Artificial Intelligence, Agents, and Environments

Md Rifat Ahmmad Rashid, Associate Professor, EWU

1 Artificial Intelligence, Agents, and Environments

1.1 What Is Artificial Intelligence (AI)?

Perspective	**Guiding Question**	**Typical Research / Engineering Goal**
Thinking Humanly	*How do people think and learn?*	Cognitive modelling; reproduce human reasoning steps.
Acting Humanly	*Can a machine pass the Turing Test?*	Natural-language dialogue, perception, and adaptive behaviour.
Thinking Rationally	*What* ought *intelligent reasoning look like?*	Derive correct conclusions through formal logic, probability, optimisation.
Acting Rationally	*How can we build agents that do the right thing?*	Maximise expected performance given goals and knowledge.

Table 1: Four classical viewpoints on Artificial Intelligence.

Working Definition (Rational View)
Artificial Intelligence (See Fig 1) is the study and design of computational agents that perceive their environment and take actions that maximise their expected utility or goal achievement over time.

2 Agents: The Core Abstraction

2.1 Formal Definition

An **agent** is an entity that *senses* and *acts*. Formally, it is a function

$$\pi : P^* \longrightarrow A,$$

mapping a finite history of percepts P^* (observations) to an action A. In words:
Percept $\rightarrow$ *Decide* $\rightarrow$ *Act* $\rightarrow$ *Repeat*

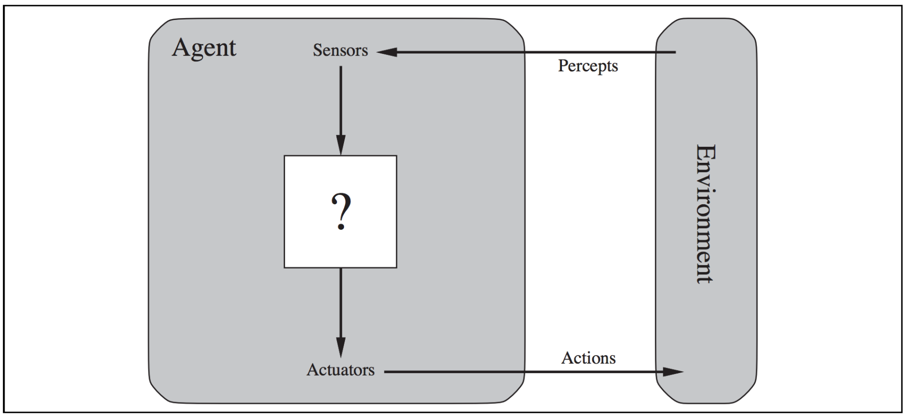

Figure 1: Artificial Intelligence System

2.2 The PEAS Framework

Before designing an agent, specify its **P**erformance measure, **E**nvironment, **A**ctuators, and **S**ensors.

Component	**Example: Autonomous Taxi**
Performance	Average trip time, safety, passenger comfort, fuel cost, legality
Environment	City roads, traffic, weather, pedestrians
Actuators	Steering, accelerator, brake, horn, dashboard displays
Sensors	Cameras, GPS, LIDAR, speedometer, microphone

Table 2: PEAS specification for an autonomous–taxi agent.

2.3 Types of Agents

1. **Simple Reflex Agents** – Rule based on the current percept only.
   e.g. if traffic_light = *red* then stop.

2. **Model-Based Reflex Agents** – Maintain an internal state s_t encoding aspects of the world not observable at t.

3. **Goal-Based Agents** – Choose actions that achieve a goal state G.

4. **Utility-Based Agents** – Maximise a utility function $U(s)$, supporting trade-offs and reasoning under uncertainty.

5. **Learning Agents** – Improve performance over time through:
   - Learning Element
   - Performance Element
   - Critic
   - Problem Generator

Details

1. **Simple Reflex Agents(See Fig. 2)** – Rule–based decision making that depends only on the current percept:
   - *Mechanism:* A fixed set of condition–action pairs ("if–then" rules).
   - *Example:*
   $$\text{if traffic\_light} = \text{red} \implies \text{stop}$$
   - *Advantages:*
     - Very efficient for fully observable, static environments.
     - Easy to implement and verify.
   - *Limitations:*
     - Cannot handle environments that are partially observable or dynamic.
     - No memory of past percepts — cannot learn or plan.

2. **Model-Based Reflex Agents(See Fig. 3)** – Enhance simple reflex agents by maintaining an internal model of the world:
   - *Internal State s_t:* Updated using the last state, the last action, and the current percept:
   $$s_t = \text{update}(s_{t-1}, a_{t-1}, o_t).$$
   - *State Representation:* Encodes unobserved aspects of the environment (e.g., locations of hidden obstacles).
   - *Decision Rules:* Condition–action rules that refer to the internal state instead of raw percepts.

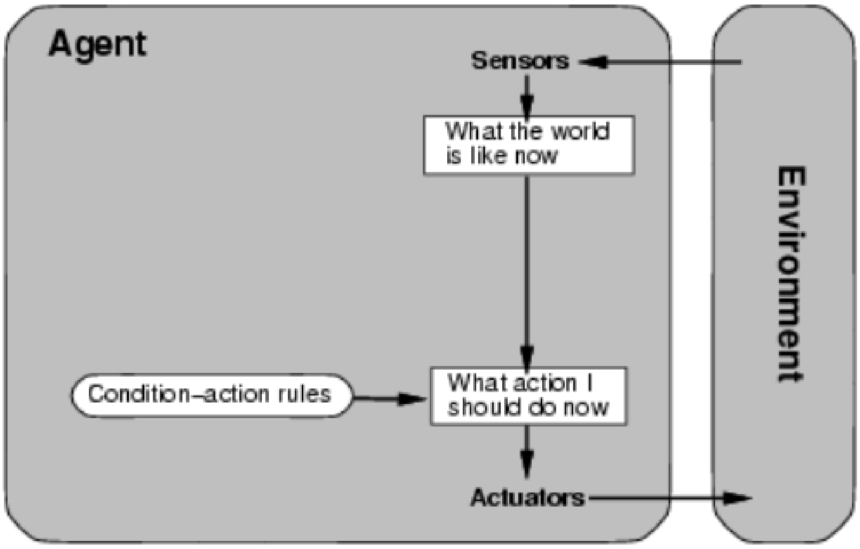

Figure 2: Simple Reflex Agents

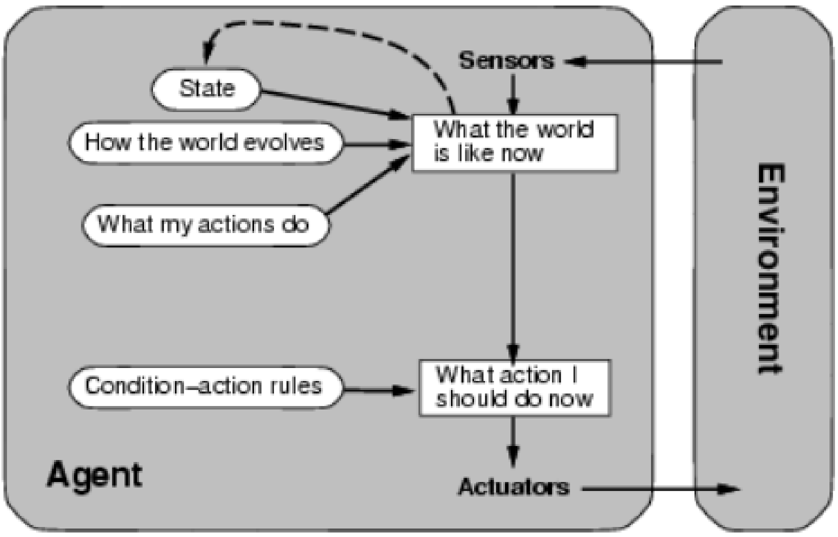

Figure 3: Model-Based Reflex Agents

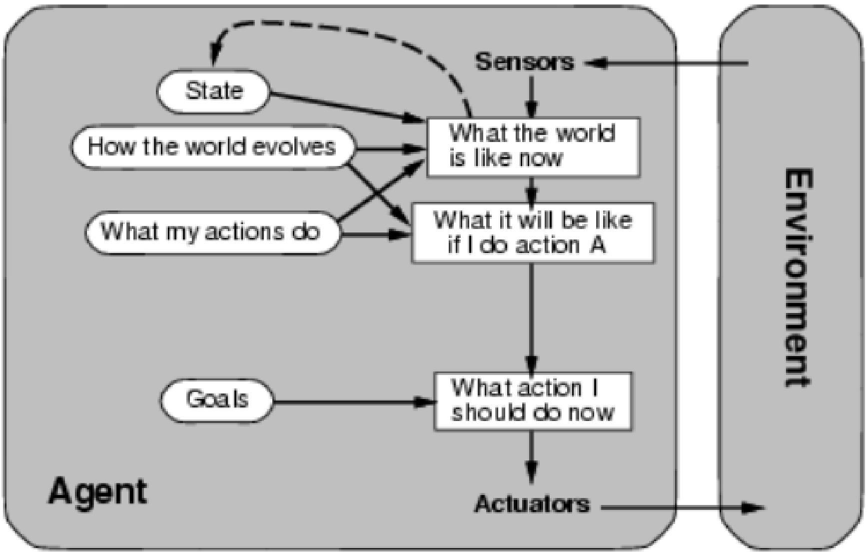

Figure 4: Goal Based Agent

- *Example:* Vacuum–cleaner agent that remembers which squares are already clean.
- *Benefits:*
  - Handles partially observable environments.
  - Can avoid repeating ineffective actions.

3. **Goal-Based Agents (See Fig. 4)** – Select actions to achieve a specified goal state G:
   - *Goal Formulation:* A Boolean predicate or set of states that the agent strives to reach.
   - *Planning:* Uses search or optimization to find a sequence of actions $\langle a_0, \ldots, a_n \rangle$ leading to G.
   - *Example:*
   $$G : \{\text{all rooms clean and no dirt remains}\}$$
   The agent plans a route that visits each dirty square exactly once.
   - *Advantages:*
     - Flexible to changing objectives.
     - Can compare alternative plans based on goal satisfaction.

4. **Utility-Based Agents (See Fig. 5)** – Go beyond Boolean goals by maximising a utility function $U(s)$:
   - *Utility Function:* Assigns a real-valued score to each state, reflecting agent preferences and trade-offs.
   - *Expected Utility:* Chooses actions a that maximise
   $$\mathbb{E}\big[U(s')\big] = \sum_{s'} P(s' \mid s, a)\, U(s').$$
   - *Example:* Autonomous taxi balances speed, safety, and passenger comfort:
   $$U(s) = w_1 \times (\text{travel\_time}) + w_2 \times (\text{safety\_score}) + w_3 \times (\text{comfort})$$
   - *Strengths:*
     - Handles uncertainty and trade-offs systematically.
     - Avoids the brittleness of hard-coded goals.

5. **Learning Agents (See Fig. 6** – Improve their performance based on experience and feedback:

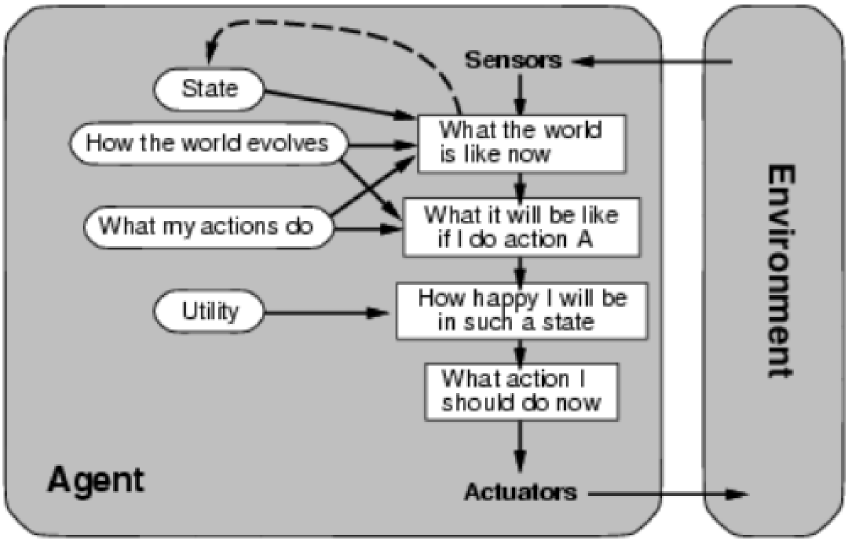

Figure 5: Utility Based Agent

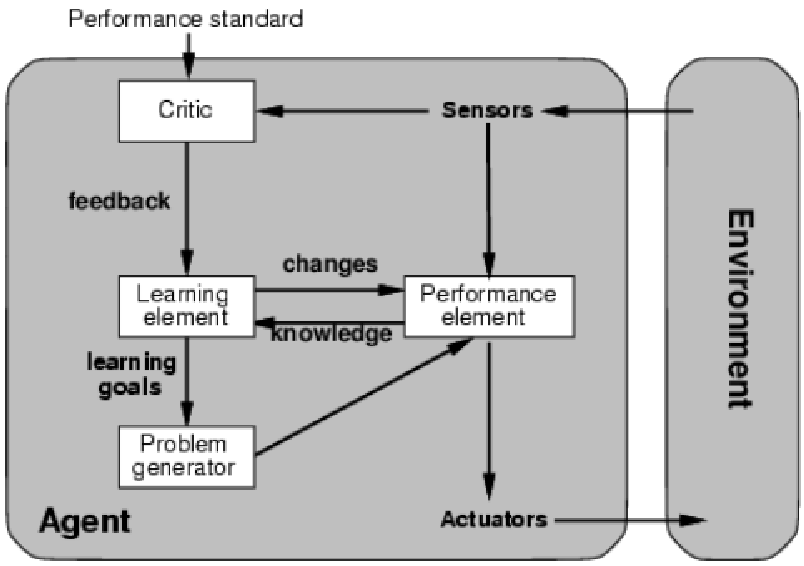

Figure 6: Learning Agent

- *Performance Element:* Chooses actions given the current knowledge (e.g., a policy network or rule set).
- *Learning Element:* Updates the performance element using data gathered from interaction.
- *Critic:* Evaluates agent behaviour by comparing actual performance to desired performance (e.g., via a reward signal).
- *Problem Generator:* Suggests exploratory actions to gather informative experiences (e.g., random moves, curiosity-driven exploration).
- *Example Workflow:*
  (a) Agent takes action a_t in state s_t.
  (b) Environment returns percept o_{t+1} and reward r_{t+1}.
  (c) Critic computes temporal-difference error $\delta = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$.
  (d) Learning element updates value estimates or policy parameters.
  (e) Problem generator occasionally overrides greedy actions to explore.
- *Advantages:*
  - Adaptation to non-stationary environments.
  - Capability to improve beyond initial programming.

3 Environments: The External World

3.1 Environment Properties

Dimension	**Opposing Values**	**Impact on Design**
Observability	Fully vs. Partially Observable	Need for belief state / hidden–state inference.
Determinism	Deterministic vs. Stochastic	Must reason with probabilities, expectimax search.
Episodicness	Episodic vs. Sequential	Can act in isolation or must plan long-term.
Dynamics	Static vs. Dynamic	Real-time processing, continual replanning.
Discreteness	Discrete vs. Continuous state/action/time	Choice of representation and control theory.
Agents	Single-agent vs. Multi-agent	Game-theoretic reasoning, cooperation/competition.
Knowledge	Known vs. Unknown	Learning and exploration vs. model-based planning.

Table 3: Key environment dimensions and their influence on agent design.

3.2 Environment Model

Let S denote the state space and A the action set. The probabilistic transition model is

$$P(s_{t+1} \mid s_t, a_t) = T(s_t, a_t, s_{t+1}),$$

the observation model is

$$P(o_t \mid s_t) = O(s_t, o_t),$$

and the reward (or utility) signal is

$$R(s_t, a_t, s_{t+1}).$$

The tuple $\langle S, A, T, O, R \rangle$ defines a *partially observable Markov decision process* (POMDP); solving it yields the optimal policy π^*.

4 Interaction Loop (Sense–Think–Act)

```
loop:
    o_t  <- sensors()
    s_t  <- update_state(s_{t-1}, a_{t-1}, o_t)
    a_t  <- (s_t)          # planning / learning / reasoning
    actuators(a_t)
```

Key Algorithms

Setting	Canonical Algorithms
Search (fully observable, deterministic)	Breadth-First Search, `A*`, Uniform-Cost Search, Iterative-Deepening `A*`, bidirectional search.
Planning with uncertainty	Markov-decision-process value iteration, policy iteration.
Partially observable domains	Belief-state filtering (Bayes, Kalman), POMDP solvers.
Learning in unknown environment	Reinforcement learning (Q-learning, SARSA, Deep RL).

Table 4: Representative algorithms for different environment settings.

5 Designing Intelligent Agents

1. **Specify PEAS** – Clearly articulate the performance measure, environment, actuators, and sensors.
2. **Analyse Environment** – Identify relevant properties from Section 3.
3. **Select Architecture** – Table-driven, rule-based, planning, learning, or hybrid.
4. **Choose Algorithms and Representations** – Logic, search trees, probabilistic graphical models, neural networks, etc.
5. **Implement & Train** – Prototype in simulation, then deploy in the real environment.
6. **Evaluate** – Measure performance under diverse scenarios against the specified metrics.
7. **Iterate** – Refine utilities, representations, and learning strategies based on empirical results.

6 Mini Case Study: Vacuum-Cleaner Agent

Property	Value
Performance	$+1$ for each clean square, -1 per time-step
Environment	Two-square world; dirt appears stochastically
Actuators	Left, Right, Suck
Sensors	Current location and dirt sensor

Table 5: PEAS specification for the vacuum-cleaner agent.

- **Simple Reflex**: *If dirty,* `Suck`*; else move Right if in left square, otherwise move Left.*
- **Model-Based**: Maintain memory of which squares have been cleaned to avoid redundant moves.
- **Utility-Based**: Plan a sequence that minimises expected cost given the probability of dirt re-appearing.

7 Mathematical Foundations

7.1 Utility Theory

For rational preferences (completeness, transitivity, continuity, independence) there exists a utility function U such that

$$\mathbb{E}[U] \;=\; \sum_{s} P(s)\,U(s).$$

Rational agents act to maximise $\mathbb{E}[U]$.

7.2 Decision-Theoretic Control

$$\pi^* \;=\; \arg\max_{\pi} \; \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \, R\big(s_t, \pi(s_t)\big)\Big],$$

where the discount factor $\gamma \in (0, 1]$ accommodates infinite-horizon problems.

8 Current Trends & Challenges

Trend	**Relevance to Agent Design**
Deep Reinforcement Learning	Scales utility-maximising agents to high-dimensional percepts (e.g. vision, speech).
Large Language Models as Agents	Enable zero-shot planning and tool use via chain-of-thought prompting; still brittle and difficult to align.
Multi-Agent Systems	Swarm robotics, negotiation, distributed optimisation.
Safety & Alignment	Mitigate negative side-effects, specification gaming, and reward hacking.
Embodied AI & Sim-to-Real	Transfer learned policies from simulation to physical robots.

Table 6: Emerging research directions and their impact on intelligent-agent design.

9 Summary

- **AI** aims to build systems that act rationally within their environments.
- **Agents** are the unifying abstraction: they perceive, decide, and act.
- **Environment analysis** (observability, determinism, etc.) dictates the algorithmic toolbox.
- The **PEAS** framework is the design blueprint; utility theory supplies the mathematical backbone.
- Advances in deep learning, large-scale simulation, and reinforcement learning are expanding both the capabilities *and* the responsibilities of AI-agent builders.