UNIT 1 INTRODUCTION TO INTELLIGENT AGENTS

DEFINITION OF AN AGENT

An agent is an autonomous software (or hardware) entity that perceives its environment through sensors and acts upon that environment through actuators to achieve a specific goal.

In simple terms: An agent is like a robot or software program that makes decisions on its own to complete tasks.

CHARACTERISTICS OF AGENTS

Characteristic	Description	Example
Autonomy	Can operate without direct human intervention	A vacuum robot cleans rooms by itself
Social Ability	Can interact with other agents or humans	A chatbot communicating with a user
Reactivity	Responds to changes in the environment	A traffic light system that changes with traffic
Pro-activeness	Takes initiative to achieve goals	A personal assistant setting reminders automatically
Perception	Senses the environment	Sensors in a drone detecting obstacles
Action	Performs operations that affect the environment	Drone moving away from an obstacle
Learning Ability	Learns from experience or past actions (optional in some agents)	Recommender system improving suggestions over time

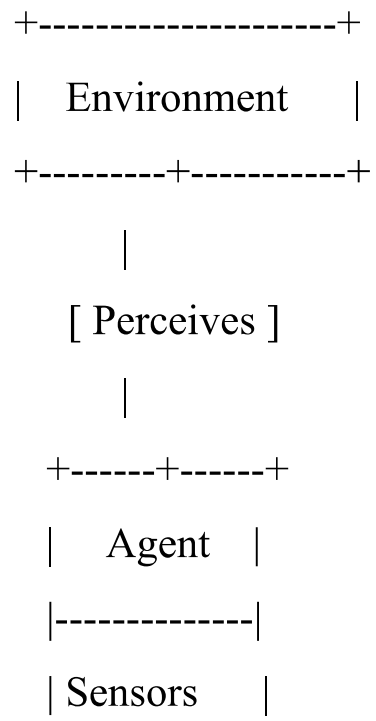
Simple Example

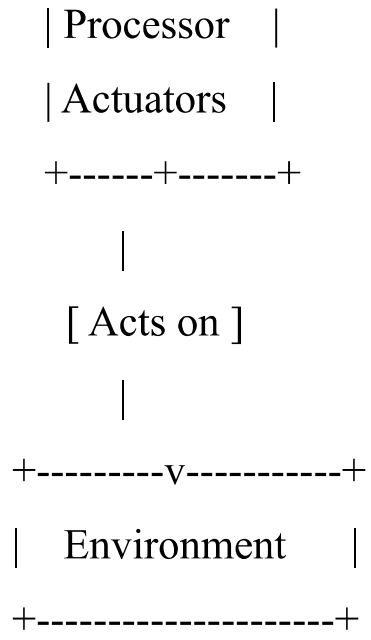
Let's take the example of a Smart Home Assistant Agent (like Alexa or Google Assistant):

Feature	Description
Sensors	Microphones, temperature sensors
Actuators	Speaker, app control for devices
Environment	Your home
Perceives	Sound (voice commands), temperature
Acts	Plays music, turns on AC
Goal	Make user comfortable

Diagram of an Agent

Here is a simple figure to explain how an agent works:





- The agent perceives the environment using sensors.
- It processes the information, decides what to do.
- Then acts using actuators to affect the environment.

Summary

- An agent is a goal-oriented, autonomous entity.
- It has sensors, processing logic, and actuators.
- Key characteristics: Autonomy, Reactivity, Pro-activeness, and Social Ability.
- Real-world examples include robots, smart assistants, trading bots, and game characters.

TYPES OF AGENTS

Types of Agents in Multi-Agent Systems

Agents can be classified based on their complexity, intelligence, and how they make decisions. The most common types are:

1. SIMPLE REFLEX AGENTS

Definition: These agents act only based on the current perception, not on history.

|(or)

A Simple Reflex Agent is an agent that selects actions based only on the current percept (input from the environment), without considering past events.

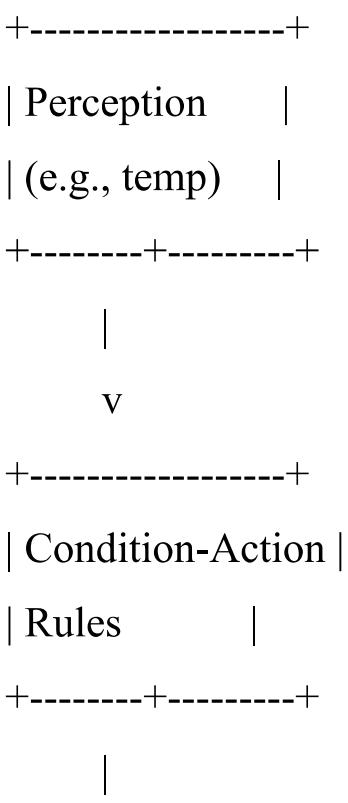
Working:
They use condition-action rules:

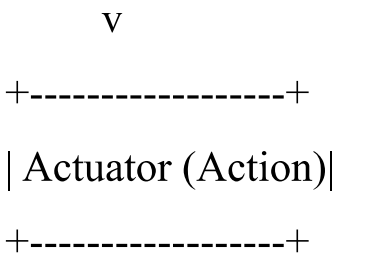
If condition, then action

Example:

- A thermostat:
  - If temperature < 22°C → Turn on heater

Diagram:





2. MODEL-BASED REFLEX AGENTS

Definition: These agents maintain a model (internal state) of the world to handle partially observable environments.

(or)

A Model-Based Reflex Agent is an agent that uses the **current percept** and an **internal model** of the environment to make decisions. It can **handle partially observable environments** by keeping track of **past states**.

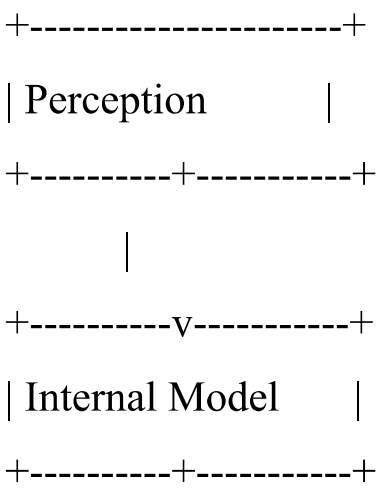
Working:

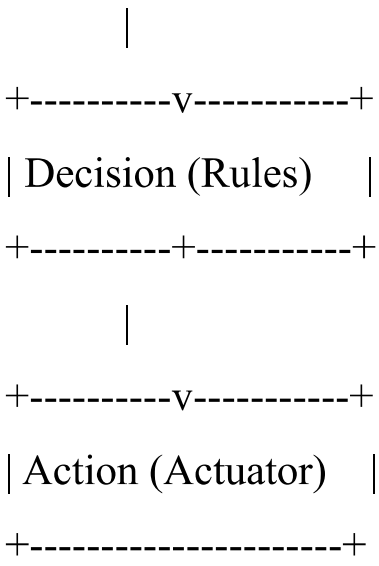
- Uses current perception + internal state to decide action.

Example:

- A vacuum cleaner that keeps track of clean/dirty areas it can't currently see.

Diagram:





3. GOAL-BASED AGENTS

Definition: These agents choose actions based on a specific goal they want to achieve.

(or)

A Goal-Based Agent is an agent that chooses actions by considering **future consequences** and selecting steps that help it **achieve a specific goal**.

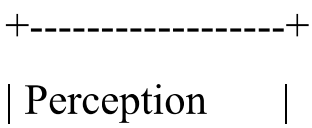
Working:

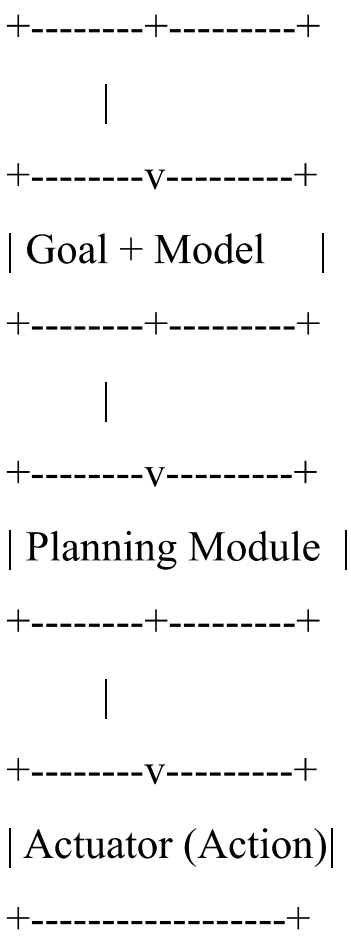
- Uses search and planning to reach the goal.

Example:

- A delivery drone:
  - Goal = Deliver package to house
  - Plans best route using GPS

Diagram:





4. UTILITY-BASED AGENTS

Definition: These agents not only aim to achieve goals, but also maximize happiness/utility.

(or)

A Utility-Based Agent is an agent that selects the **best action** based on a **utility function**, which measures how **satisfactory or beneficial** an outcome is.
It not only aims to achieve goals but also to **maximize performance or happiness**.

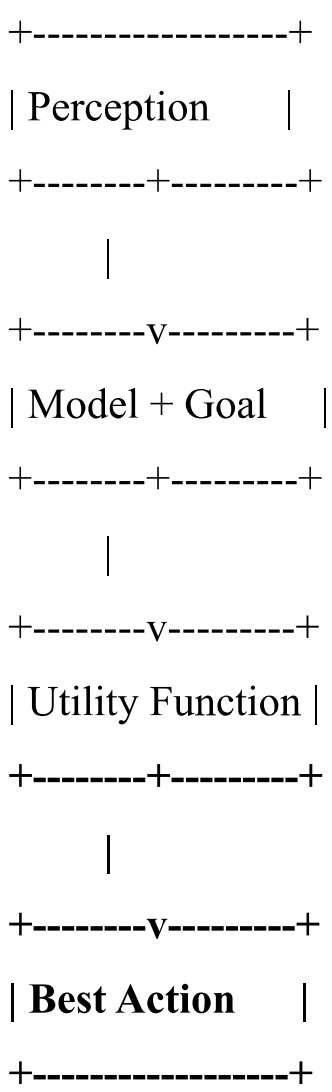
Working:

- Choose the best among multiple options using a utility function.

Example:

- A recommendation system that not only recommends relevant items, but also those with the highest user rating.

Diagram:



5. LEARNING AGENTS

Definition: These agents can learn from experience and improve over time.

(or)

Working Agent (also called a Rational Agent) is an agent that perceives its environment and acts rationally to achieve its designed goals.
It uses a combination of sensing, reasoning (decision-making), and acting to **perform tasks intelligently**.

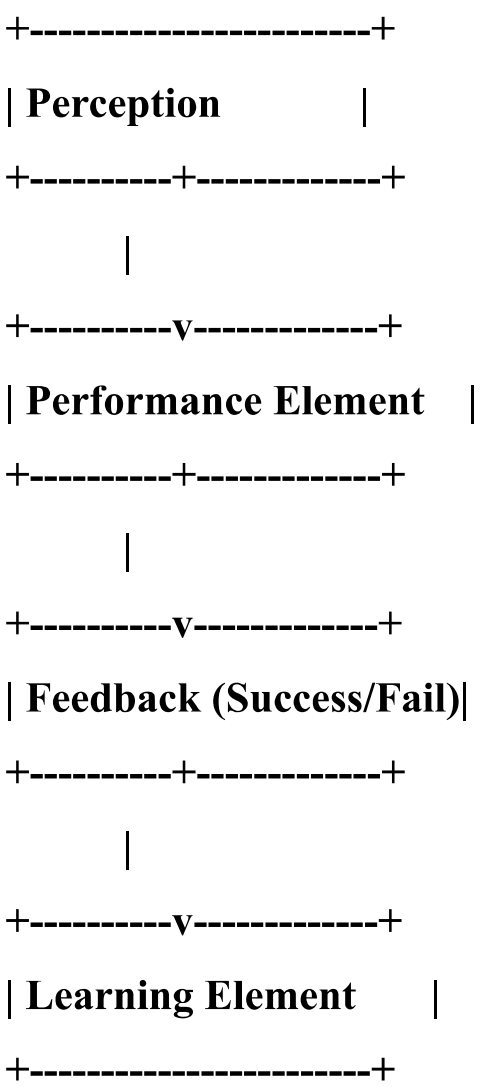
Working:

- Have a learning element and a performance element**.**
- Learn from feedback (reward/punishment).

Example:

- A chess-playing agent that gets better the more it plays.

Diagram:



✅ **Summary Table**

Type	Uses History	Has Goal	Learns	Example
Simple Reflex Agent	❌	❌	❌	Thermostat
Model-Based Reflex Agent	✅	❌	❌	Smart Vacuum
Goal-Based Agent	✅	✅	❌	Delivery Drone
Utility-Based Agent	✅	✅	❌	Product Recommender
Learning Agent	✅	✅	✅	Chess AI, Self-driving car

1. REACTIVE AGENTS

Definition:

Reactive agents immediately respond to current perceptions without internal models or reasoning.

Example:

- Obstacle-avoiding robot: Moves away when it detects an obstacle.

Key Features:

- Fast response
- No planning or memory
- Based on stimulus-response behavior

Diagram:

[Perception]

↓

[Condition-Action Rules]

↓

[Action]

2.DELIBERATIVE AGENTS

Definition:

Deliberative agents build an internal model of the world and use reasoning and planning to decide actions.

Example:

- Self-driving car: Plans route, predicts traffic, and makes decisions.

Key Features:

- Maintains internal state
- Uses logic and goal-directed reasoning
- Slower but intelligent

Diagram:

[Perception]

↓

[World Model]

↓

[Reasoning & Planning]

↓

[Action]

3. HYBRID AGENTS

Definition:

Hybrid agents combine both reactive and deliberative approaches to balance speed and intelligence.

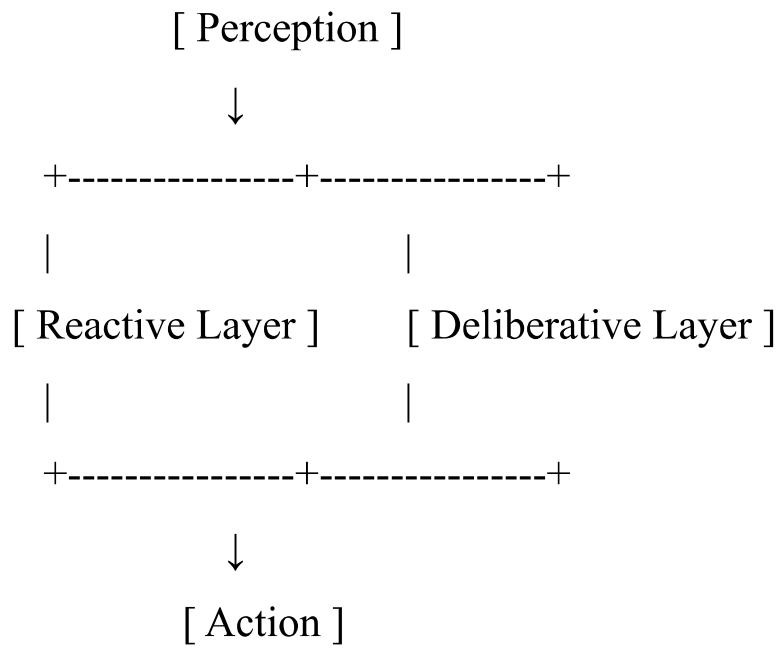
Example:

- Mobile robot: Reacts to obstacles instantly (reactive), but also plans a global path (deliberative).

Key Features:

- Fast local reactions
- Intelligent goal planning
- Most practical agents in real world

Diagram:



4.Learning Agents

Definition: Learning agents improve their performance over time by learning from past experiences or feedback

Example:

- Game-playing AI: Learns better strategies as it plays more.

Key Features:

- Uses feedback or rewards
- Has a learning module
- Adapts to new environments

Diagram:

[Perception]

↓

[Performance Element]

↓

[Action]

↓

[Feedback]

↓

[Learning Element]

↑

[Knowledge]

Summary Table

Agent Type	Memory	Planning	Learning	Example
Reactive Agent	✗	✗	✗	Obstacle-avoiding robot

Deliberative Agent	✔	✔	✖	Self-driving car
Hybrid Agent	✔	✔	✖/✔	Mobile robot with map
Learning Agent	✔	✔	✔	Chess-playing AI, Chatbot AI

1. SUBSUMPTION ARCHITECTURE

Definition:

The **Subsumption Architecture** is a **bottom-up, reactive** agent architecture where behavior is organized in **layers**, and **higher layers can suppress lower layers**.

Key Points:

- No central control or planning
- Each layer is reactive
- Behaviors are prioritized (lower layers = basic, higher = complex)

Example:

- **Mobile Robot**:
  - Layer 1: Avoid obstacles
  - Layer 2: Follow wall
  - Layer 3: Explore environment

Diagram:

+--------------------+ <-- Layer 3: Explore

| Behavior Layer 3 |

+-------------------+ <-- Can override Layer 2

| Behavior Layer 2 |

+-------------------+ <-- Can override Layer 1

| Behavior Layer 1 |

+-------------------+ <-- Basic (e.g., avoid obstacles)

2. BDI ARCHITECTURE (BELIEF-DESIRE-INTENTION)

Definition:

The **BDI architecture** models rational agents based on **three mental attitudes**:

- **Beliefs**: What the agent knows
- **Desires**: Goals to be achieved
- **Intentions**: Plans agent commits to

Key Points:

- Inspired by human reasoning
- Supports planning and decision-making
- Dynamic: Agent can change beliefs and plans

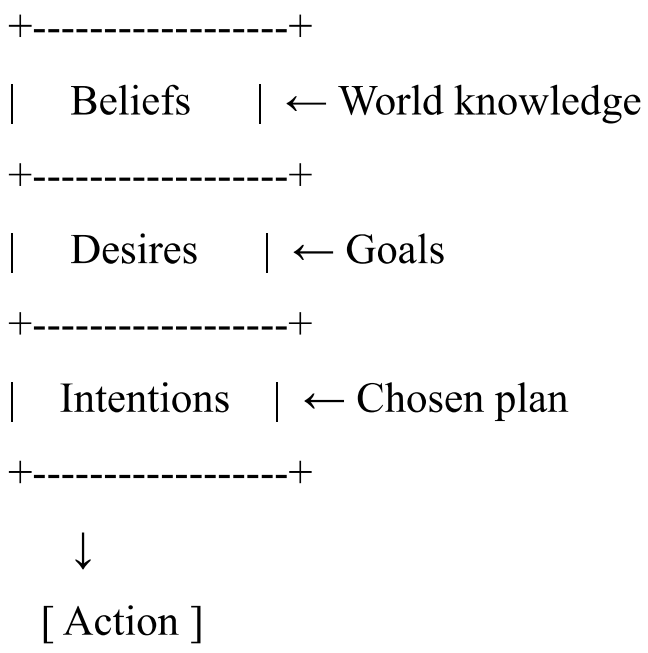
Example:

- **Smart Assistant**:
    - Belief: User has a meeting at 10 AM
    - Desire: User wants to reach on time
    - Intention: Set alarm for 9 AM

Diagram:

[Perception]

↓



3. LAYERED ARCHITECTURE

Definition:

Layered architectures combine both **reactive and deliberative** layers, each responsible for different levels of processing.

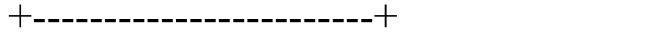
Key Points:

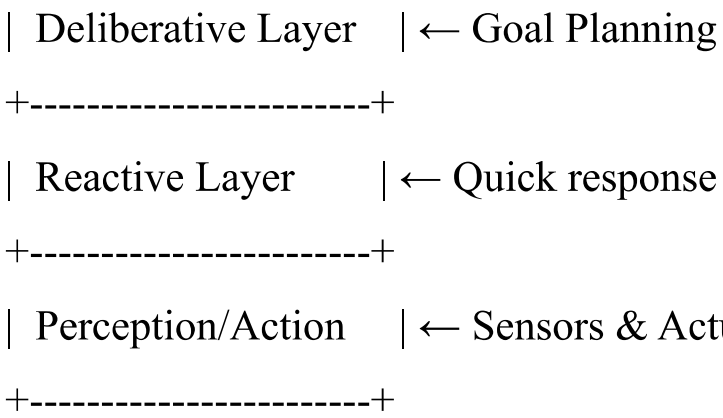
- Organized in layers (physical → reactive → planning)
- Different versions: Horizontal, Vertical, Hybrid
- Enables both **quick responses** and **long-term planning**

Example:

- **Self-driving Car**:
  - Bottom Layer: Obstacle avoidance
  - Middle Layer: Path tracking
  - Top Layer: Route planning

Diagram (Hybrid Layered):





Summary Table

Architecture	Key Concept	Decision Style	Example
Subsumption	Layered behaviors	Reactive	Robot obstacle avoidance
BDI	Beliefs, Desires, Intentions	Deliberative	Smart personal assistant
Layered	Mix of reactive & planning	Hybrid	Autonomous vehicle

1. TYPES OF ENVIRONMENTS

Environments where agents operate can be classified as:

1. DETERMINISTIC VS. STOCHASTIC

Type	Description	Example
Deterministic	Next state is fully determined by current state and action	Chess game
Stochastic	Outcome is uncertain; includes	Weather

randomness forecasting, poker

DETERMINISTIC ENVIRONMENT

An environment is deterministic if the next state of the environment is completely predictable, based on the current state and the agent's action.

Example:

Chess – Every move has a predictable outcome.

STOCHASTIC ENVIRONMENT

An environment is stochastic if there is uncertainty; actions may lead to multiple possible outcomes, due to randomness.

Example:

Weather forecasting – Rain may or may not occur even with same conditions.

Type	**Outcome Predictable?**	**Example**
Deterministic	✓ Yes	Chess, Calculator
Stochastic	✗ No (random involved)	Weather, Dice game

Let me know if you want the same brief format for episodic/sequential next!

Diagram:

[Current State] + [Action] →

↓

[Next State] (Determined) ← Deterministic

[Current State] + [Action] →

↓

[Possible Outcomes] (Random) ← Stochastic

2. EPISODIC VS. SEQUENTIAL

Type	Description	Example
Episodic	Agent's experience is divided into independent episodes	Image recognition, spam filter
Sequential	Current decision affects future decisions	Driving a car, playing chess

EPISODIC ENVIRONMENT

An environment is episodic if the agent's current decision does not affect future decisions.
Each episode is independent.

Example:

- Spam filter – Each email is processed separately.

SEQUENTIAL ENVIRONMENT

An environment is sequential if the current action affects future actions.
Decisions are interdependent.

Example:

- Chess game – Each move affects future moves and strategy.

Are	actions	Example

Type	linked?	
Episodic	No	Spam detection
Sequential	Yes	Chess, Driving

Diagram:

Episodic:

[Percept1] → [Action1]

[Percept2] → [Action2]

(Independent)

Sequential:

[Percept1] → [Action1] → [State changes] → [Percept2] → ...

(Linked decisions)

3. RATIONALITY AND AUTONOMY

Rationality

Definition: An agent is rational if it selects actions that maximize its performance measure, based on percepts and knowledge.

Rational Agent Function:

Agent(percept) → action

Example:

- A thermostat that turns on the heater only when needed, saving power.

Autonomy

Definition: An agent is autonomous if it can operate without human intervention, and learn or adapt from its experiences.

Example:

- A robot vacuum that maps the room and avoids obstacles without instructions.

4. SIMPLE AGENT PROGRAMMING MODELS

Simple Agent Programming Models are basic approaches used to design the logic or behavior of an agent.
These models define how an agent maps percepts (inputs) to actions (outputs) to function in an environment.

A. Agent Function

Maps percepts to actions.

f : Percept → Action

B. Agent Program (Algorithm)

An implementation of the agent function using code or rules.

Example:

```
def simple_agent(percept):
    if percept == "dirty":
        return "clean"
    else:
        return "move"
```

Types of Programming Models

Model Type	Description	Example
Table-driven	Maps every percept sequence to an action	Not practical for large spaces
Simplerule-based	Uses condition–action rules	Vacuum agent
Model-based	Keeps track of environment's state	Smart robot
Goal-based	Chooses actions to achieve goals	Delivery drone
Utility-based	Selects best outcome (max utility)	Movie recommender
Learning-based	Improves from experience	Game AI

UNIT 2:MULTI-AGENT SYSTEM FUNDAMENTALS

DEFINITION AND PROPERTIES OF MULTI-AGENT SYSTEMS (MAS)

Definition:

A Multi-Agent System (MAS) is a system composed of multiple interacting intelligent agents, which can be software-based or physical (like robots), working either cooperatively or competitively.

Key Properties:

- Autonomy: Each agent operates independently.
- Local View: No agent has complete global knowledge.
- Decentralization: No central controller.
- Interaction: Agents communicate, cooperate, or compete.

Example:

- Google Maps uses MAS for traffic updates by gathering data from multiple mobile devices (agents).

AGENT COMMUNICATION LANGUAGES

Agent Communication Languages (ACLs) are formal languages that allow software agents to exchange information, make requests, and coordinate actions in a structured way.

They define how agents talk to each other in Multi-Agent Systems (MAS)

Used by agents to communicate and understand each other's messages.

KQML (KNOWLEDGE QUERY AND MANIPULATION LANGUAGE)

KQML is a high-level Agent Communication Language (ACL) used for communication between intelligent software agents.
It focuses on sharing knowledge, asking questions, and requesting actions using special performatives**.**

- Standard for agent communication
- Contains performatives like: ask, tell, achieve
- Enables knowledge sharing

FIPA ACL (FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS – AGENT COMMUNICATION LANGUAGE)

FIPA ACL is a standardized Agent Communication Language developed by the Foundation for Intelligent Physical Agents (FIPA).
It enables agents to communicate using structured, meaningful messages based on speech act theory (like *inform*, *request*, *propose*).

Messages have well-defined fields:

sender, :receiver, :performative, :content, etc.

- FIPA standard language
- Messages have structure:
- (inform
- :sender agent1
- :receiver agent2
- :content "Weather is sunny")

Example:
Two smart assistants (Alexa and Google Assistant) exchanging temperature data.

INTERACTION PROTOCOLS

Interaction Protocols are predefined rules or patterns that guide how agents communicate and interact with each other to complete tasks, negotiate, or cooperate in Multi-Agent Systems (MAS).

They define how conversations start, proceed, and end between agents.

Define the rules for conversation between agents.

Contract Net Protocol

- One agent (manager) announces a task
- Others (contractors) bid
- Manager assigns based on best bid

Diagram:

Manager → Call for Proposal → Agents

Agents → Bids → Manager

Manager → Accept/Reject → Agents

Auctions

- English Auction: Highest bidder wins
- Dutch Auction: Price drops until someone accepts
- Vickrey Auction: Sealed bids; highest wins, pays 2nd price

Example:

eBay bidding between seller and multiple buyers (agents).

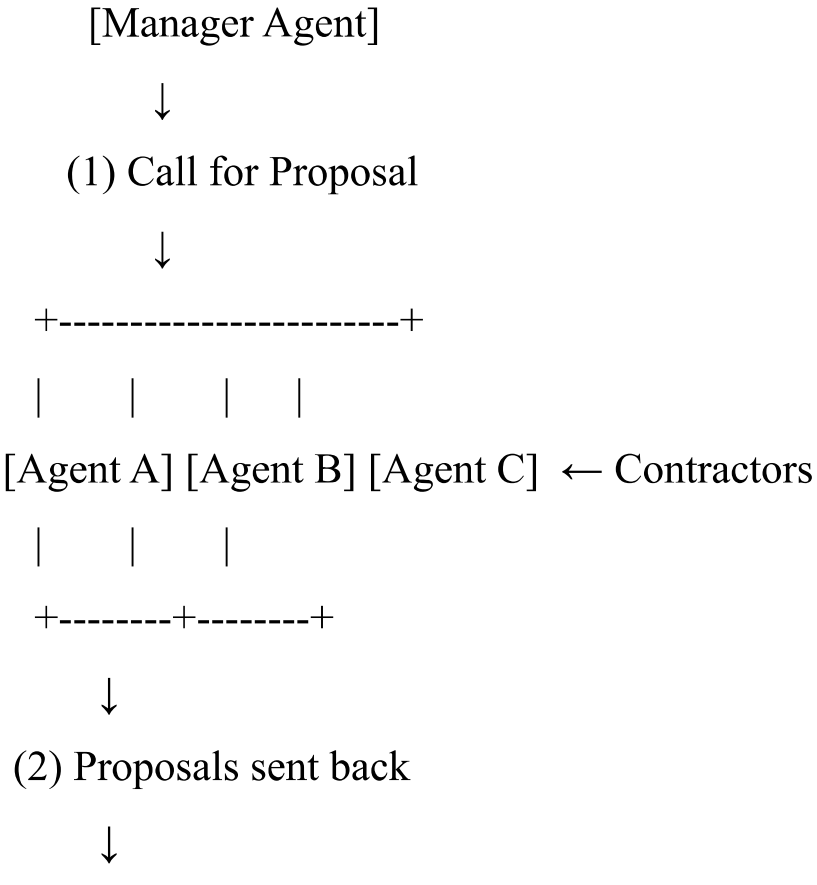
CONTRACT NET PROTOCOL (CNP) – BRIEF EXPLANATION

The **Contract Net Protocol** is a **task-sharing interaction protocol** where one agent (called the **manager**) **announces a task**, and other agents (**contractors**) **bid** to take it up. The manager then **assigns** the task to the best-suited contractor.

Key Steps of CNP:

1. **Manager Agent** announces a task – *Call for Proposal (CFP)*
2. **Contractor Agents** send bids (proposals)
3. **Manager** evaluates and selects the best proposal
4. Task is **assigned** to the selected contractor
5. **Other agents** receive a rejection message

Diagram – Contract Net Protocol



[Manager Agent]

↓

(3) Accept best bid

↓

(4) Assign task to winner

Example – Multi-Robot Cleaning System:

- **Manager Agent**: A central robot detects a dirty room.
- **Contractor Agents**: Other robots (A, B, C) are idle.
- Manager sends **CFP**: "Room 5 needs cleaning."
- Agents bid based on **battery level, distance, or schedule**
- Manager assigns the task to **Robot B** (best suited)

Advantages:

- Dynamic task allocation
- Efficient use of resources
- Scalable in distributed environments

Used in:

- Robotics
- Distributed AI
- Cloud computing
- Workflow scheduling

Let me know if you'd like similar brief notes on **Auction Protocols** next!

COORDINATION TECHNIQUES

Coordination techniques in Multi-Agent Systems are strategies and mechanisms that enable multiple agents to work together effectively to achieve shared or individual goals without conflict.

They help agents organize tasks, avoid interference, and collaborate smoothly used to organize agent behavior when working on tasks.
Blackboard Model

- Agents read/write to a shared blackboard
- Central space for communication

Mediator

- A neutral agent that facilitates interaction among others

Broker

- Matches service requesters with providers

Example:
In a travel system:

- Broker finds hotel
- Mediator resolves overlapping bookings
- Blackboard shares info like available flights

BLACKBOARD MODEL – BRIEF EXPLANATION

Definition:

The Blackboard Model is a coordination technique where agents communicate indirectly by reading and writing to a shared data space called the blackboard.

Key Points:

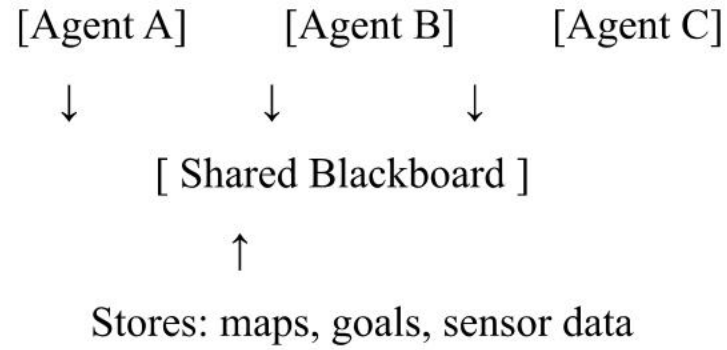
- Agents don't talk directly
- The blackboard holds partial solutions, sensor data, or task updates

- Useful in problem-solving and multi-sensor fusion

Example:

In a robot exploration task, each robot writes discovered map data to the blackboard. Other robots read and update their path.

Diagram:



MEDIATOR

Definition:

A Mediator is a special agent in MAS that manages communication, coordination, or conflict resolution among other agents.

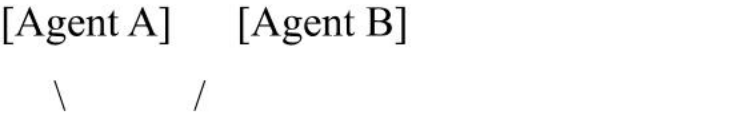
Key Points:

- Acts as a central controller or helper
- Helps resolve conflicts, avoid deadlocks
- Coordinates interactions or schedules tasks

Example:

In a scheduling system, if two agents want the same time slot, the mediator resolves who gets it.

Diagram:



\ /

[Mediator Agent]

↓

Resolves tasks, conflicts, or priorities

DISTRIBUTED PROBLEM SOLVING (DPS) CONCEPTS

In DPS, agents work together to solve parts of a large problem.

Key Concepts:

- Task decomposition
- Solution sharing
- Distributed decision making

Example:
Multi-robot warehouse:

- Robot 1 brings boxes
- Robot 2 sorts them
- Robot 3 loads them

Diagram:

[Problem] → [Subtask 1 → Agent A]

→ [Subtask 2 → Agent B]

→ [Subtask 3 → Agent C]

→ [Combined Solution]

ROLES AND TEAMWORK IN MAS

Roles:

- Define agent's function in a group (e.g., leader, worker, helper)

Teamwork:

- Agents cooperate using shared plans, task allocation, and communication

Example:

In a drone delivery team:

- One drone maps route (planner role)
- One carries package (carrier role)
- One monitors obstacles (guard role)

Diagram:

[Team Goal]

|

+--------+--------+--------+

| Agent A| Agent B| Agent C|

| Mapper| Carrier| Guard |

Summary Table of Unit II Topics

Topic	Brief Description	Example
Definition of MAS	Multiple agents interacting in one system	Traffic systems
Agent Communication	KQML, FIPA ACL for message exchange	Smart devices sharing data
Interaction Protocols	Rules: Contract Net, Auctions	Bidding for tasks
Coordination	Methods: Blackboard,	Agents booking

Techniques	Broker, Mediator	tickets
Distributed Problem Solving	Agents solve parts of a problem collaboratively	Robots in warehouse
Roles and Teamwork	Agents take specific roles and work in teams	Drone delivery system

UNIT 3 COOPERATION,NEGOTIATION,AND LEARNING

COOPERATIVE AND NON-COOPERATIVE AGENTS

Cooperative Agents:

Cooperative agents are agents in a Multi-Agent System (MAS) that work together to achieve a common goal or shared objective.
They communicate, coordinate, and often share knowledge or resources to complete tasks more efficiently.

- Agents work together toward a common goal.
- They share knowledge, coordinate actions, and communicate.

Example: Multi-robot team cleaning different rooms in a house.

Non-Cooperative Agents:

Non-Cooperative agents are agents in a Multi-Agent System (MAS) that act independently with their own goals, often without collaboration.
They may compete with each other and usually do not share information or resources.

- Agents act independently, often with conflicting goals.
- No sharing or collaboration; competition may occur.

Example: Trading bots in stock market competing for profit.

Diagram:

Cooperative Agents → [Shared Goal]

Non-Cooperative Agents → [Competing Goals]

NEGOTIATION TECHNIQUES

Negotiation techniques in Multi-Agent Systems (MAS) are strategies that agents use to reach agreements when they have conflicting goals or interests.
These techniques help agents communicate, bargain, and cooperate to decide who does what, when, and how.

Agents use these strategies to reach agreements:

TECHNIQUE	DESCRIPTION	EXAMPLE
Bidding	Agents bid for tasks/resources	Auction-based delivery tasks
Bargaining	Agents negotiate price/conditions	Two suppliers discussing price
Argumentation	Agents present reasons to influence others	Justifying task priority

Diagram – Bidding:

[Manager] → Call for Bids

[Agent A] → Bid: $50

[Agent B] → Bid: $40 (✓ Accepted)

BIDDING

Definition:

Bidding is a negotiation technique where one agent offers a task or resource, and other agents **submit bids** to win it. The **best bid** is selected based on predefined criteria (e.g., cost, time, efficiency).

Key Points:

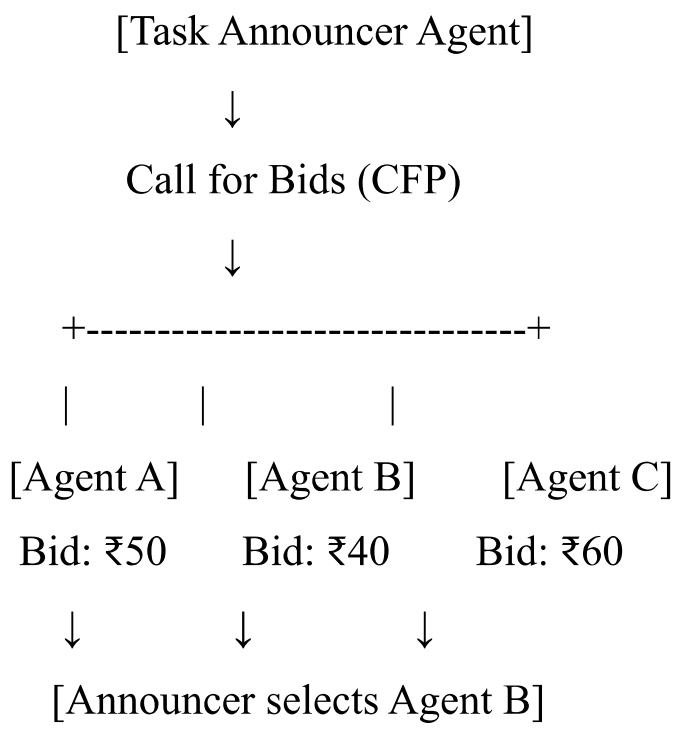
- Common in **auction-based systems**
- Used for **task allocation**, **resource distribution**
- Supports **dynamic decision-making**

Example:

A central controller needs to assign a delivery task.

- Drone A bids: ₹50
- Drone B bids: ₹40
- Controller chooses Drone B.

Diagram – Bidding (Auction Style):



BARGAINING – BRIEF EXPLANATION

Definition:

Bargaining is a negotiation process where agents **exchange proposals and counter-proposals** to reach a **mutual agreement**.

Key Points:

- Agents may adjust offers to find a **win–win outcome**
- Involves **dialogue**, **compromise**, and **multiple rounds**
- Used when both parties have **negotiation power**

Example:

Buyer Agent wants to pay ₹400 for a service.
Seller Agent asks for ₹500.
They settle at ₹450 after bargaining.

Diagram – Bargaining:

[Agent A (Buyer)] — Offer: ₹400

↑ ↓

Counter-Offer: ₹500 ← [Agent B (Seller)]

↑ ↓

Final Offer: ₹450 ←→ Accepted

ARGUMENTATION – BRIEF EXPLANATION

Definition:

Argumentation is a more advanced negotiation technique where agents use **reasoning** and **justifications** to **persuade** other agents during decision-making.

Key Points:

- Involves **logical arguments**, **facts**, or **rules**
- Useful in **conflict resolution**, **preference handling**, or **persuasive dialogs**

Example:

Agent A: "Let me deliver the parcel — I'm 2 km away."
Agent B: "But I have 90% battery; you're at 30%."
Agent C (Mediator): "Assign to Agent B due to power advantage."

Diagram – Argumentation:

[Agent A]: I'm closer!

[Agent B]: I have more battery!

↓

[Decision-Maker Agent]: Evaluates arguments → Chooses best-suited agent

Summary Table

Technique	Description	MAS Use Case
Bidding	Agents compete by placing bids	Auctioning tasks to drones
Bargaining	Agents negotiate via offer and counter	Pricing between buyer/seller bots
Argumentation	Agents persuade using logic/reasons	Selecting best agent for mission

GAME THEORY: BASICS AND APPLICATIONS IN MAS

Game Theory:

A mathematical framework to model strategic decision-making among agents.

In MAS:

- Used when agents have conflicting goals
- Helps design strategies for cooperation or competition

Example:

Two delivery drones deciding routes to avoid collision (Nash Equilibrium)

Diagram – Payoff Matrix (Prisoner's Dilemma):

	Agent B: Cooperate	**Agent B: Defect**
Agent A: Cooperate	(3,3)	(0,5)
Agent A: Defect	(5,0)	(1,1)

REINFORCEMENT LEARNING IN MULTI-AGENT SETTINGS

Definition:

Agents learn from experience by receiving rewards or penalties from the environment.

Multi-Agent RL (MARL):

- Multiple agents learn simultaneously
- Each agent considers others' actions while learning

Example:
Autonomous cars adjusting speed based on traffic patterns.

Diagram – RL Loop:

[Agent] → Action → [Environment]

← Reward, New State ←

CASE STUDIES

MULTI-ROBOT COORDINATION

- Robots collaborate on tasks like cleaning, search & rescue, or delivery.
- Use task allocation, shared planning, and path coordination.

Example:Amazon warehouse robots avoiding collisions while picking items.

RESOURCE ALLOCATION

Resource Allocation in Multi-Agent Systems (MAS) is the process of distributing limited resources (like time, bandwidth, energy, or tasks) among multiple agents in a fair, efficient, or goal-oriented manner.

Key Points:

- Involves decision-making and sometimes negotiation
- Can be centralized (one agent allocates) or distributed (agents coordinate)
- Techniques: auctions, bidding, prioritization, fair sharing
- Agents distribute limited resources (CPU, energy, bandwidth).
- Can use auctions, negotiation, or fair sharing.

Example: Multiple VMs requesting bandwidth from a shared cloud host.

CONFLICT RESOLUTION

Conflict Resolution – Brief Definition

Conflict Resolution in Multi-Agent Systems (MAS) refers to the techniques and processes used to resolve disagreements or clashes between agents when they have conflicting goals, actions, or resource needs.

Key Points:

- Ensures smooth cooperation in shared environments
- Prevents deadlocks, overlapping actions, and goal conflicts
- May involve mediation, priority rules, negotiation, or voting

Example:

Two warehouse robots want to access the same narrow path at the same time.
A mediator agent assigns priority to the one with a more urgent delivery, allowing smooth flow.

Methods Used:

- Mediation: A neutral agent makes a decision
- Negotiation: Agents settle the conflict themselves
- Priority Rules: Predefined rules decide who proceeds
- Agents resolve task overlaps or goal clashes using:
    - Mediators
    - Argumentation
    - Priority rules

Example: Two agents want same time slot; mediator assigns based on urgency.

CONSENSUS BUILDING – BRIEF DEFINITION

Consensus Building in Multi-Agent Systems (MAS) is the process by which multiple agents agree on a common decision, belief, or plan, especially in distributed or decentralized systems.

Key Points:

- Ensures **group agreement** despite diverse preferences
- Crucial in **distributed control**, **sensor networks**, and **multi-robot systems**
- Achieved through **voting**, **majority rule**, or **reputation-based trust**

Examples:

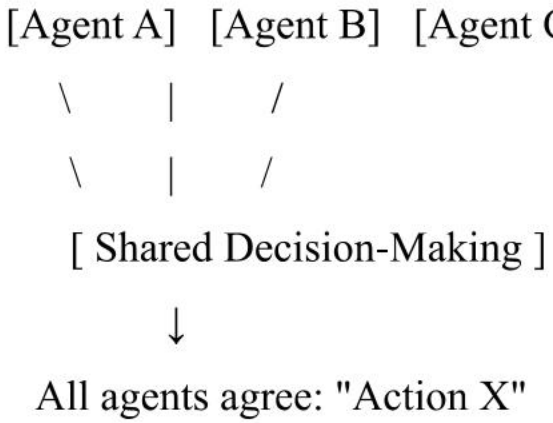
SensorNetwork:

Multiple temperature sensors (agents) agree on the **final average reading** before sending to a server.

Multi-RobotSystem:

Robots agree on **who leads** in a collaborative search task based on consensus.

Diagram – Consensus Building:



Techniques Used:

- Majority voting
- Leader election
- Distributed averaging
- Agreement protocols

Summary Table for UNIT III

Topic	**Description**	**Example**
Cooperative Agents	Work together for shared goals	Cleaning robots in a team
Non-Cooperative Agents	Compete with individual goals	Bidding agents in auctions
Negotiation Techniques	Bidding, bargaining, argumentation	Supplier pricing negotiations

Game Theory in MAS	Decision-making strategies in conflict	Route selection in self-driving cars
Reinforcement Learning	Learn by rewards in shared environment	Learning to avoid traffic jams
Multi-Robot Coordination	Agents collaborate on physical tasks	Robots in warehouse
Resource Allocation	Fair and efficient resource distribution	CPU scheduling in cloud
Conflict Resolution	Solving goal or action clashes	Time slot assignment
Consensus Building	Reaching group agreement	Voting-based system decisions

UNIT 4.AGENT-ORIENTED SOFTWARE ENGINEERING

AGENT-BASED SYSTEM DESIGN METHODOLOGIES

Definition:

These are structured methods used to design, model, and implement multi-agent systems (MAS) efficiently, ensuring agents' behaviors, interactions, and environment are properly defined.

Popular Methodologies:

Methodology	Description	Example Use
Gaia	Focuses on roles, interactions, and agent responsibilities	Designing a smart building system
Tropos	Based on early requirement analysis; uses goals and intentions	Healthcare assistant agents

Diagram – Gaia Methodology:

[Roles] → [Permissions] → [Protocols] → [Agent Types]

AGENT UML: NOTATIONS AND MODELING

Definition:

Agent UML (AUML) is an extension of UML tailored to model agent systems, especially for interactions and protocols between agents.

Agent UML: Notations and Modeling – Brief Definition

Agent UML (AUML) is an extension of Unified Modeling Language (UML) specifically designed to model agent-based systems. It includes special notations to represent agent roles, interactions, behaviors, and protocols.

Key Features:

- Models agent communication and coordination
- Uses sequence diagrams, statecharts, and activity diagrams tailored for agents
- Supports protocol-based and goal-driven agent modeling

Example:

A Buyer Agent and Seller Agent interacting through a Request–Quote–Confirm sequence can be modeled using an AUML sequence diagram.

Let me know if you want an AUML diagram example or comparison with regular UML!

Key AUML Diagrams:

- Agent Interaction Diagrams (like sequence diagrams)
- Activity Diagrams (agent workflow)
- State Diagrams (agent behavior states)

Example:

Modeling a negotiation between Buyer Agent and Seller Agent using AUML Sequence Diagram.

Sample AUML Diagram:

Buyer Agent → Request Quote → Seller Agent

Seller Agent → Send Quote → Buyer Agent

Buyer Agent → Confirm Order → Seller Agent

MAS DESIGN PATTERNS AND BEST PRACTICES

Definition:

MAS (Multi-Agent System) design patterns are reusable solutions to common design problems encountered in agent-based system development.
Best practices are guidelines that help developers create efficient, scalable, and reliable MAS architectures.

Common MAS Design Patterns:

Pattern	Description	Example
Mediator	Central agent manages coordination	Central controller in robot team
Contract Net	Task distributor collects bids from agents	Job allocation in delivery bots
Broker	Matches service seekers and providers	Travel planner matches hotels

Best Practices:

- Define clear agent roles and behaviors
- Ensure modularity and scalability
- Implement error handling and backup agents
- Use standard communication protocols

Let me know if you'd like a visual of one of these design patterns!

Design patterns are reusable solutions for common problems in MAS development, just like software engineering design patterns.

Common MAS Patterns:

- Mediator Pattern: Central agent manages interactions
- Contract Net Pattern: Agents bid for tasks
- Broker Pattern: Matches service seekers and providers

Best Practices:

- Use clear agent roles and responsibilities
- Modular and scalable architecture
- Error handling and fallback agents

ONTOLOGIES AND SEMANTIC WEB INTEGRATION

Definition:

Ontologies in Multi-Agent Systems define a shared vocabulary and structure for knowledge representation, allowing agents to understand and interpret data consistently.
Semantic Web integration enables agents to access, share, and reason over web data using semantic technologies like RDF, OWL, and SPARQL.

Ontology is a shared vocabulary or conceptual structure used by agents to understand, reason, and communicate.

Semantic Web + MAS:

- Enables agents to access, interpret, and process web data meaningfully
- Uses technologies like RDF, OWL, SPARQL

Example:

Agents in e-commerce use ontologies to agree on product categories and prices.

Diagram – Semantic Communication:

[Agent A] ↔ (Uses Ontology) ↔ [Agent B]

↑ ↑

Shared Vocabulary: "Item", "Price", "Availability"

MIDDLEWARE AND FRAMEWORKS FOR MAS

Definition:

Middleware and Frameworks for MAS – Brief Definition

Middleware in Multi-Agent Systems (MAS) is a software layer that provides the infrastructure needed for agents to communicate, coordinate, and manage tasks in a distributed environment.

Frameworks are development platforms or toolkits that help in building, deploying, and running multi-agent systems easily.

Key Features:

- Support for agent creation, messaging, lifecycle management, and mobility
- Abstracts low-level network operations
- Enables scalable and interoperable MAS development

Popular MAS Frameworks:

Framework	Description
JADE	Java-based platform for FIPA-compliant agents
SPADE	Python-based agent framework using XMPP for communication

Example:

Using JADE, you can create a group of chat agents that communicate and perform tasks like scheduling meetings.

Le t me know if you want architecture diagrams of JADE or SPADE!

Middleware provides a platform or infrastructure for building, deploying, and managing agents.

Popular MAS Frameworks:

Framework	Description
JADE	Java-based agent development framework
SPADE	Python-based, ideal for XMPP agent communication

Example:

JADE used to build a multi-agent chat system where agents can send/receive messages.

SCALABILITY, FAULT-TOLERANCE, AND DEPLOYMENT CHALLENGES

Scalability:

System should handle increasing number of agents or tasks efficiently.

Fault Tolerance:

System continues functioning even if some agents fail (use of backup agents, retries, logging).

Deployment Challenges:

- Heterogeneous platforms
- Network failures

- Real-time coordination
- Agent discovery and migration

Diagram – Fault-Tolerant MAS:

[Agent A] → [Main Task]

↓

[Agent B (Backup)] — steps in if A fails

Summary Table – UNIT IV Topics

Topic	**Key Idea**	**Example**
Agent Design Methodologies	Gaia, Tropos structure MAS design	Smart city coordination agents
Agent UML	AUML diagrams model agent interactions	Buyer-Seller negotiation model
MAS Design Patterns	Reusable solutions like Mediator, Broker	Broker finding hotel for travel agent
Ontologies & Semantic Web	Shared vocabulary enables understanding	Shopping agents using product ontology
Middleware & Frameworks	Platforms like JADE, SPADE for MAS	Chatbot system using JADE
Scalability & Fault Tolerance	Robust MAS under growth or failures	Backup agents, task redistribution

UNIT 5.ADVANCED TOPICS AND APPLICATIONS

EMERGENCE AND SELF-ORGANIZATION IN MAS

Definition:

Emergence refers to complex behaviors arising from simple local agent interactions.
Self-organization is when agents organize themselves without central control.

Example:

Ant colony finding shortest path to food through pheromone trails.

Diagram:

[Simple Rules in Agents] → [Local Interactions] → [Global Order Emerges]

SWARM INTELLIGENCE AND DISTRIBUTED OPTIMIZATION

Definition:

Swarm intelligence is inspired by collective behavior of social insects. Distributed optimization uses multiple agents to solve optimization problems cooperatively.

Example:

- Swarm Intelligence: Boids (flocking birds) simulation

- Distributed Optimization: Load balancing by cloud agents

Diagram (Swarm Movement):

[Agent A] ←→ [Agent B] ←→ [Agent C]

↑ Rules: Separation, Alignment, Cohesion

TRUST, PRIVACY, AND ETHICS IN MAS

Definition:

- Trust: Agents rely on trustworthy agents based on reputation or past behavior.
- Privacy: Ensuring that sensitive agent data is not leaked.
- Ethics: Ensuring agents behave in morally acceptable ways (e.g., fairness, non-discrimination).

Example:

E-commerce agents trust only verified sellers; health agents preserve user data privacy.

REAL-TIME AND EMBEDDED MAS APPLICATIONS

Definition:

Real-time MAS operate under strict time constraints.
Embedded MAS are integrated into physical systems like IoT or robotics.

- Real-Time MAS: Multi-Agent Systems that must operate within strict time constraints—agents must sense, decide, and act within a specific time window.
- Embedded MAS: Agent systems that are integrated into physical hardware or devices, such as robots, sensors, or IoT systems.

Key Characteristics:

Aspect	Real-Time MAS	Embedded MAS
Time Sensitivity	High (deadlines must be met)	Moderate to High (depends on task)
Integration	Software-focused (with fast response)	Hardware + software (often embedded)
Examples	Autonomous cars, air traffic control	Smart homes, robotic systems

Examples:

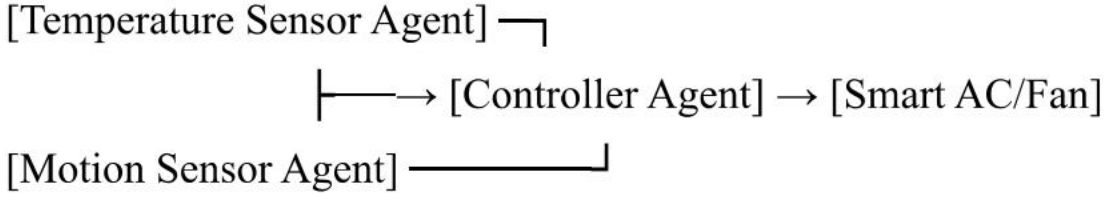
Real-Time MAS:

- Autonomous drone swarm delivering packages in a city where each drone must update its route every few seconds.
- Traffic light control system that responds to vehicle flow in real time.

Embedded MAS:

- Smart thermostat system where multiple sensor agents control heating/cooling devices.
- Wearable health monitors using agent-based decision logic to alert medical conditions.

Diagram – Embedded MAS Example: Smart Home



All agents operate in real-time to maintain comfort and energy efficiency.

Benefits:

- Faster and autonomous responses

- Local decision-making reduces need for central control
- Adaptable to dynamic environments

Examples:

- Real-time: Traffic light coordination
- Embedded: Smart thermostats using agents to adjust temperature

MAS IN SMART GRID, IOT, AND TRAFFIC SYSTEMS

Smart Grid:

Agents control and balance electric power loads, ensuring efficiency.

IoT:

Agents coordinate among smart devices (like lights, sensors, fans).

Traffic Systems:

Agents manage signals, optimize vehicle flow, and detect congestion.

Diagram (Traffic MAS):

[Sensor Agent] → [Traffic Agent] → [Signal Controller Agent]

CASE STUDIES

Amazon Robotics:

Warehouse robots use MAS for:

- Item picking
- Route optimization
- Collision avoidance

Diagram (Amazon Robot MAS):

[Central Agent] → Coordinates → [Mobile Robots] ← Sensor Inputs

Autonomous Trading Agents:

Agents perform stock trading, forecasting, and risk analysis based on real-time market data.

Summary Table

Topic	Description	Example
Emergence & Self-Organization	Global behavior from local actions	Ant colony, flocking birds
Swarm Intelligence	Collective intelligence of simple agents	Boids, bee colonies
Trust, Privacy, Ethics	Security and moral behavior in agents	Trusted e-commerce bots
Real-Time & Embedded MAS	Timely agent decisions in real-world systems	Smart thermostats, real-time alerts
MAS in Smart Grid, IoT, Traffic	MAS for urban & industrial automation	Load balancing, smart homes, traffic
Case Studies	Real-world MAS usage examples	Amazon robots, Trading bots