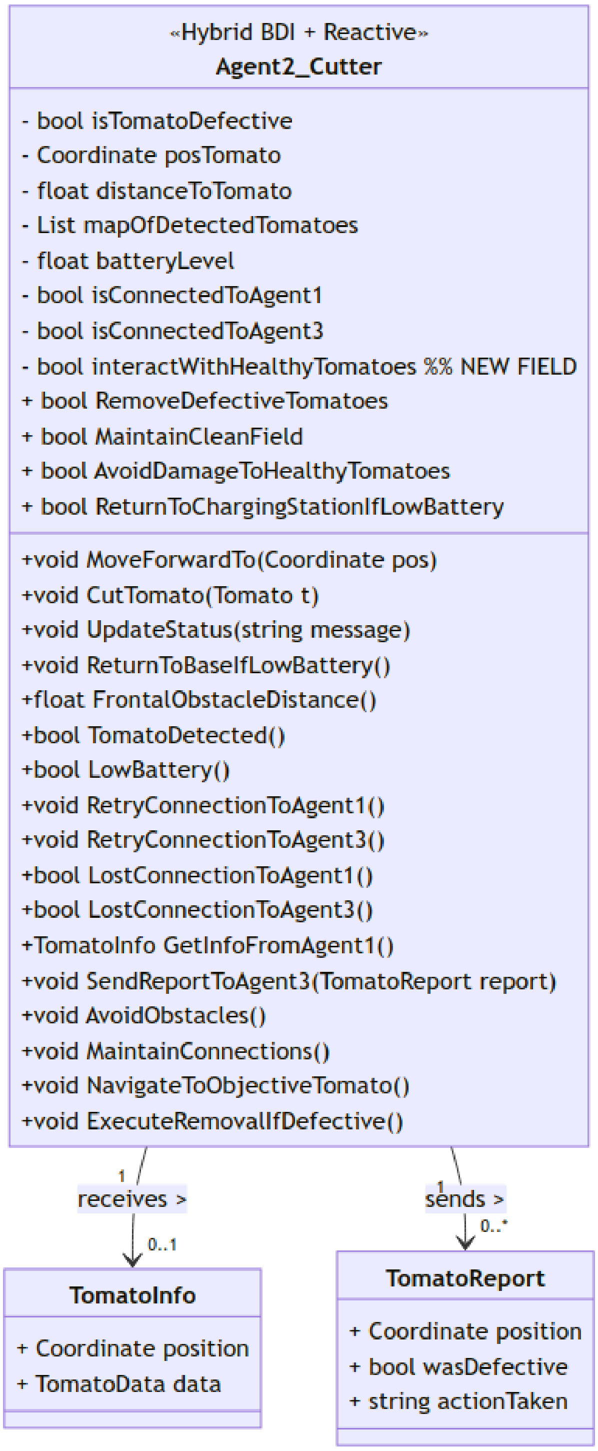
November 18, 2025

Multiagent system

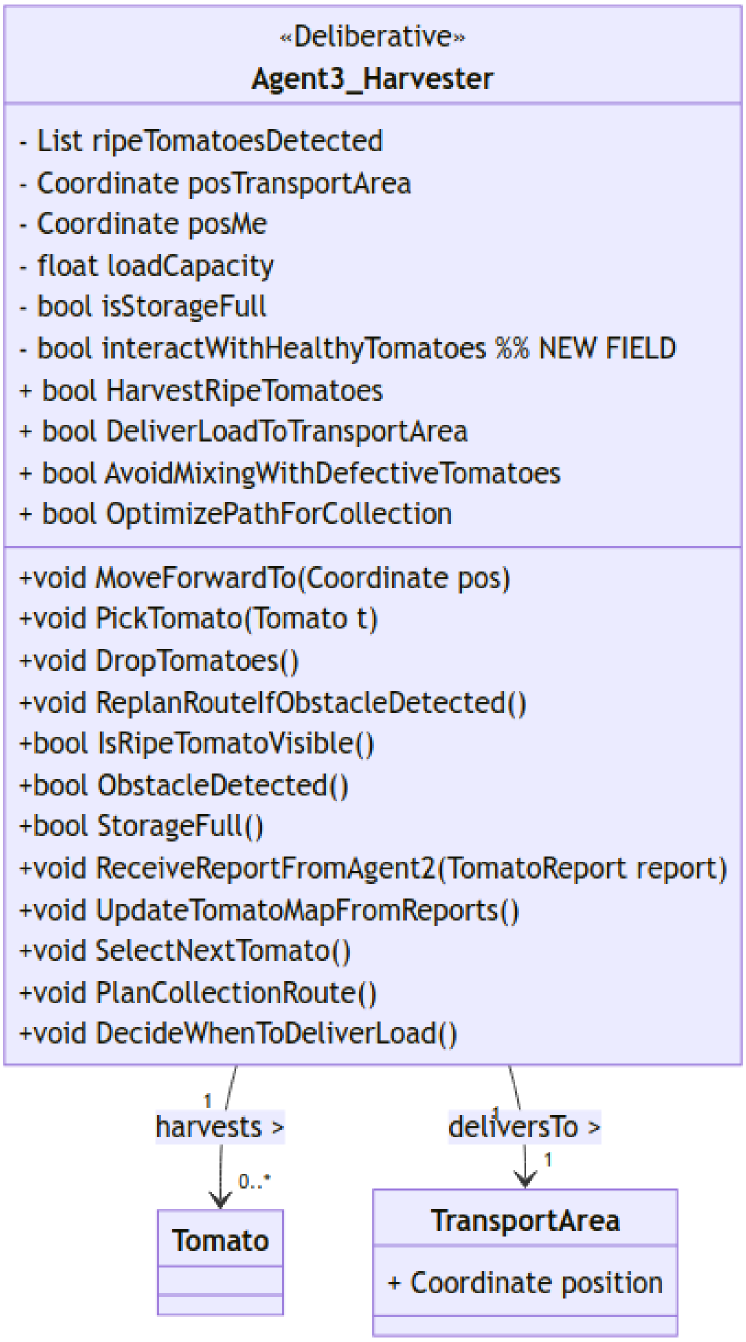
Agent Class Diagrams for each agent type/role.



Agent1_Scout is a reactive agent that patrols the environment, avoids obstacles, detects tomatoes and anomalies, processes them into TomatoData, and communicates this information to Agent 2 while managing its own battery and connection status; Tomato represents raw tomato attributes, TomatoData stores processed inspection data, and the scout may detect and generate many such objects.


«Hybrid BDI + Reactive»
Agent2_Cutter
- bool isTomatoDefective
- Coordinate posTomato
- float distanceToTomato
- List mapOfDetectedTomatoes
- float batteryLevel
- bool isConnectedToAgent1
- bool isConnectedToAgent3
- bool interactWithHealthyTomatoes %% NEW FIELD
+ bool RemoveDefectiveTomatoes
+ bool MaintainCleanField
+ bool AvoidDamageToHealthyTomatoes
+ bool ReturnToChargingStationIfLowBattery
+void MoveForwardTo(Coordinate pos)
+void CutTomato(Tomato t)
+void UpdateStatus(string message)
+void ReturnToBaseIfLowBattery()
+float FrontalObstacleDistance()
+bool TomatoDetected()
+bool LowBattery()
+void RetryConnectionToAgent1()
+void RetryConnectionToAgent3()
+bool LostConnectionToAgent1()
+bool LostConnectionToAgent3()
+TomatoInfo GetInfoFromAgent1()
+void SendReportToAgent3(TomatoReport report)
+void AvoidObstacles()
+void MaintainConnections()
+void NavigateToObjectiveTomato()
+void ExecuteRemovalIfDefective()
1
receives >
0..1
1
sends >
0..*
TomatoInfo
+ Coordinate position
+ TomatoData data
TomatoReport
+ Coordinate position
+ bool wasDefective
+ string actionTaken

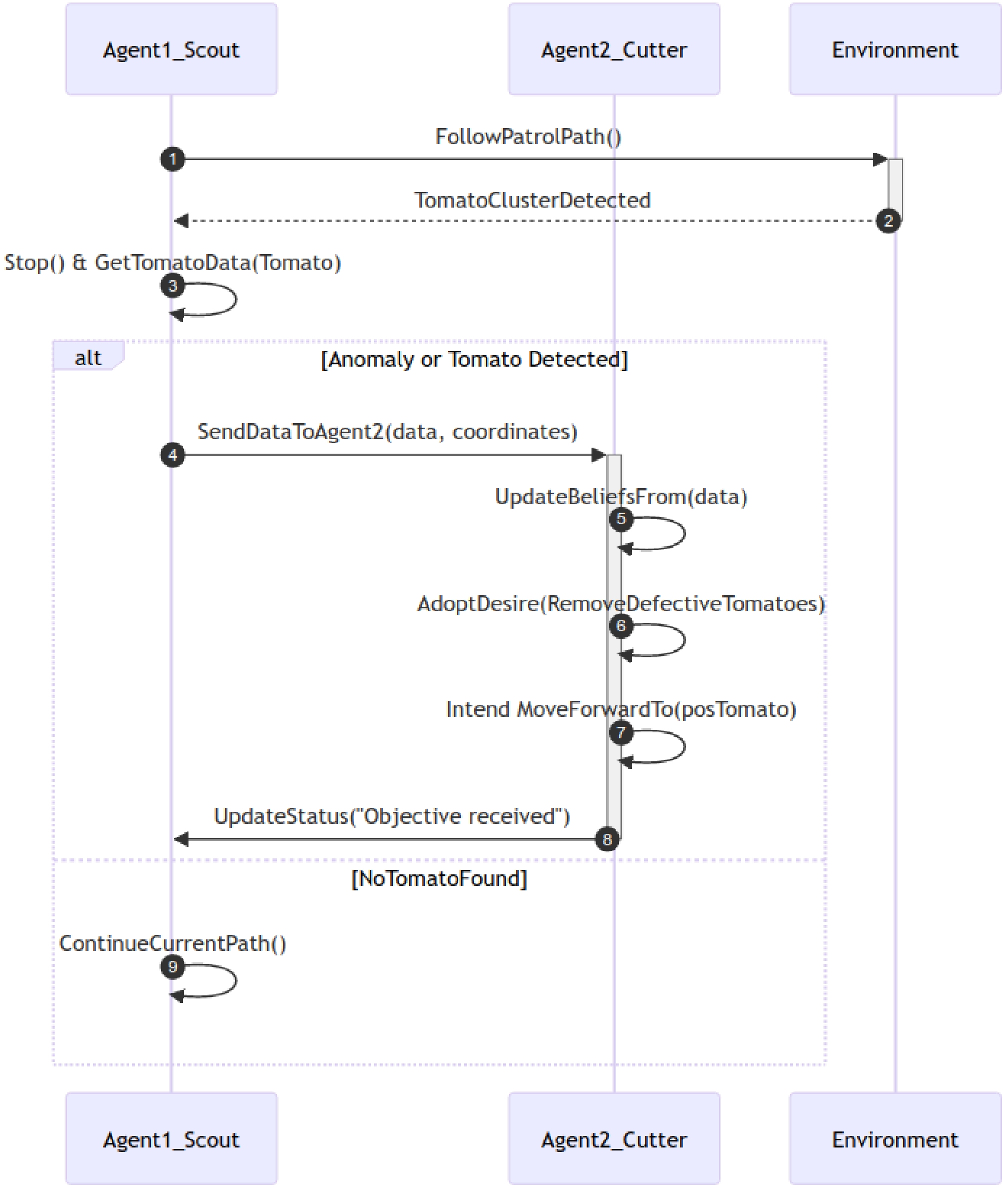
Agent2_Cutter is a hybrid BDI-reactive agent that receives processed tomato information, navigates toward the target, determines whether the tomato is defective, removes it if necessary, avoids obstacles, manages communication with Agents 1 and 3, updates its beliefs and intentions based on incoming data, and sends TomatoReport messages back to Agent 3; TomatoInfo provides the incoming inspection data from Agent 1, while TomatoReport represents the status updates the cutter sends out, with the agent receiving at most one TomatoInfo at a time and producing any number of reports.


«Deliberative»
Agent3_Harvester
- List ripeTomatoesDetected
- Coordinate posTransportArea
- Coordinate posMe
- float loadCapacity
- bool isStorageFull
- bool interactWithHealthyTomatoes %% NEW FIELD
+ bool HarvestRipeTomatoes
+ bool DeliverLoadToTransportArea
+ bool AvoidMixingWithDefectiveTomatoes
+ bool OptimizePathForCollection
+void MoveForwardTo(Coordinate pos)
+void PickTomato(Tomato t)
+void DropTomatoes()
+void ReplanRouteIfObstacleDetected()
+bool IsRipeTomatoVisible()
+bool ObstacleDetected()
+bool StorageFull()
+void ReceiveReportFromAgent2(TomatoReport report)
+void UpdateTomatoMapFromReports()
+void SelectNextTomato()
+void PlanCollectionRoute()
+void DecideWhenToDeliverLoad()
1
harvests >
0..*
Tomato
1
deliversTo >
1
TransportArea
+ Coordinate position

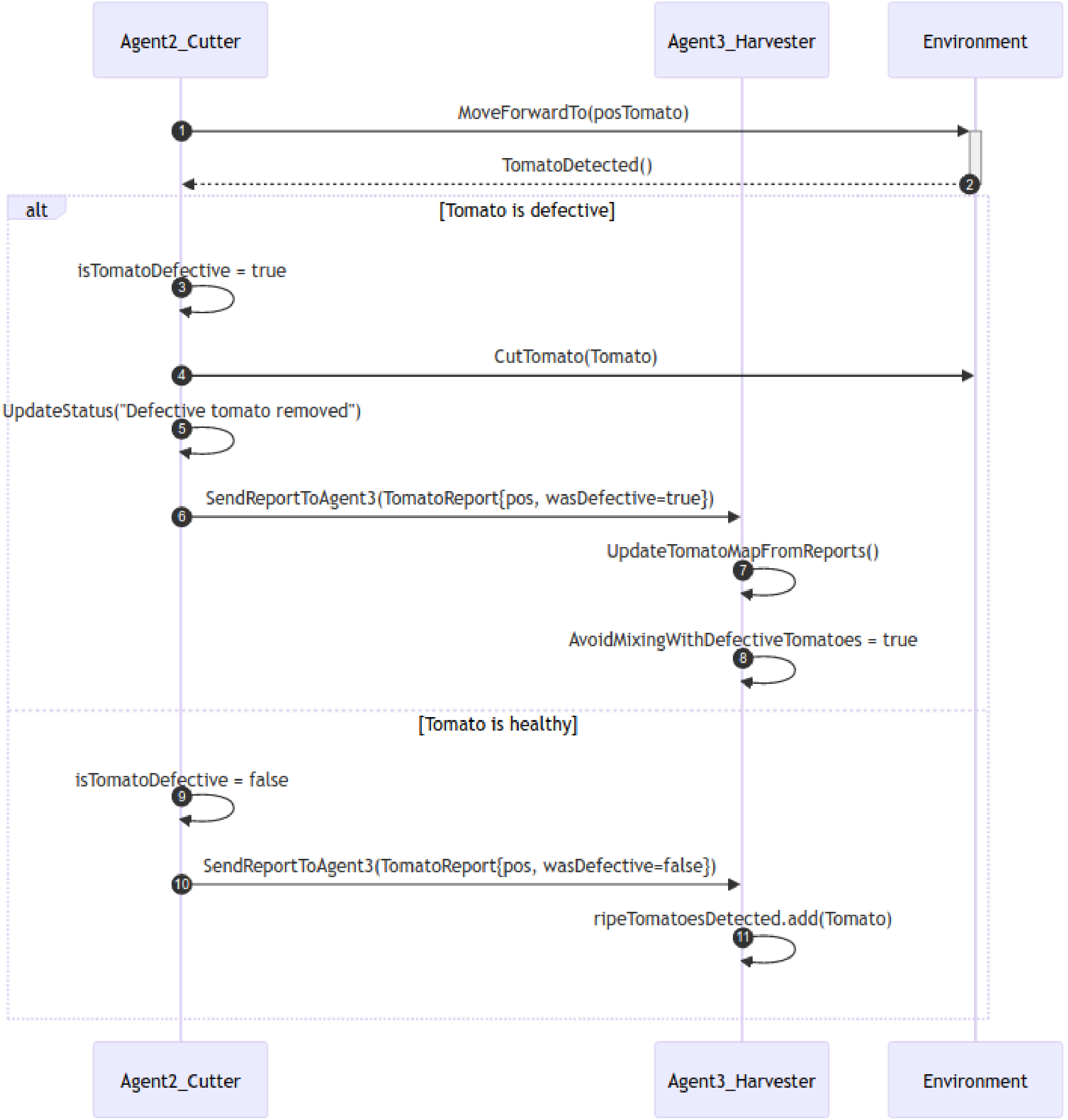
Agent3_Harvester is a deliberative agent that plans and executes the collection and delivery of ripe tomatoes, maintains an updated map of tomato conditions using reports from Agent 2, selects optimal routes, avoids mixing defective tomatoes, manages its load capacity, and transports harvested tomatoes to a designated TransportArea; it can detect ripe tomatoes, avoid obstacles, decide when to unload, and may harvest many tomatoes while always delivering to exactly one transport location.

Interaction Diagrams (AIP) for all the interactions considered in your simulation.

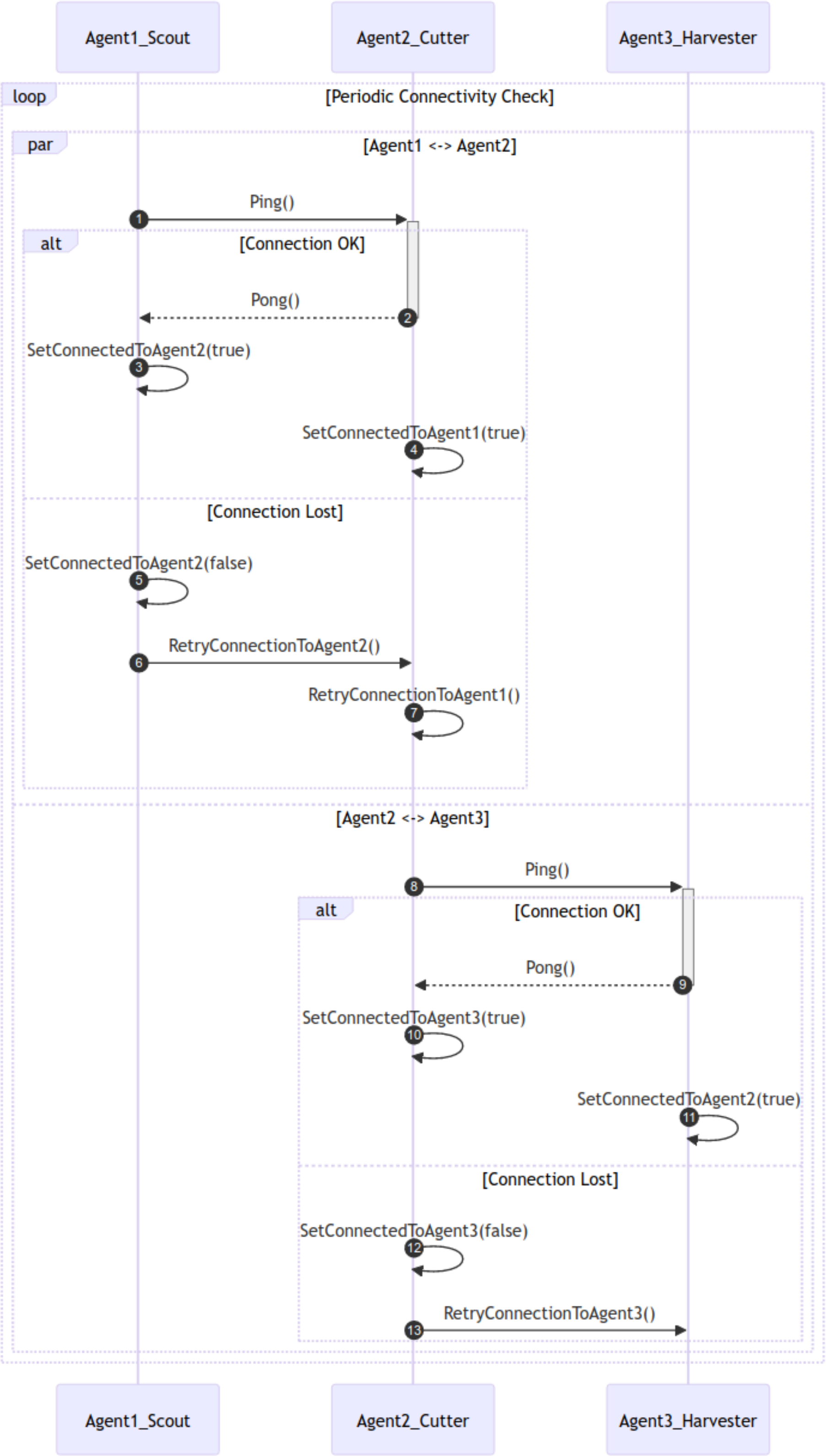
Every relationship between agents is 1 to1.


Agent1_Scout
Agent2_Cutter
Environment
FollowPatrolPath()
1
TomatoClusterDetected
2
Stop() & GetTomatoData(Tomato)
3
alt
[Anomaly or Tomato Detected]
SendDataToAgent2(data, coordinates)
4
UpdateBeliefsFrom(data)
5
AdoptDesire(RemoveDefectiveTomatoes)
6
Intend MoveForwardTo(posTomato)
7
UpdateStatus("Objective received")
8
[NoTomatoFound]
ContinueCurrentPath()
9
Agent1_Scout
Agent2_Cutter
Environment

This interaction diagram shows how Agent1_Scout patrols the environment, detects a tomato cluster, extracts tomato data, and sends it to Agent2_Cutter, which then updates its beliefs, adopts a goal to remove defective tomatoes, forms an intention to move toward the tomato's position, and replies with a status update, while the alternative branch shows Agent1 simply continuing its patrol if no tomato is found.


Agent2_Cutter
Agent3_Harvester
Environment
MoveForwardTo(posTomato)
1
TomatoDetected()
2
alt
[Tomato is defective]
isTomatoDefective = true
3
CutTomato(Tomato)
4
UpdateStatus("Defective tomato removed")
5
SendReportToAgent3(TomatoReport{pos, wasDefective=true})
6
UpdateTomatoMapFromReports()
7
AvoidMixingWithDefectiveTomatoes = true
8
[Tomato is healthy]
isTomatoDefective = false
9
SendReportToAgent3(TomatoReport{pos, wasDefective=false})
10
ripeTomatoesDetected.add(Tomato)
11
Agent2_Cutter
Agent3_Harvester
Environment

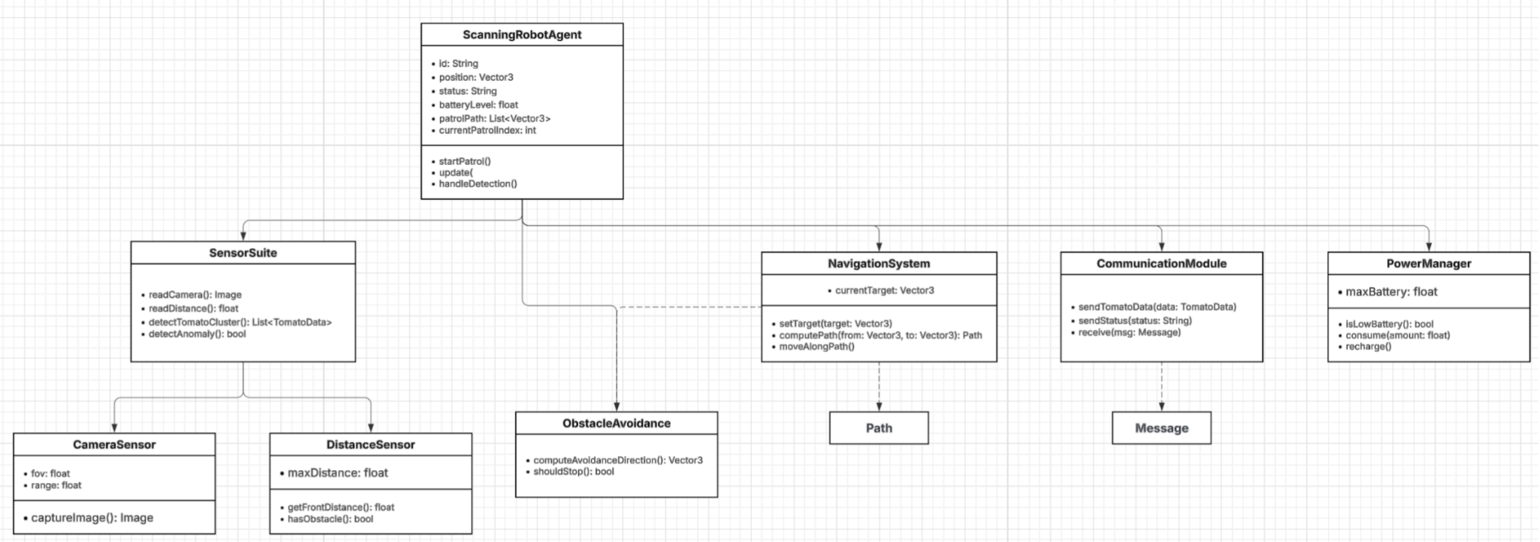
This interaction diagram shows how Agent2_Cutter moves toward a detected tomato, determines whether it is defective, performs the cutting action if necessary, and reports the result to Agent3_Harvester, which then updates its internal tomato map—either marking the tomato as defective and adjusting its harvesting policy or adding it to the list of ripe tomatoes when it is healthy.


Agent1_Scout
Agent2_Cutter
Agent3_Harvester
loop
[Periodic Connectivity Check]
par
[Agent1 <-> Agent2]
1 Ping()
alt
[Connection OK]
2 Pong()
3 SetConnectedToAgent2(true)
4 SetConnectedToAgent1(true)
[Connection Lost]
5 SetConnectedToAgent2(false)
6 RetryConnectionToAgent2()
7 RetryConnectionToAgent1()
[Agent2 <-> Agent3]
8 Ping()
alt
[Connection OK]
9 Pong()
10 SetConnectedToAgent3(true)
11 SetConnectedToAgent2(true)
[Connection Lost]
12 SetConnectedToAgent3(false)
13 RetryConnectionToAgent3()
Agent1_Scout
Agent2_Cutter
Agent3_Harvester

This interaction diagram describes the periodic connectivity checks between the three agents, where Agent1 and Agent2 exchange ping-pong messages to confirm communication, Agent2 and Agent3 do the same, and whenever a connection is lost the corresponding agent updates its connection status and initiates a retry procedure to restore communication before continuing normal operation.

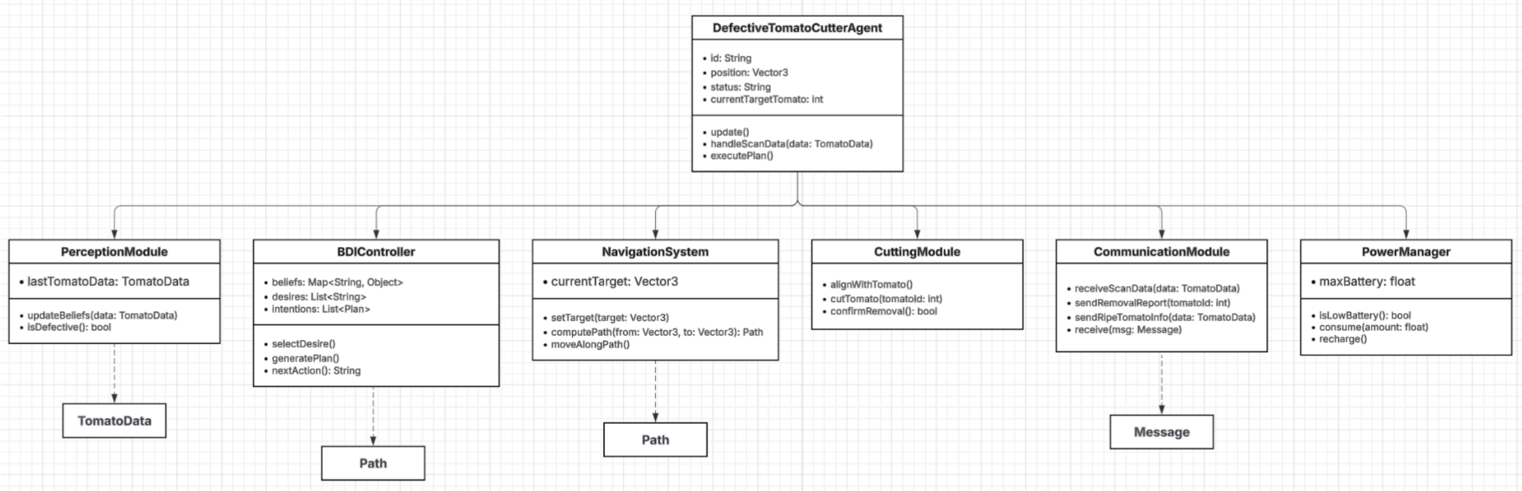
Standard Class Diagrams to describe the agents' subsystems.

ScanningRobotAgent Subsystem



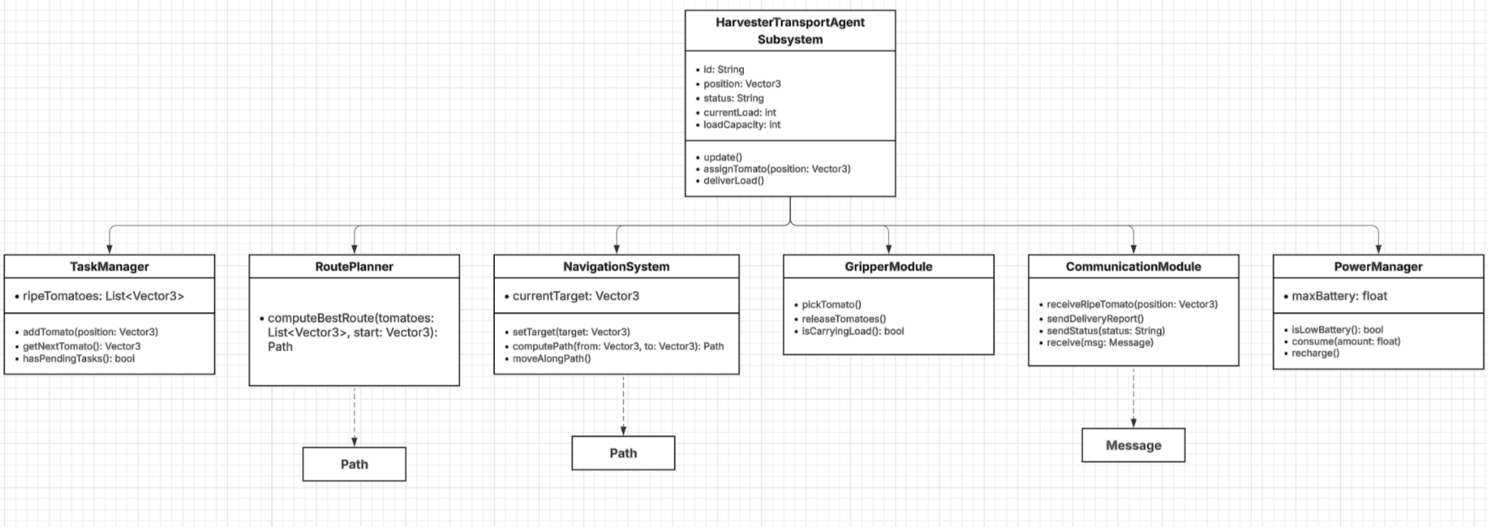
The ScanningRobotAgent subsystem includes the internal components that allow the agent to patrol the environment, perceive tomato clusters, detect anomalies, and report findings to other agents. Its SensorSuite, composed of a CameraSensor and DistanceSensor, provides visual and range information used for cluster detection and obstacle awareness. The NavigationSystem manages movement across the field, assisted by the ObstacleAvoidance module, which computes safe directions and prevents collisions. The CommunicationModule enables the agent to send tomato data and status messages to other agents, while the PowerManager monitors battery levels and regulates energy usage. Together, these modules provide the sensing, mobility, and communication capabilities required for the agent's autonomous scanning behavior.

DefectiveTomatoCutterAgent Subsystem



The DefectiveTomatoCutterAgent subsystem models the internal architecture that allows the agent to analyze incoming tomato data, classify defects, and perform cutting actions. The PerceptionModule processes information received from the scanning agent and updates the agent’s beliefs, while the BDIController manages reasoning, goal selection, and plan generation. A NavigationSystem enables movement toward target tomatoes, and the CuttingModule performs precise alignment and removal of defective items. The CommunicationModule handles the exchange of messages, including scan data, removal confirmations, and notifications for ripe tomatoes. The PowerManager supervises the agent’s energy usage. Altogether, these components support autonomous decision-making and execution of defect-removal tasks.

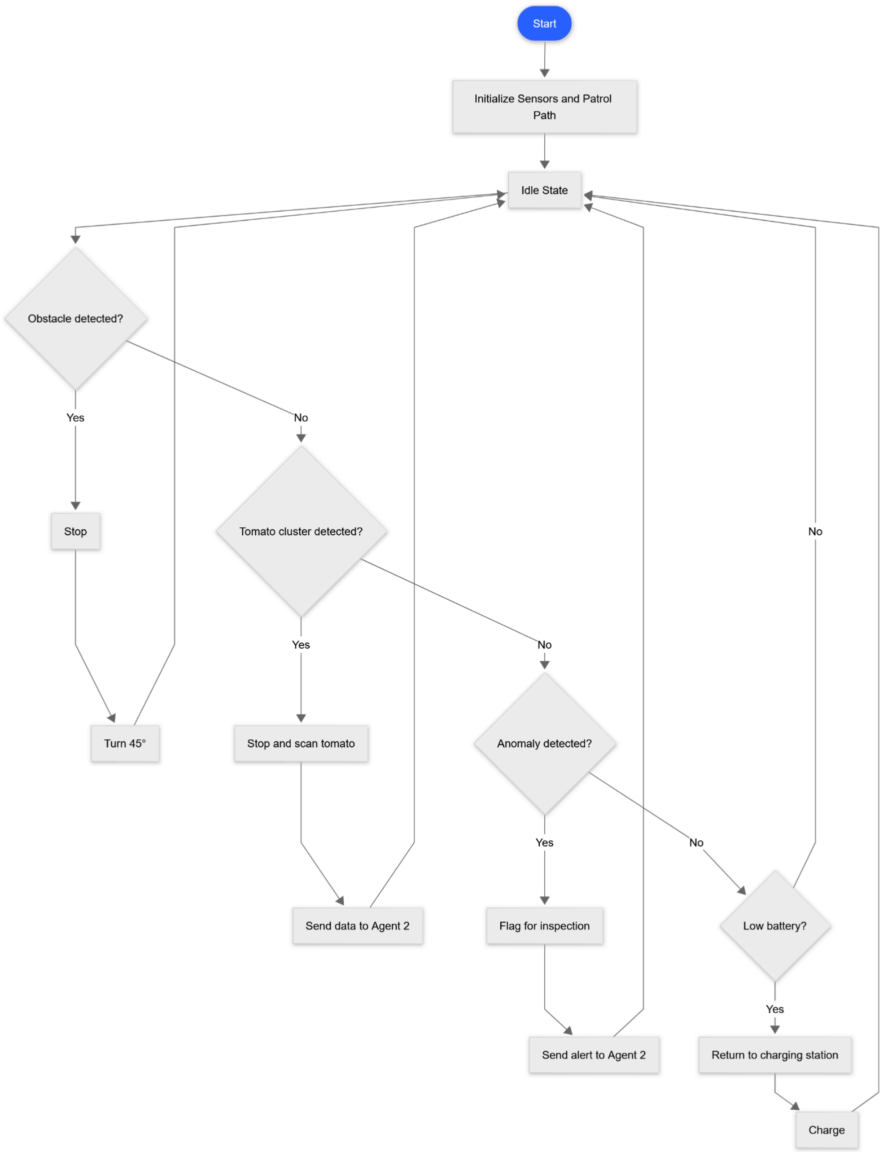
HarvesterTransportAgent Subsystem



The HarvesterTransportAgent subsystem contains the components responsible for receiving ripe-tomato locations, planning harvesting routes, collecting tomatoes, and delivering them to the transport area. The TaskManager stores and organizes harvesting tasks, while the RoutePlanner computes efficient paths. The NavigationSystem guides the agent along these routes, and the GripperModule enables physical picking and dropping of tomatoes. A CommunicationModule manages message exchange with other agents, including receiving ripe-tomato positions and reporting completed deliveries. The PowerManager controls battery usage to ensure continuous operation. Collectively, these modules provide the coordination, mobility, and manipulation capabilities needed for autonomous harvesting and transport tasks within the multi-agent system.

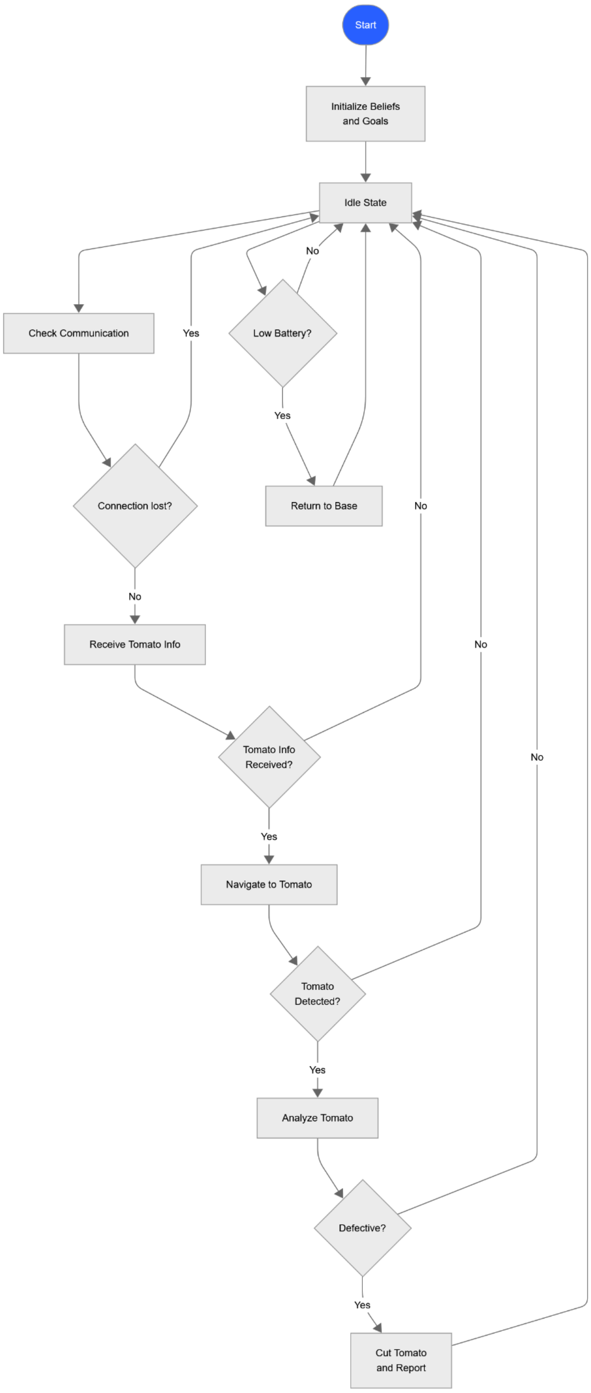
Activity or State Diagrams to describe the behavior of the agents' subsystems.

Agent 1



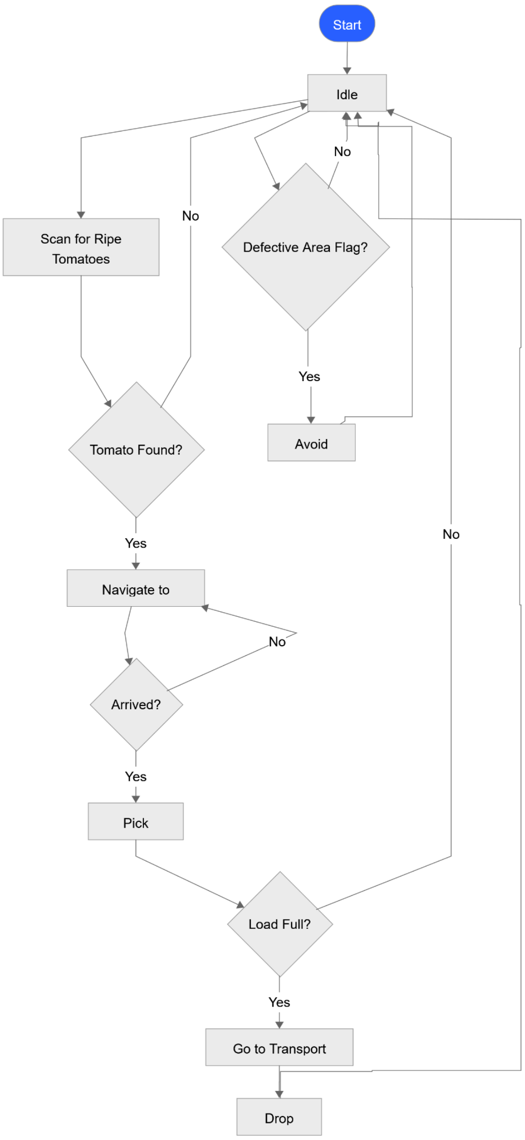
The Scanning Robot is an agent responsible for navigating the crop field, detecting tomatoes, and identifying possible infections among the crops. It follows a predefined patrol path while continuously monitoring its surroundings to avoid hypothetical collisions. The agent prioritizes obstacle avoidance to ensure safe movement, then maintains communication with the other agents to share detection events. When a tomato cluster or a potential anomaly is detected, the robot temporarily stops its patrol to scan the object and send the collected data to Agent 2 for further analysis. If its battery becomes low, the robot automatically returns to the charging station before resuming its tasks.

Agent 2



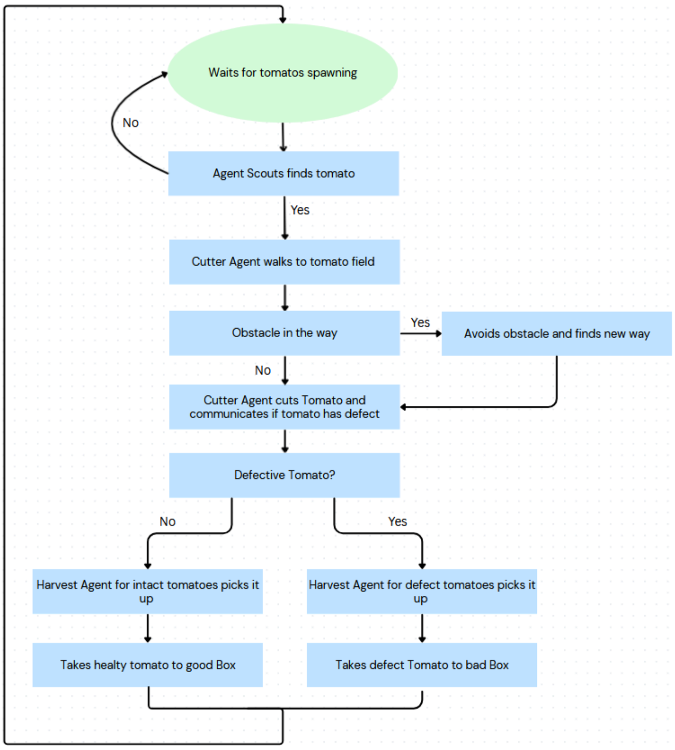
Agent 2 is an agent that receives information from the Scanning Robot and determines whether a tomato is healthy or defective, then the agent evaluates each tomato and decides whether removal is necessary. Once a defective tomato is confirmed, it moves to the location, cuts and removes it, and then reports the action to Agent 3. Its reactive layers also handle obstacle avoidance and communication maintenance.

Agent 3



The agent 3 focuses on the collection of tomatoes and transporting them to the designated delivery area. It plans and follows a path, navigates the environment, and picks tomatoes while managing its load capacity to avoid mixing with defective ones. When its storage is full, it delivers the collected tomatoes before returning to pick more crops. It also responds to anomaly reports from Agent 2 to avoid contaminated areas that could be entirely cut if the infection propagates.

General sequence of Agents (flow chart)


Waits for tomatos spawning
No
Agent Scouts finds tomato
Yes
Cutter Agent walks to tomato field
Obstacle in the way
Yes
Avoids obstacle and finds new way
No
Cutter Agent cuts Tomato and communicates if tomato has defect
Defective Tomato?
No
Yes
Harvest Agent for intact tomatoes picks it up
Harvest Agent for defect tomatoes picks it up
Takes healty tomato to good Box
Takes defect Tomato to bad Box

This diagram is a general description of how the work flow of the agents will look like and how they need to happen. Serving as a much more easy to read and to understand diagram that would simplify the concepts for the whole team.

Part 2 - Computer graphics

How the Virtual World Is Visualized

The virtual world is implemented in Unity and visualized through a stylized, grid-based environment dominated by sharp edges and cubic geometry. The entire scene is arranged on a modular grid, which provides both structural clarity and a consistent spatial reference for the agents’ navigation. The aesthetic emphasizes an earthy, organic atmosphere while maintaining a geometric, almost architectural discipline. Colors, textures, and lighting are chosen to highlight the contrast between natural forms and strict cubic compositions.

Key Visuals

The central theme is to create a strong visual language based on sharped edges and basic geometric shapes, contrasting with the interactable objects, which will strive to have a more organic feel.

The environment is composed of a set of recurring cubic elements that define its visual identity:

- **Cubic Terrain Blocks:** The ground and main structural layout are built from cube modules. These blocks serve as the foundation on which all other objects are positioned.
- **Tomato Plant Structures:** One of the most visually distinctive components is the tomato plant. Each plant is represented as a sequence of sharply-turning segments arranged into an inward-facing cubic spiral. These angular shapes reinforce the world's geometric theme.

  Inside these structures grow round, smooth tomatoes, intentionally contrasting with the rigidity of the cubic plant structure.
- **Bushes and Floral Elements:** Vegetation other than tomato plants has a simplified, almost 2D cubic appearance—flat, planar shapes with minimal depth. Their simplicity adds variety while maintaining consistency with the sharp-edge theme.
- **Crates and Rock Cubes:** Scattered throughout the environment, crates and rock-like cube objects provide obstacles, navigational markers, and environmental richness. Their textures and colors reinforce the earthy tone of the world.
- **Fences:** Though their material and color may vary, they help give the environment an organic and rural feel.

Visualization of Agents and Other Object Models

The agents operating in this environment will also follow the cubic and sharp-edged design philosophy. They are represented using primary shapes that are linked to their role. Their animations, if present, emphasize abrupt, straight motions with lots of impact and snappy movements.

Any additional interactive objects follow the same principle: clear silhouettes with circular shapes that contrast with the sharpness of the environment, trying to be transparent with the viewer with what objects will be relevant to the agents.

Schematics and Sketches

Schematics accompanying this description illustrate the grid arrangement, the cubic tomato-plant spiral, the simple flattened bushes, etc. These drawings help convey the intended contrast between sharp-edged architecture and organic elements, while also clarifying how agents move and interact within the structured environment.