**Team 6: Sample Data Examples**

**Realistic Data Flows Between All Components**

**Table of Contents**

1. [Core Data Structures](https://claude.ai/chat/6de0fb32-38a2-40c0-b003-470028f959a0#1-core-data-structures)
2. [Sanidhaya's Data Examples](https://claude.ai/chat/6de0fb32-38a2-40c0-b003-470028f959a0#2-sanidhaya-core-agent-system)
3. [John's Data Examples](https://claude.ai/chat/6de0fb32-38a2-40c0-b003-470028f959a0#3-john-communication-system)
4. [Lauren's Data Examples](https://claude.ai/chat/6de0fb32-38a2-40c0-b003-470028f959a0#4-lauren-swarm-intelligence)
5. [Anthony's Data Examples](https://claude.ai/chat/6de0fb32-38a2-40c0-b003-470028f959a0#5-anthony-user-interface)
6. [Complete Scenario Walkthrough](https://claude.ai/chat/6de0fb32-38a2-40c0-b003-470028f959a0#6-complete-scenario-walkthrough)
7. [JSON Format Examples](https://claude.ai/chat/6de0fb32-38a2-40c0-b003-470028f959a0#7-json-format-examples)

**1. Core Data Structures**

**1.1 Basic Agent State Example**

// Example of what a typical agent looks like at any moment

AgentState agent\_005 = new AgentState();

agent\_005.agentId = 5;

agent\_005.agentName = "Agent\_005";

agent\_005.position = new Point2D(245.7, 156.3);

agent\_005.velocity = new Vector2D(12.5, -8.2);

agent\_005.heading = 2.1; // radians

agent\_005.maxSpeed = 50.0;

agent\_005.maxTurnRate = 1.5;

agent\_005.communicationRange = 100.0;

agent\_005.status = AgentStatus.ACTIVE;

agent\_005.batteryLevel = 0.87; // 87% battery

agent\_005.lastUpdateTime = 1635789432156L; // timestamp

agent\_005.currentTask = new Task("patrol\_waypoint\_3");

agent\_005.teamId = 2; // part of formation team 2

**1.2 Sample Swarm Configuration**

// What a typical 5-agent swarm looks like during operation

List<AgentState> currentSwarm = Arrays.asList(

// Leader agent

new AgentState(1, "Leader", new Point2D(200, 200), AgentStatus.ACTIVE, 0.95),

// Follower agents in loose formation

new AgentState(2, "Scout\_A", new Point2D(180, 190), AgentStatus.ACTIVE, 0.82),

new AgentState(3, "Scout\_B", new Point2D(220, 190), AgentStatus.ACTIVE, 0.91),

new AgentState(4, "Guard\_A", new Point2D(190, 210), AgentStatus.ACTIVE, 0.76),

new AgentState(5, "Guard\_B", new Point2D(210, 210), AgentStatus.ACTIVE, 0.88)

);

**2. Sanidhaya - Core Agent System**

**2.1 Input Data Examples**

**From Anthony - System Commands**

// User clicks to spawn a new agent

SystemCommand spawnCommand = new SystemCommand();

spawnCommand.type = CommandType.SPAWN\_AGENT;

spawnCommand.timestamp = System.currentTimeMillis();

spawnCommand.parameters = Map.of(

"position", new Point2D(150.0, 250.0),

"agentType", "SCOUT",

"maxSpeed", 45.0,

"communicationRange", 120.0,

"teamId", 1

);

// User adjusts simulation speed

SystemCommand speedCommand = new SystemCommand();

speedCommand.type = CommandType.SET\_SIMULATION\_SPEED;

speedCommand.parameters = Map.of("speedMultiplier", 1.5);

// User sets boundary limits

SystemCommand boundaryCommand = new SystemCommand();

boundaryCommand.type = CommandType.SET\_BOUNDARIES;

boundaryCommand.parameters = Map.of(

"minX", 0.0, "maxX", 800.0,

"minY", 0.0, "maxY", 600.0

);

**From Lauren - Movement Commands**

// Basic flocking behavior command

MovementCommand flockingCmd = new MovementCommand();

flockingCmd.agentId = 3;

flockingCmd.type = MovementType.FLOCKING\_BEHAVIOR;

flockingCmd.priority = CommandPriority.NORMAL;

flockingCmd.timestamp = System.currentTimeMillis();

flockingCmd.parameters = Map.of(

"separationWeight", 1.8,

"alignmentWeight", 1.2,

"cohesionWeight", 1.0,

"avoidanceRadius", 25.0,

"targetSpeed", 30.0

);

// Formation flying command

MovementCommand formationCmd = new MovementCommand();

formationCmd.agentId = 7;

formationCmd.type = MovementType.FORMATION\_POSITION;

formationCmd.priority = CommandPriority.HIGH;

formationCmd.parameters = Map.of(

"formationType", "LINE",

"formationCenter", new Point2D(300, 300),

"formationSpacing", 40.0,

"positionIndex", 2 // 3rd position in line

);

// Emergency avoidance command

MovementCommand emergencyCmd = new MovementCommand();

emergencyCmd.agentId = 4;

emergencyCmd.type = MovementType.AVOID\_OBSTACLE;

emergencyCmd.priority = CommandPriority.EMERGENCY;

emergencyCmd.parameters = Map.of(

"obstaclePosition", new Point2D(200, 150),

"obstacleRadius", 50.0,

"avoidanceForce", 2.0

);

**From John - Communication Events**

// Agent receives a message notification

CommunicationEvent messageEvent = new CommunicationEvent();

messageEvent.receiverAgentId = 5;

messageEvent.senderAgentId = 3;

messageEvent.message = new Message();

messageEvent.message.type = MessageType.VOTE\_PROPOSAL;

messageEvent.message.payload = "Navigation decision: go left or right around obstacle?";

messageEvent.deliverySuccess = true;

messageEvent.timestamp = System.currentTimeMillis();

// Network topology update

NetworkUpdate topologyUpdate = new NetworkUpdate();

topologyUpdate.agentId = 6;

topologyUpdate.currentNeighbors = Arrays.asList(4, 7, 8);

topologyUpdate.lostNeighbors = Arrays.asList(2); // agent 2 moved out of range

topologyUpdate.newNeighbors = Arrays.asList(8); // agent 8 moved into range

**2.2 Output Data Examples**

**To John - Agent State Updates**

// Regular position update (sent 30-60 times per second)

AgentStateUpdate positionUpdate = new AgentStateUpdate();

positionUpdate.agentId = 7;

positionUpdate.newPosition = new Point2D(267.3, 198.7);

positionUpdate.newVelocity = new Vector2D(15.2, -5.8);

positionUpdate.newHeading = 1.96;

positionUpdate.status = AgentStatus.ACTIVE;

positionUpdate.batteryLevel = 0.83;

positionUpdate.timestamp = System.currentTimeMillis();

// Status change update

AgentStateUpdate statusUpdate = new AgentStateUpdate();

statusUpdate.agentId = 4;

statusUpdate.status = AgentStatus.BATTERY\_LOW; // battery running low

statusUpdate.batteryLevel = 0.15;

statusUpdate.timestamp = System.currentTimeMillis();

**To Lauren - Capability Information**

// What agent 5 can currently do

AgentCapabilities capabilities\_5 = new AgentCapabilities();

capabilities\_5.agentId = 5;

capabilities\_5.maxSpeed = 50.0;

capabilities\_5.maxTurnRate = 1.5;

capabilities\_5.currentSpeed = 32.7;

capabilities\_5.currentTurnRate = 0.8;

capabilities\_5.canCommunicate = true;

capabilities\_5.canMove = true;

capabilities\_5.efficiency = 0.91; // performing at 91% efficiency

// Task completion report

TaskCompletionReport taskReport = new TaskCompletionReport();

taskReport.agentId = 3;

taskReport.taskId = 12;

taskReport.status = TaskStatus.COMPLETED;

taskReport.completionPercentage = 100.0;

taskReport.statusMessage = "Reached waypoint successfully";

**To Anthony - Visualization Data**

// Complete system state for visualization (sent 30 times per second)

VisualizationUpdate vizUpdate = new VisualizationUpdate();

vizUpdate.allAgents = Arrays.asList(

new AgentState(1, new Point2D(200, 200), AgentStatus.ACTIVE, 0.95),

new AgentState(2, new Point2D(180, 190), AgentStatus.ACTIVE, 0.82),

new AgentState(3, new Point2D(220, 190), AgentStatus.ACTIVE, 0.91),

new AgentState(4, new Point2D(190, 210), AgentStatus.BATTERY\_LOW, 0.15),

new AgentState(5, new Point2D(210, 210), AgentStatus.ACTIVE, 0.88)

);

vizUpdate.systemMetrics = new SystemMetrics();

vizUpdate.systemMetrics.totalAgents = 5;

vizUpdate.systemMetrics.activeAgents = 4; // one has low battery

vizUpdate.systemMetrics.averageSpeed = 28.5;

vizUpdate.systemMetrics.systemLoad = 0.45; // 45% CPU usage

vizUpdate.systemMetrics.updatesPerSecond = 35;

vizUpdate.systemMetrics.memoryUsage = 2.1; // 2.1 GB

vizUpdate.recentEvents = Arrays.asList(

new SystemEvent("agent\_spawned", "Agent 5 created at position (210, 210)"),

new SystemEvent("battery\_warning", "Agent 4 battery level below 20%"),

new SystemEvent("formation\_complete", "Line formation established")

);

vizUpdate.timestamp = System.currentTimeMillis();

**3. John - Communication System**

**3.1 Input Data Examples**

**From Sanidhaya - Agent State Updates**

// John receives these to calculate communication ranges

List<AgentStateUpdate> receivedUpdates = Arrays.asList(

new AgentStateUpdate(1, new Point2D(200, 200), new Vector2D(10, 0), AgentStatus.ACTIVE),

new AgentStateUpdate(2, new Point2D(180, 190), new Vector2D(8, 5), AgentStatus.ACTIVE),

new AgentStateUpdate(3, new Point2D(350, 180), new Vector2D(-5, 2), AgentStatus.ACTIVE), // far away

new AgentStateUpdate(4, new Point2D(190, 210), new Vector2D(0, -3), AgentStatus.BATTERY\_LOW),

new AgentStateUpdate(5, new Point2D(210, 210), new Vector2D(-2, -1), AgentStatus.ACTIVE)

);

// John calculates: agents 1, 2, 4, 5 can all talk to each other (within 100 unit range)

// Agent 3 is too far away (150+ units from others)

**From Lauren - Messages to Send**

// Voting proposal from agent 1 to all nearby agents

OutgoingMessage voteProposal = new OutgoingMessage();

voteProposal.senderId = 1;

voteProposal.receiverId = -1; // broadcast

voteProposal.priority = MessagePriority.HIGH;

voteProposal.maxHops = 2;

voteProposal.expirationTime = System.currentTimeMillis() + 10000; // expires in 10 seconds

voteProposal.messageContent = new Message();

voteProposal.messageContent.messageId = "vote\_001";

voteProposal.messageContent.type = MessageType.VOTE\_PROPOSAL;

voteProposal.messageContent.payload = new VoteProposal(

"obstacle\_navigation",

"Large obstacle detected ahead. How should swarm navigate?",

Arrays.asList("GO\_LEFT", "GO\_RIGHT", "GO\_OVER"),

System.currentTimeMillis() + 8000 // voting deadline

);

// Task assignment from swarm coordinator

OutgoingMessage taskAssignment = new OutgoingMessage();

taskAssignment.senderId = 1; // leader agent

taskAssignment.receiverId = 3; // specific agent

taskAssignment.messageContent = new Message();

taskAssignment.messageContent.type = MessageType.TASK\_ASSIGNMENT;

taskAssignment.messageContent.payload = new TaskAssignment(

"patrol\_sector\_alpha",

"Patrol the northern sector for 5 minutes",

new Rectangle(100, 50, 200, 100), // patrol area

TaskPriority.NORMAL

);

// Position update broadcast

OutgoingMessage positionUpdate = new OutgoingMessage();

positionUpdate.senderId = 2;

positionUpdate.receiverId = -1; // broadcast

positionUpdate.priority = MessagePriority.LOW; // routine update

positionUpdate.messageContent = new Message();

positionUpdate.messageContent.type = MessageType.POSITION\_UPDATE;

positionUpdate.messageContent.payload = new PositionData(

new Point2D(185, 195),

new Vector2D(12, -3),

System.currentTimeMillis()

);

**From Anthony - Network Configuration**

// User adjusts communication settings

NetworkConfiguration networkConfig = new NetworkConfiguration();

networkConfig.communicationRange = 120.0; // increased from default 100

networkConfig.messageLatency = 150; // 150ms simulated delay

networkConfig.failureRate = 0.05; // 5% of messages fail

networkConfig.interferenceLevel = 0.2; // 20% interference reduces range

networkConfig.enableMultiHop = true; // agents can relay messages

networkConfig.maxRetries = 3;

**3.2 Output Data Examples**

**To Sanidhaya - Communication Events**

// Successful message delivery notification

CommunicationEvent successEvent = new CommunicationEvent();

successEvent.receiverAgentId = 4;

successEvent.senderAgentId = 1;

successEvent.message = /\* the vote proposal from above \*/;

successEvent.deliverySuccess = true;

successEvent.timestamp = System.currentTimeMillis();

// Failed message delivery (agent out of range)

CommunicationEvent failEvent = new CommunicationEvent();

failEvent.receiverAgentId = 3; // the far away agent

failEvent.senderAgentId = 1;

failEvent.deliverySuccess = false;

failEvent.timestamp = System.currentTimeMillis();

**To Lauren - Incoming Messages**

// Vote response received by agent 1

IncomingMessage voteResponse = new IncomingMessage();

voteResponse.receiverId = 1; // leader receiving the vote

voteResponse.originalSenderId = 4;

voteResponse.routePath = Arrays.asList(4, 2, 1); // routed through agent 2

voteResponse.signalStrength = 0.85; // 85% signal strength

voteResponse.actualDeliveryTime = System.currentTimeMillis();

voteResponse.messageContent = new Message();

voteResponse.messageContent.type = MessageType.VOTE\_RESPONSE;

voteResponse.messageContent.payload = new VoteResponse(

"vote\_001", // responding to proposal

"GO\_LEFT", // agent 4 votes to go left

4, // voter ID

"Obstacle too large to go over, left path seems clearer"

);

// Neighbor information for agent 2

NeighborInformation neighborInfo = new NeighborInformation();

neighborInfo.agentId = 2;

neighborInfo.neighbors = Arrays.asList(

new NeighborAgent(1, 22.3, 0.92, true, System.currentTimeMillis()), // agent 1, distance 22.3

new NeighborAgent(4, 31.6, 0.87, true, System.currentTimeMillis()), // agent 4, distance 31.6

new NeighborAgent(5, 45.2, 0.78, true, System.currentTimeMillis()) // agent 5, distance 45.2

);

**To Anthony - Network Status**

// Current network status for visualization

NetworkStatus networkStatus = new NetworkStatus();

networkStatus.totalConnections = 8; // 8 active communication links

networkStatus.messagesPerSecond = 12.5;

networkStatus.averageLatency = 145.0; // milliseconds

networkStatus.health = NetworkHealth.GOOD;

networkStatus.activeConnections = Arrays.asList(

new ConnectionInfo(1, 2, 0.92, true, System.currentTimeMillis() - 5000),

new ConnectionInfo(1, 4, 0.89, true, System.currentTimeMillis() - 3000),

new ConnectionInfo(1, 5, 0.85, true, System.currentTimeMillis() - 2000),

new ConnectionInfo(2, 4, 0.87, true, System.currentTimeMillis() - 4000),

new ConnectionInfo(2, 5, 0.91, true, System.currentTimeMillis() - 1000),

new ConnectionInfo(4, 5, 0.88, true, System.currentTimeMillis() - 6000)

// Note: agent 3 is not connected to anyone (too far away)

);

networkStatus.recentMessages = Arrays.asList(

new MessageLog("vote\_001", 1, -1, MessageType.VOTE\_PROPOSAL, true, System.currentTimeMillis() - 2000),

new MessageLog("vote\_001\_resp", 4, 1, MessageType.VOTE\_RESPONSE, true, System.currentTimeMillis() - 1500),

new MessageLog("task\_assign\_3", 1, 3, MessageType.TASK\_ASSIGNMENT, false, System.currentTimeMillis() - 1000) // failed

);

**4. Lauren - Swarm Intelligence**

**4.1 Input Data Examples**

**From John - Incoming Messages**

// Lauren receives vote responses to process

IncomingMessage vote1 = createVoteResponse("vote\_001", 2, "GO\_LEFT", "Path looks clear");

IncomingMessage vote2 = createVoteResponse("vote\_001", 4, "GO\_LEFT", "Agree with agent 2");

IncomingMessage vote3 = createVoteResponse("vote\_001", 5, "GO\_RIGHT", "Right side has less obstacles");

// Lauren receives neighbor information for flocking calculations

NeighborInformation flockingData = new NeighborInformation();

flockingData.agentId = 3;

flockingData.neighbors = Arrays.asList(

new NeighborAgent(1, 45.2, 0.88, true, System.currentTimeMillis()),

new NeighborAgent(2, 62.1, 0.75, true, System.currentTimeMillis()),

new NeighborAgent(4, 38.7, 0.91, true, System.currentTimeMillis())

);

// Emergency message received

IncomingMessage emergency = new IncomingMessage();

emergency.messageContent = new Message();

emergency.messageContent.type = MessageType.EMERGENCY\_ALERT;

emergency.messageContent.payload = new EmergencyAlert(

"OBSTACLE\_COLLISION\_IMMINENT",

new Point2D(250, 200), // danger location

50.0, // danger radius

"Large obstacle detected, immediate avoidance required"

);

**From Sanidhaya - Agent Capabilities**

// Current capabilities of all agents for decision making

List<AgentCapabilities> swarmCapabilities = Arrays.asList(

new AgentCapabilities(1, 50.0, 1.5, 35.2, 0.8, true, true, 0.95), // leader, good condition

new AgentCapabilities(2, 45.0, 1.4, 28.1, 1.2, true, true, 0.87), // scout, active

new AgentCapabilities(3, 55.0, 1.6, 0.0, 0.0, false, false, 0.0), // FAILED agent

new AgentCapabilities(4, 48.0, 1.3, 31.8, 0.9, true, true, 0.76), // guard, low battery

new AgentCapabilities(5, 52.0, 1.7, 33.5, 1.1, true, true, 0.91) // guard, good condition

);

// Task completion reports

List<TaskCompletionReport> taskReports = Arrays.asList(

new TaskCompletionReport(2, 15, TaskStatus.COMPLETED, 100.0, "Waypoint reached successfully"),

new TaskCompletionReport(4, 16, TaskStatus.IN\_PROGRESS, 65.0, "En route to patrol zone"),

new TaskCompletionReport(5, 17, TaskStatus.FAILED, 0.0, "Cannot reach target due to obstacle")

);

**From Anthony - Behavior Configuration**

// User adjusts flocking parameters

FlockingParameters flockingParams = new FlockingParameters();

flockingParams.separationRadius = 30.0;

flockingParams.separationWeight = 2.0; // stronger separation

flockingParams.alignmentRadius = 50.0;

flockingParams.alignmentWeight = 1.2;

flockingParams.cohesionRadius = 80.0;

flockingParams.cohesionWeight = 1.0;

flockingParams.avoidanceRadius = 40.0;

flockingParams.maxSpeed = 45.0;

// User adjusts voting parameters

VotingParameters votingParams = new VotingParameters();

votingParams.consensusThreshold = 0.6; // 60% majority required

votingParams.votingTimeout = 8000; // 8 seconds to vote

votingParams.maxVotingRounds = 3;

votingParams.allowAbstention = true;

// User sets mission parameters

BehaviorConfiguration behaviorConfig = new BehaviorConfiguration();

behaviorConfig.flocking = flockingParams;

behaviorConfig.voting = votingParams;

behaviorConfig.missionPriority = MissionPriority.HIGH;

behaviorConfig.formationTightness = 0.8; // tight formations

**4.2 Output Data Examples**

**To Sanidhaya - Movement Commands**

// Flocking command for agent 2 based on neighbor calculations

MovementCommand flockingCommand = new MovementCommand();

flockingCommand.agentId = 2;

flockingCommand.type = MovementType.FLOCKING\_BEHAVIOR;

flockingCommand.priority = CommandPriority.NORMAL;

flockingCommand.parameters = Map.of(

"separationForce", new Vector2D(-5.2, 3.1), // avoid crowding

"alignmentForce", new Vector2D(8.7, -2.4), // match neighbors

"cohesionForce", new Vector2D(2.1, 1.8), // stay with group

"combinedForce", new Vector2D(5.6, 2.5), // total movement force

"targetSpeed", 32.0

);

// Formation command after voting decision

MovementCommand formationCommand = new MovementCommand();

formationCommand.agentId = 4;

formationCommand.type = MovementType.FORMATION\_POSITION;

formationCommand.priority = CommandPriority.HIGH;

formationCommand.parameters = Map.of(

"formationType", "WEDGE",

"formationCenter", new Point2D(300, 250),

"positionIndex", 3, // 4th position in wedge

"spacing", 35.0,

"targetPosition", new Point2D(285, 265) // calculated position

);

// Emergency avoidance command

MovementCommand emergencyCommand = new MovementCommand();

emergencyCommand.agentId = 5;

emergencyCommand.type = MovementType.AVOID\_OBSTACLE;

emergencyCommand.priority = CommandPriority.EMERGENCY;

emergencyCommand.parameters = Map.of(

"avoidanceForce", new Vector2D(-15.0, 8.0), // strong avoidance

"urgencyLevel", "HIGH",

"obstaclePosition", new Point2D(250, 200),

"safeDirection", new Vector2D(-1.0, 0.5)

);

**To John - Messages to Send**

// New voting proposal after obstacle detection

OutgoingMessage newVoteProposal = new OutgoingMessage();

newVoteProposal.senderId = 1; // leader initiates

newVoteProposal.receiverId = -1; // broadcast

newVoteProposal.messageContent = new Message();

newVoteProposal.messageContent.type = MessageType.VOTE\_PROPOSAL;

newVoteProposal.messageContent.payload = new VoteProposal(

"formation\_change\_002",

"Current formation inefficient for navigation. Change formation?",

Arrays.asList("KEEP\_CURRENT", "SWITCH\_TO\_LINE", "SWITCH\_TO\_COLUMN"),

System.currentTimeMillis() + 10000

);

// Task reassignment after agent failure

OutgoingMessage taskReassignment = new OutgoingMessage();

taskReassignment.senderId = 1;

taskReassignment.receiverId = 5; // reassign to agent 5

taskReassignment.messageContent = new Message();

taskReassignment.messageContent.type = MessageType.TASK\_ASSIGNMENT;

taskReassignment.messageContent.payload = new TaskAssignment(

"patrol\_sector\_beta",

"Take over patrol duties from failed agent 3",

new Rectangle(150, 100, 180, 120),

TaskPriority.HIGH

);

// Coordination message for formation

OutgoingMessage coordinationMsg = new OutgoingMessage();

coordinationMsg.senderId = 1;

coordinationMsg.receiverId = -1; // broadcast

coordinationMsg.messageContent = new Message();

coordinationMsg.messageContent.type = MessageType.FORMATION\_COMMAND;

coordinationMsg.messageContent.payload = new FormationCommand(

"WEDGE",

new Point2D(350, 300), // formation center

40.0, // spacing

Arrays.asList(1, 2, 4, 5), // participating agents (3 failed)

"MOVE\_FORWARD" // formation movement direction

);

**To Anthony - Decision Status**

// Current voting status

DecisionStatus votingStatus = new DecisionStatus();

votingStatus.decisionId = "vote\_001";

votingStatus.type = DecisionType.VOTING;

votingStatus.state = DecisionState.VOTING\_IN\_PROGRESS;

votingStatus.startTime = System.currentTimeMillis() - 3000; // started 3 seconds ago

votingStatus.estimatedCompletion = System.currentTimeMillis() + 5000; // 5 seconds remaining

votingStatus.currentData = Map.of(

"question", "Navigate around obstacle: left or right?",

"options", Arrays.asList("GO\_LEFT", "GO\_RIGHT"),

"votesReceived", 3,

"totalVoters", 4, // agent 3 failed, can't vote

"currentTally", Map.of("GO\_LEFT", 2, "GO\_RIGHT", 1),

"consensusThreshold", 0.6

);

// Task allocation status

DecisionStatus taskStatus = new DecisionStatus();

taskStatus.decisionId = "task\_reallocation\_001";

taskStatus.type = DecisionType.TASK\_ALLOCATION;

taskStatus.state = DecisionState.EXECUTING;

taskStatus.currentData = Map.of(

"failedAgent", 3,

"reassignedTo", 5,

"newTaskLoad", Map.of(

"agent\_1", 2, // 2 active tasks

"agent\_2", 1, // 1 active task

"agent\_4", 2, // 2 active tasks

"agent\_5", 3 // 3 active tasks (took over from agent 3)

)

);

// Current behavior status for each agent

List<BehaviorStatus> behaviorStatuses = Arrays.asList(

new BehaviorStatus(1, BehaviorType.LEADER, Map.of("decisionsMade", 5, "effectiveness", 0.92)),

new BehaviorStatus(2, BehaviorType.FLOCKING, Map.of("separationActive", true, "effectiveness", 0.87)),

new BehaviorStatus(4, BehaviorType.FORMATION, Map.of("positionError", 12.3, "effectiveness", 0.81)),

new BehaviorStatus(5, BehaviorType.TASK\_EXECUTION, Map.of("tasksActive", 3, "effectiveness", 0.76))

);

**5. Anthony - User Interface**

**5.1 Input Data Examples**

**From Sanidhaya - Visualization Data**

// Real-time visualization update (received 30 times per second)

VisualizationUpdate realtimeUpdate = new VisualizationUpdate();

realtimeUpdate.timestamp = System.currentTimeMillis();

// Current agent positions and status

realtimeUpdate.allAgents = Arrays.asList(

createAgentState(1, 205.3, 198.7, 0.95, AgentStatus.ACTIVE, "Leader"),

createAgentState(2, 182.1, 189.2, 0.82, AgentStatus.ACTIVE, "Scout"),

createAgentState(3, 0.0, 0.0, 0.0, AgentStatus.FAILED, "Failed"), // failed agent

createAgentState(4, 193.8, 212.4, 0.18, AgentStatus.BATTERY\_LOW, "Guard"),

createAgentState(5, 217.2, 207.9, 0.89, AgentStatus.ACTIVE, "Guard")

);

// System performance metrics

realtimeUpdate.systemMetrics = new SystemMetrics();

realtimeUpdate.systemMetrics.totalAgents = 5;

realtimeUpdate.systemMetrics.activeAgents = 3; // 1 failed, 1 low battery

realtimeUpdate.systemMetrics.averageSpeed = 29.7;

realtimeUpdate.systemMetrics.systemLoad = 0.52; // 52% CPU

realtimeUpdate.systemMetrics.updatesPerSecond = 32;

realtimeUpdate.systemMetrics.memoryUsage = 2.3; // GB

// Recent system events for event log

realtimeUpdate.recentEvents = Arrays.asList(

new SystemEvent("AGENT\_FAILED", "Agent 3 failed due to system error", System.currentTimeMillis() - 5000),

new SystemEvent("BATTERY\_WARNING", "Agent 4 battery level critical (18%)", System.currentTimeMillis() - 3000),

new SystemEvent("VOTE\_COMPLETED", "Navigation vote completed: GO\_LEFT wins", System.currentTimeMillis() - 1000)

);

**From John - Network Status**

// Network status for communication visualization

NetworkStatus currentNetwork = new NetworkStatus();

currentNetwork.totalConnections = 6; // active communication links

currentNetwork.messagesPerSecond = 8.3;

currentNetwork.averageLatency = 142.0; // milliseconds

currentNetwork.health = NetworkHealth.FAIR; // degraded due to failed agent

// Active communication links for drawing

currentNetwork.activeConnections = Arrays.asList(

new ConnectionInfo(1, 2, 0.91, true, System.currentTimeMillis() - 2000), // leader to scout

new ConnectionInfo(1, 4, 0.67, true, System.currentTimeMillis() - 1000), // leader to guard (weak signal)

new ConnectionInfo(1, 5, 0.88, true, System.currentTimeMillis() - 500), // leader to guard

new ConnectionInfo(2, 4, 0.72, true, System.currentTimeMillis() - 3000), // scout to guard

new ConnectionInfo(2, 5, 0.85, true, System.currentTimeMillis() - 1500), // scout to guard

new ConnectionInfo(4, 5, 0.93, true, System.currentTimeMillis() - 800) // guard to guard

// Note: agent 3 has no connections (failed)

);

// Recent message activity for monitoring

currentNetwork.recentMessages = Arrays.asList(

new MessageLog("vote\_001", 1, -1, MessageType.VOTE\_PROPOSAL, true, System.currentTimeMillis() - 4000),

new MessageLog("vote\_resp\_2", 2, 1, MessageType.VOTE\_RESPONSE, true, System.currentTimeMillis() - 3500),

new MessageLog("vote\_resp\_4", 4, 1, MessageType.VOTE\_RESPONSE, true, System.currentTimeMillis() - 3200),

new MessageLog("vote\_resp\_5", 5, 1, MessageType.VOTE\_RESPONSE, true, System.currentTimeMillis() - 3000),

new MessageLog("formation\_cmd", 1, -1, MessageType.FORMATION\_COMMAND, true, System.currentTimeMillis() - 1000)

);

**From Lauren - Decision Status**

// Active voting process for display

DecisionStatus activeVoting = new DecisionStatus();

activeVoting.decisionId = "obstacle\_navigation\_003";

activeVoting.type = DecisionType.VOTING;

activeVoting.state = DecisionState.CONSENSUS\_REACHED;

activeVoting.startTime = System.currentTimeMillis() - 8000;

activeVoting.estimatedCompletion = System.currentTimeMillis() + 2000;

activeVoting.currentData = Map.of(

"question", "Formation change needed for narrow passage?",

"options", Arrays.asList("SINGLE\_FILE", "KEEP\_CURRENT", "SPLIT\_GROUP"),

"finalDecision", "SINGLE\_FILE",

"voteResults", Map.of("SINGLE\_FILE", 3, "KEEP\_CURRENT", 0, "SPLIT\_GROUP", 1),

"consensusLevel", 0.75 // 75% agreement

);

// Mission progress status

DecisionStatus missionStatus = new DecisionStatus();

missionStatus.decisionId = "patrol\_mission\_alpha";

missionStatus.type = DecisionType.MISSION\_PLANNING;

missionStatus.state = DecisionState.EXECUTING;

missionStatus.currentData = Map.of(

"missionType", "AREA\_PATROL",

"completionPercentage", 67.5,

"waypointsVisited", 8,

"waypointsRemaining", 4,

"estimatedTimeRemaining", 180000, // 3 minutes

"participatingAgents", Arrays.asList(1, 2, 4, 5) // agent 3 failed

);

// Current behavior analysis

List<BehaviorStatus> currentBehaviors = Arrays.asList(

new BehaviorStatus(1, BehaviorType.LEADER,

Map.of("decisions\_per\_minute", 2.3, "success\_rate", 0.89, "leadership\_effectiveness", 0.91)),

new BehaviorStatus(2, BehaviorType.SCOUT,

Map.of("exploration\_efficiency", 0.84, "information\_gathered", 15, "flocking\_compliance", 0.92)),

new BehaviorStatus(4, BehaviorType.GUARD,

Map.of("formation\_accuracy", 0.73, "battery\_concern", true, "performance\_degraded", true)),

new BehaviorStatus(5, BehaviorType.GUARD,

Map.of("formation\_accuracy", 0.88, "task\_load", 3, "stress\_level", 0.65))

);

**5.2 Output Data Examples**

**To Sanidhaya - System Commands**

// User clicks to spawn new agent

SystemCommand userSpawnCommand = new SystemCommand();

userSpawnCommand.type = CommandType.SPAWN\_AGENT;

userSpawnCommand.timestamp = System.currentTimeMillis();

userSpawnCommand.parameters = Map.of(

"position", new Point2D(350, 280), // where user clicked

"agentType", "REPLACEMENT", // replacing failed agent

"maxSpeed", 48.0,

"communicationRange", 110.0,

"initialBattery", 1.0 // full battery

);

// User adjusts simulation parameters

SystemCommand parameterCommand = new SystemCommand();

parameterCommand.type = CommandType.SET\_AGENT\_PARAMETER;

parameterCommand.parameters = Map.of(

"targetAgentId", 4, // the low battery agent

"parameter", "maxSpeed",

"newValue", 25.0 // reduced speed to conserve battery

);

// User sets new boundaries after map change

SystemCommand boundaryCommand = new SystemCommand();

boundaryCommand.type = CommandType.SET\_BOUNDARIES;

boundaryCommand.parameters = Map.of(

"minX", 50.0, "maxX", 750.0,

"minY", 50.0, "maxY", 550.0,

"safeZones", Arrays.asList(

new Rectangle(100, 100, 150, 100), // safe zone 1

new Rectangle(500, 300, 100, 150) // safe zone 2

)

);

**To John - Network Configuration**

// User adjusts communication settings

NetworkConfiguration userNetworkConfig = new NetworkConfiguration();

userNetworkConfig.communicationRange = 130.0; // increased range

userNetworkConfig.messageLatency = 100; // reduced latency

userNetworkConfig.failureRate = 0.02; // improved reliability (2% failure)

userNetworkConfig.interferenceLevel = 0.1; // reduced interference

userNetworkConfig.enableMultiHop = true;

userNetworkConfig.maxRetries = 2;

// User enables debug mode

NetworkConfiguration debugConfig = new NetworkConfiguration();

debugConfig.enableLogging = true;

debugConfig.logLevel = "DETAILED";

debugConfig.visualizeMessagePaths = true;

debugConfig.showSignalStrength = true;

**To Lauren - Behavior Configuration**

// User adjusts flocking behavior after observing performance

BehaviorConfiguration userBehaviorConfig = new BehaviorConfiguration();

// More aggressive flocking for tighter formation

userBehaviorConfig.flocking = new FlockingParameters();

userBehaviorConfig.flocking.separationRadius = 25.0; // closer together

userBehaviorConfig.flocking.separationWeight = 2.5; // stronger separation

userBehaviorConfig.flocking.alignmentWeight = 1.8; // stronger alignment

userBehaviorConfig.flocking.cohesionWeight = 1.5; // stronger cohesion

userBehaviorConfig.flocking.maxSpeed = 40.0;

// Faster decision making for dynamic environment

userBehaviorConfig.voting = new VotingParameters();

userBehaviorConfig.voting.consensusThreshold = 0.5; // simple majority

userBehaviorConfig.voting.votingTimeout = 5000; // faster decisions (5 seconds)

userBehaviorConfig.voting.maxVotingRounds = 2;

// Mission priority adjustment

userBehaviorConfig.taskAllocation = new TaskAllocationParameters();

userBehaviorConfig.taskAllocation.loadBalancing = true;

userBehaviorConfig.taskAllocation.failureRecoveryTime = 3000; // 3 seconds to reassign

userBehaviorConfig.taskAllocation.prioritizeHealthyAgents = true;

**User Interface Events**

// User clicks to place waypoint

UserEvent waypointPlacement = new UserEvent();

waypointPlacement.type = EventType.PLACE\_WAYPOINT;

waypointPlacement.clickPosition = new Point2D(420, 180);

waypointPlacement.timestamp = System.currentTimeMillis();

waypointPlacement.parameters = Map.of(

"waypointType", "CHECKPOINT",

"priority", "HIGH",

"radius", 30.0,

"dwellTime", 5000 // wait 5 seconds at waypoint

);

// User selects agent for detailed information

UserEvent agentSelection = new UserEvent();

agentSelection.type = EventType.SELECT\_AGENT;

agentSelection.clickPosition = new Point2D(193.8, 212.4); // agent 4's position

agentSelection.parameters = Map.of(

"selectedAgentId", 4,

"showDetails", true,

"followAgent", false

);

// User starts new mission

UserEvent missionStart = new UserEvent();

missionStart.type = EventType.START\_MISSION;

missionStart.parameters = Map.of(

"missionType", "SEARCH\_AND\_RESCUE",

"searchArea", new Rectangle(200, 150, 300, 250),

"searchPattern", "SPIRAL",

"participatingAgents", Arrays.asList(1, 2, 5), // exclude failed and low battery agents

"timeLimit", 600000 // 10 minutes

);

**6. Complete Scenario Walkthrough**

**6.1 Scenario: Obstacle Avoidance with Voting**

**Initial State:**

* 5 agents in loose formation moving toward waypoint
* Agent 3 suddenly fails (simulated system failure)
* Large obstacle detected in path

**Step 1: Agent Failure Detection (Sanidhaya → All)**

// Sanidhaya detects agent 3 failure

AgentStateUpdate failureUpdate = new AgentStateUpdate();

failureUpdate.agentId = 3;

failureUpdate.status = AgentStatus.FAILED;

failureUpdate.timestamp = System.currentTimeMillis();

// Broadcasted to all components

VisualizationUpdate vizUpdate = new VisualizationUpdate();

vizUpdate.recentEvents.add(

new SystemEvent("AGENT\_FAILURE", "Agent 3 system failure detected", System.currentTimeMillis())

);

**Step 2: Network Topology Update (John)**

// John removes failed agent from network

NetworkUpdate topologyUpdate = new NetworkUpdate();

topologyUpdate.agentId = 1; // updating leader's neighbors

topologyUpdate.lostNeighbors = Arrays.asList(3); // lost connection to agent 3

topologyUpdate.currentNeighbors = Arrays.asList(2, 4, 5); // remaining connections

**Step 3: Obstacle Detection & Voting Proposal (Lauren → John)**

// Lauren detects obstacle and initiates vote

OutgoingMessage obstacleVote = new OutgoingMessage();

obstacleVote.senderId = 1; // leader proposes

obstacleVote.receiverId = -1; // broadcast

obstacleVote.messageContent = new Message();

obstacleVote.messageContent.type = MessageType.VOTE\_PROPOSAL;

obstacleVote.messageContent.payload = new VoteProposal(

"obstacle\_avoid\_001",

"Large obstacle detected at (300, 200). Navigation options:",

Arrays.asList("GO\_LEFT", "GO\_RIGHT", "FORMATION\_CHANGE"),

System.currentTimeMillis() + 8000 // 8 second deadline

);

**Step 4: Vote Distribution (John → Lauren)**

// John delivers vote to remaining agents

List<IncomingMessage> voteDeliveries = Arrays.asList(

createVoteDelivery(2, "obstacle\_avoid\_001"), // to scout

createVoteDelivery(4, "obstacle\_avoid\_001"), // to guard (low battery)

createVoteDelivery(5, "obstacle\_avoid\_001") // to guard

// Agent 3 doesn't receive vote (failed)

);

**Step 5: Vote Responses (Lauren → John)**

// Agents respond with votes

List<OutgoingMessage> voteResponses = Arrays.asList(

createVoteResponse(2, 1, "GO\_LEFT", "Left path appears clear"),

createVoteResponse(4, 1, "FORMATION\_CHANGE", "Low battery, need efficient path"),

createVoteResponse(5, 1, "GO\_LEFT", "Agree with scout assessment")

);

**Step 6: Consensus Reached (Lauren)**

// Lauren calculates consensus

VoteResult result = new VoteResult();

result.proposalId = "obstacle\_avoid\_001";

result.consensusReached = true;

result.winningOption = "GO\_LEFT";

result.voteBreakdown = Map.of("GO\_LEFT", 2, "GO\_RIGHT", 0, "FORMATION\_CHANGE", 1);

result.consensusLevel = 0.67; // 2 out of 3 votes = 67%

**Step 7: Movement Commands (Lauren → Sanidhaya)**

// Execute decision with coordinated movement

List<MovementCommand> executionCommands = Arrays.asList(

// Leader coordinates the turn

new MovementCommand(1, MovementType.FLOCKING\_BEHAVIOR,

Map.of("targetDirection", new Vector2D(-0.8, 0.6), "speed", 35.0)),

// Scout takes point position

new MovementCommand(2, MovementType.FORMATION\_POSITION,

Map.of("formationType", "COLUMN", "positionIndex", 0)),

// Guards follow in formation, agent 4 conserves energy

new MovementCommand(4, MovementType.FORMATION\_POSITION,

Map.of("formationType", "COLUMN", "positionIndex", 1, "energyConservation", true)),

new MovementCommand(5, MovementType.FORMATION\_POSITION,

Map.of("formationType", "COLUMN", "positionIndex", 2))

);

**Step 8: Task Reallocation (Lauren)**

// Reassign failed agent's tasks

TaskAssignment reallocation = new TaskAssignment();

reallocation.originalAgentId = 3; // failed agent

reallocation.newAgentId = 5; // healthy guard takes over

reallocation.taskType = TaskType.REAR\_GUARD;

reallocation.priority = TaskPriority.HIGH;

reallocation.parameters = Map.of(

"position", "REAR",

"alertLevel", "HIGH",

"compensateForFailure", true

);

**Step 9: UI Updates (Anthony)**

// Anthony receives updates and displays process

DecisionStatus displayStatus = new DecisionStatus();

displayStatus.decisionId = "obstacle\_avoid\_001";

displayStatus.type = DecisionType.VOTING;

displayStatus.state = DecisionState.COMPLETED;

displayStatus.currentData = Map.of(

"decision", "GO\_LEFT",

"executionStatus", "IN\_PROGRESS",

"participatingAgents", Arrays.asList(1, 2, 4, 5),

"failedAgents", Arrays.asList(3),

"estimatedCompletion", System.currentTimeMillis() + 15000

);

// Visual updates show:

// - Agent 3 marked as failed (red X)

// - Communication lines updated (no connections to agent 3)

// - Movement vectors showing left turn

// - Vote results displayed in status panel

// - Task reallocation notification

**Final State:**

* 4 agents successfully navigate left around obstacle
* Agent 3 marked as failed, tasks redistributed
* Formation maintained despite loss of one agent
* Battery level of agent 4 monitored closely
* System continues normal operation

**7. JSON Format Examples**

**7.1 Agent State JSON**

{

"agentId": 5,

"agentName": "Guard\_B",

"position": {

"x": 217.2,

"y": 207.9

},

"velocity": {

"x": -2.1,

"y": -1.3

},

"heading": 2.67,

"maxSpeed": 52.0,

"maxTurnRate": 1.7,

"communicationRange": 110.0,

"status": "ACTIVE",

"batteryLevel": 0.89,

"lastUpdateTime": 1635789432156,

"currentTask": {

"taskId": 17,

"taskType": "FORMATION\_GUARD",

"priority": "NORMAL",

"parameters": {

"position": "REAR\_RIGHT",

"formationType": "WEDGE"

}

},

"teamId": 1

}

**7.2 Communication Message JSON**

{

"messageId": "vote\_proposal\_001",

"senderId": 1,

"receiverId": -1,

"messageType": "VOTE\_PROPOSAL",

"priority": "HIGH",

"timestamp": 1635789432156,

"payload": {

"proposalId": "obstacle\_navigation\_001",

"question": "Large obstacle detected ahead. Navigation strategy?",

"options": ["GO\_LEFT", "GO\_RIGHT", "SPLIT\_FORMATION"],

"deadline": 1635789440156,

"minimumVotes": 3,

"consensusThreshold": 0.6

},

"metadata": {

"urgency": "HIGH",

"decisionCategory": "NAVIGATION",

"affectedAgents": [1, 2, 4, 5]

}

}

**7.3 System Status JSON**

{

"timestamp": 1635789432156,

"systemMetrics": {

"totalAgents": 5,

"activeAgents": 4,

"failedAgents": 1,

"averageSpeed": 28.7,

"systemLoad": 0.48,

"memoryUsage": 2.3,

"updatesPerSecond": 33,

"networkHealth": "GOOD"

},

"activeDecisions": [

{

"decisionId": "obstacle\_avoid\_001",

"type": "VOTING",

"state": "CONSENSUS\_REACHED",

"result": "GO\_LEFT",

"consensusLevel": 0.75

}

],

"recentEvents": [

{

"eventType": "AGENT\_FAILURE",

"agentId": 3,

"timestamp": 1635789427156,

"description": "Agent 3 system failure - tasks redistributed"

},

{

"eventType": "VOTE\_COMPLETED",

"decisionId": "obstacle\_avoid\_001",

"timestamp": 1635789431156,

"description": "Navigation vote completed: GO\_LEFT selected"

}

],

"networkStatus": {

"totalConnections": 6,

"messagesPerSecond": 7.2,

"averageLatency": 145,

"failureRate": 0.03

}

}

**7.4 Configuration JSON**

{

"simulationConfig": {

"timeStep": 0.033,

"maxAgents": 20,

"worldBounds": {

"minX": 0, "maxX": 800,

"minY": 0, "maxY": 600

}

},

"communicationConfig": {

"defaultRange": 100.0,

"latencySimulation": 150,

"failureRate": 0.05,

"enableMultiHop": true

},

"behaviorConfig": {

"flocking": {

"separationRadius": 30.0,

"separationWeight": 1.5,

"alignmentRadius": 50.0,

"alignmentWeight": 1.0,

"cohesionRadius": 80.0,

"cohesionWeight": 1.0

},

"voting": {

"consensusThreshold": 0.6,

"votingTimeout": 8000,

"maxRounds": 3

}

},

"uiConfig": {

"refreshRate": 30,

"showCommunicationLinks": true,

"showDecisionProcess": true,

"agentDisplaySize": 8.0

}

}

This sample data document provides realistic examples of every data structure and message type that will flow between your components. Each team member can use these examples to:

1. **Understand exactly what their inputs look like**
2. **Know the precise format for their outputs**
3. **Test their components with realistic data**
4. **Validate their implementations against concrete examples**

The examples show both normal operation and edge cases (like agent failures), giving everyone a complete picture of what to expect during development and integration.

*Sample Data Examples for Team 6*  
*Distributed Multi-Agent System for Autonomous Drones/Robots*  
*Software Engineering Graduate Project*