

Automatic Reasoning and Planning

Course Summary - Master's Degree

Carlos Alberto Botina Carpio
Universidad Internacional de la Rioja
carlos.botina621@comunidadunir.net

November 13, 2025

Abstract

This document contains a summary of the Automatic Reasoning and Planning course syllabus for the Master's Degree. It includes a summary of the main topics covered during the sessions, as well as additional explanations and extensions of the concepts and techniques referenced in class. The purpose of this document is to serve as study material and reference for the course contents.

Notice that this document is fully customized to the author's needs, placing greater emphasis on topics that the author struggles with or has not yet mastered, while covering more briefly those topics that are already well understood by the author.

Contents

1	Introduction to Decision Making	3
1.1	Decision Classification	3
1.2	Problem Classification	3
2	Problem Solving Stages	4
2.1	First Stage: Understand the Problem's Complexity	4
2.2	Second Stage: Create a Strategy	4
2.3	Third Stage: Solve the Problem	5
3	Intelligent Agents	5
3.1	Phases of Agent Decision Making	5
4	Intelligent Agents Architectures	6
4.1	Deliberative Architecture	6
4.1.1	Examples of Deliberative Architectures	8
4.2	Reactive Architecture	8
4.2.1	Examples of Reactive Architectures	10
4.3	Hybrid Architecture	10
4.3.1	Examples of Hybrid Architectures	11
4.4	Cognitive Architecture	12
4.4.1	Examples of Cognitive Architectures	12
4.5	Other Architectures	13
5	Summary	14

1 Introduction to Decision Making

Decision making is a complex process that requires evaluating and understanding environmental conditions. In many cases, decisions become so intricate that we need to exhaustively evaluate every possible scenario. This is where mathematics becomes useful in helping us make informed decisions.

When making decisions, the following elements are involved:

- **Future effect:** What is the duration of the decision's impact? (short-term, long-term)
- **Reversibility:** Is it easy to reverse the decision?
- **Impact:** How many areas or domains are affected?
- **Quality:** Ethics, legality, behavioral principles, workplace relationships, etc.
- **Periodicity:** How frequently must this decision be made?

1.1 Decision Classification

Decisions can be classified into two main categories: **high-level** and **low-level** decisions. High-level decisions are strategic, have long-term consequences, are difficult to reverse, and affect multiple areas of an organization or system. They are typically exceptional and require careful consideration of various quality factors. In contrast, low-level decisions are operational, have minimal future impact, are easily reversible, and affect fewer areas. They are made frequently and have limited impact on important quality factors.

Table 1 summarizes the characteristics that distinguish high-level from low-level decisions based on the elements involved in decision making.

Table 1: Classification of decisions: High-level vs. Low-level

	High Level	Low Level
Future Effect	Affect the future	Don't affect the future
Reversibility	Difficult reversibility	Reversible
Impact	Broad impact	Little impact
Quality Factors	Affect many important	Affect few important
Periodicity	Exceptional	Frequent

Decisions can also be classified as **programmed** or **non-programmed**. Programmed decisions have a well-defined step-by-step sequence that is known and can be followed. For example, in case of an emergency, one calls the emergency number. Non-programmed decisions are unique and specific to the situation, with no defined rules or steps to follow.

1.2 Problem Classification

Problems can be classified as **structured** or **non-structured**:

- **Structured problems:** The problem contains all the information needed to solve it. All necessary data, constraints, and conditions are available from the start.
 - *Example:* Solving a system of linear equations where all coefficients and constants are given.
- **Non-structured problems:** The problem does not contain all the information needed to solve it. To solve it, we need to search for additional information.
 - *Example:* Diagnosing a medical condition where symptoms are present but additional tests, patient history, or expert consultation are required to reach a diagnosis.

2 Problem Solving Stages

2.1 First Stage: Understand the Problem's Complexity

There are many techniques to understand a problem's complexity, which can be organized into two main categories:

- **Identify the problem:** Self-questioning about the problem (origin, magnitude, focus, or history), SWOT analysis (Strengths, Weaknesses, Opportunities, Threats), discussion meetings (Scrum), etc.
- **Explain the problem:** Go beyond the surface and investigate the underlying causes of the problem. This can be done using various techniques such as ERIM problem classification, the 20 causes technique, the Ishikawa diagram (fishbone diagram), and other root cause analysis methods.

We must, therefore, create problem definitions that present characteristics that allow us to work with them efficiently. Some characteristics or assumptions that can be made in simple environments (well-defined problems) are:

- **Discrete:** The world can be conceived in states. In each state there is a finite set of perceptions and actions.
- **Accessible:** The agent can access the relevant characteristics of the environment. It can determine the **current state** of the world and the **state it would like to reach**.
- **Static and deterministic:** There is no temporal pressure nor uncertainty. The world changes only when the agent acts. The result of each action is totally defined and predictable.

2.2 Second Stage: Create a Strategy

The steps for creating a strategy are:

1. **Define strategies:** Generate potential strategies using techniques such as brainstorming, 4x4x4, etc.

2. **Choose a strategy:** Select a strategy by evaluating:

- Benefits
- Probability of success
- Dependencies
- Resources needed (time, cost)

3. **Design the strategy:** Create a roadmap defining which actions will be performed. This involves planning the sequence and details of the actions to be executed.

2.3 Third Stage: Solve the Problem

This stage is achieved by implementing the strategy, evaluating the outcome, and if necessary, refining the strategy.

3 Intelligent Agents

An **intelligent agent** is an autonomous entity that perceives its environment through sensors and acts upon that environment through actuators to achieve its goals or objectives. Intelligent agents are characterized by their ability to operate independently, make decisions based on their perceptions, and take actions that affect their environment in pursuit of their goals.

An agent is considered intelligent based on its **autonomy** and **rationality**:

- **Autonomy:** The agent operates independently without direct human intervention or control. It has control over its own actions and internal state.
- **Rationality:** The agent acts in a way that maximizes its performance measure, given the available information and its knowledge. A rational agent selects actions that are expected to achieve its goals most effectively.

3.1 Phases of Agent Decision Making

An agent must go through three fundamental phases: **feel**, **think**, and **act**.

- **Feel:** Using perception of the environment through their sensors, the agent extracts and processes information. This phase involves gathering raw data from the environment and converting it into a usable format.
- **Think:** The agent reasons and decides through a deliberative process, using the information obtained from the environment and its internal memory. This phase involves analyzing the current situation, considering possible actions, and selecting the best course of action to achieve its goals.
- **Act:** Through actuators, the agent produces changes in the environment that will help it achieve its goal. For this, the agent must convert its decisions into information that the actuators can understand and execute.

Example: Consider a robotic vacuum cleaner agent. In the **feel** phase, it uses sensors (cameras, bump sensors, dirt detectors) to perceive the room layout, detect obstacles, and identify dirty areas. In the **think** phase, it processes this information along with its internal map and battery level, deciding which areas to clean next and planning an efficient path. In the **act** phase, it converts these decisions into motor commands that control its wheels and vacuum mechanism, moving through the room and cleaning the identified areas.

4 Intelligent Agents Architectures

Agent architectures define the internal structure and organization of an intelligent agent, determining how it processes information, makes decisions, and acts. There are several main types of agent architectures: **deliberative**, **reactive**, **hybrid**, and **cognitive**.

4.1 Deliberative Architecture

A **deliberative architecture** (also known as a *symbolic* or *thinking* architecture) is characterized by the agent's ability to maintain an internal model of the world and engage in explicit reasoning and planning before taking action. The agent uses symbolic representations of knowledge and performs logical reasoning to determine the best course of action.

Key characteristics:

- Maintains an internal model or representation of the world
- Uses symbolic knowledge representation
- Performs explicit reasoning and planning
- Makes decisions based on logical inference
- Typically slower to respond but more thoughtful

This process involves explicit reasoning and planning, making it a deliberative approach. The agent "thinks before it acts," considering multiple possibilities and their outcomes.

Figure 1 illustrates the classic deliberative agent architecture, which follows a Sense-Plan-Act cycle. The architecture consists of three main internal components:

- **Deliberative Planner:** Generates high-level plans based on goals and current state information. It receives failure signals from the monitoring component and creates or refines plans accordingly.
- **Monitoring:** Monitors the execution of the plan, compares the current state with the expected state, and sends failure signals to the planner if deviations occur. It receives plans from the planner and sends specific actions to the execution component.
- **Execution:** Interacts directly with the environment through sensors and actuators. It receives sensor data from the environment, updates the monitoring component with the current state, receives action commands from monitoring, and executes actions in the environment.

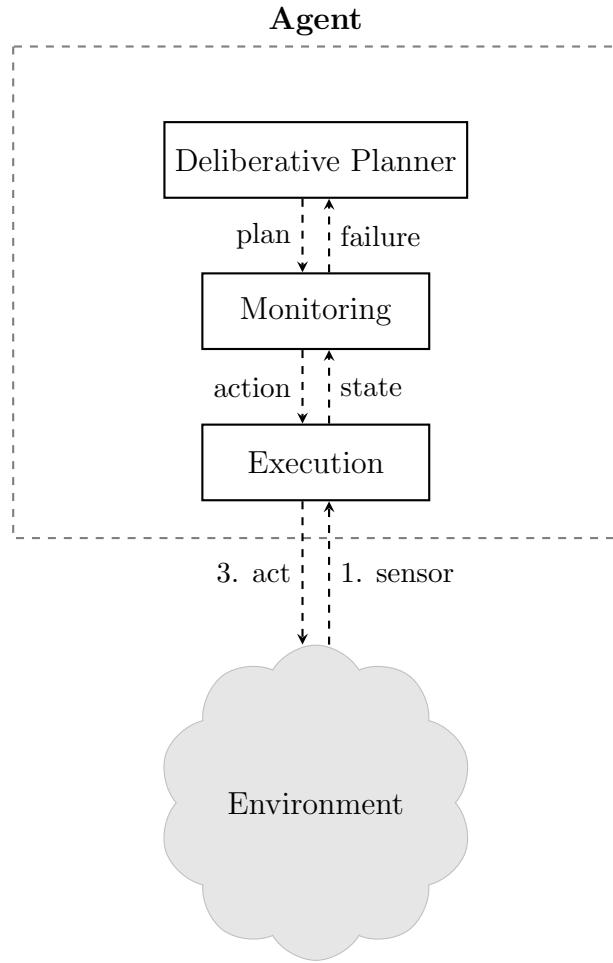


Figure 1: Classic deliberative agent architecture (Sense-Plan-Act cycle)

The flow operates as a continuous cycle:

1. The agent **senses** the environment through sensors (1. sensor), providing raw data to the Execution component.
2. Execution updates Monitoring with the current **state**.
3. Monitoring compares the state with the **plan** received from the Deliberative Planner and determines the next **action** for Execution. If the plan is not progressing as expected, Monitoring sends a **failure** signal to the Deliberative Planner.
4. The Deliberative Planner receives failure signals and generates or refines a **plan**, which is sent to Monitoring.
5. Execution **acts** (3. act) upon the environment based on the action received from Monitoring.

This cycle represents how a deliberative agent continuously perceives, plans, and acts to achieve its goals, with a mechanism for detecting and responding to plan failures.

4.1.1 Examples of Deliberative Architectures

- **Planning agents:** These agents use automated planning algorithms to generate sequences of actions that achieve specific goals. They maintain a symbolic representation of the world state and use search algorithms to find optimal or near-optimal plans. Planning agents are commonly used in robotics, autonomous systems, and game AI where complex sequences of actions need to be coordinated.
- **Belief Desire & Intention (BDI):** This architecture models agents based on three mental attitudes: **Beliefs** (what the agent knows about the world), **Desires** (the agent's goals or objectives), and **Intentions** (the commitments to specific plans of action). BDI agents reason about their beliefs, select desires to pursue, and commit to intentions (plans) to achieve those desires. This architecture is particularly useful for modeling complex, goal-oriented behavior in multi-agent systems and autonomous agents.

4.2 Reactive Architecture

A **reactive architecture** (also known as a *behavior-based* architecture) is characterized by the agent's direct mapping from perceptions to actions without maintaining an internal world model or engaging in complex reasoning. The agent responds quickly to environmental stimuli through simple stimulus-response rules.

Key characteristics:

- No internal world model or symbolic representation
- Direct mapping from sensors to actuators
- Simple stimulus-response rules or behaviors
- Fast response time
- Emergent behavior from simple rules

Example: Consider a simple obstacle-avoidance robot with a reactive architecture. The robot has sensors on its front and sides, and it follows these simple rules:

- If the front sensor detects an obstacle, turn right
- If the right sensor detects an obstacle, turn left
- If no obstacles are detected, move forward
- If both sensors detect obstacles, move backward

The robot doesn't maintain a map of its environment or plan a path. It simply reacts to what it perceives at each moment. This makes it very fast and efficient for simple tasks, though it may not find the optimal path. The robot's navigation behavior emerges from these simple reactive rules.

Figure 2 illustrates a reactive agent architecture that incorporates a **Reactive Planner** component:

- **Reactive Planner:** Positioned at the top, this component generates reactive action sequences in direct response to the current state. Unlike deliberative planners, it does not maintain a complex world model but instead quickly generates actions based on immediate perceptions. It receives failure signals from Monitoring and sends actions back to Monitoring.
- **Monitoring:** Positioned in the middle, it monitors the execution of actions and the current state. It receives state information from Execution, sends failure signals to the Reactive Planner when deviations occur, receives actions from the Reactive Planner, and sends action commands to Execution. It also receives external plans.
- **Execution:** Located at the bottom, it interacts directly with the environment through sensors and actuators. It receives sensor data from the environment, updates Monitoring with the current state, receives action commands from Monitoring, and executes actions in the environment.

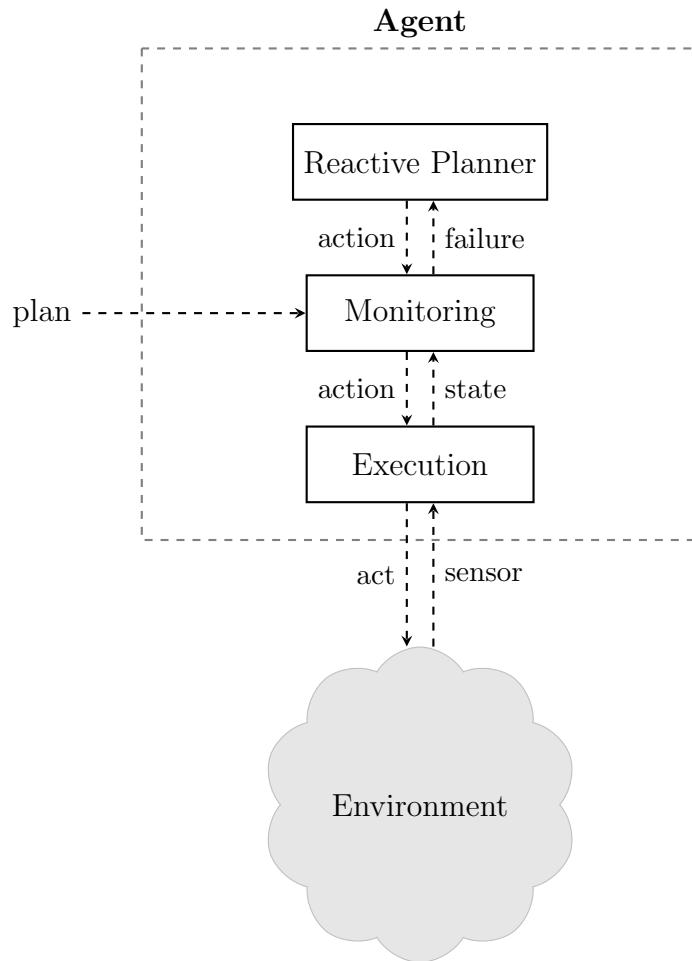


Figure 2: Reactive agent architecture with Reactive Planner

The flow operates as follows:

1. An external **plan** is provided to the Monitoring component (this could come from a higher-level planner or user input).

2. The Environment sends **sensor** data to the Execution component, providing raw information about the current state of the environment.
3. Execution processes the sensor data and updates Monitoring with the current **state**.
4. Monitoring evaluates the state against the plan. If there is a deviation or problem, it sends a **failure** signal to the Reactive Planner.
5. The Reactive Planner receives the failure signal and immediately generates a reactive **action** response, which it sends back to Monitoring.
6. Monitoring receives the action from the Reactive Planner and forwards the **action** command to Execution.
7. Execution **acts** upon the environment based on the action received from Monitoring.

This architecture emphasizes fast, reactive responses to environmental changes while still allowing for external plan guidance. The Reactive Planner provides quick adaptation without the overhead of maintaining complex internal models. The bidirectional flow between Monitoring and the Reactive Planner enables rapid failure detection and action generation.

4.2.1 Examples of Reactive Architectures

- **Subsumption architecture:** A layered architecture where behaviors are organized in levels of increasing complexity. Lower-level behaviors (like obstacle avoidance) can subsume or override higher-level behaviors (like exploration) when triggered by environmental conditions. Each layer operates independently and reactively, with no central control or world model.
- **Agent network architecture from Pattie Maes:** A distributed reactive architecture where multiple simple agents (or behaviors) are connected in a network. Each agent has local rules and can activate or inhibit other agents. The overall behavior emerges from the interactions between these agents, without centralized planning.
- **Reactive execution model:** Is domain independent and operates with structures precalculated at runtime.

4.3 Hybrid Architecture

A **hybrid architecture** combines elements of both deliberative and reactive architectures. It typically has multiple layers: a reactive layer for fast, immediate responses to critical situations, and a deliberative layer for complex planning and reasoning. This architecture attempts to get the best of both worlds: the speed of reactive systems and the intelligence of deliberative systems.

Key characteristics:

- Combines reactive and deliberative components
- Multiple layers of control (reactive at the bottom, deliberative at the top)

- Fast response for urgent situations (reactive layer)
- Complex reasoning and planning for strategic decisions (deliberative layer)
- Coordination between layers

Example: Consider an autonomous vehicle with a hybrid architecture. The vehicle has two main layers:

- **Reactive layer:** Handles immediate, critical situations. For example:
 - If a pedestrian suddenly appears in front, immediately apply brakes (no time for planning)
 - If another vehicle swerves into the lane, quickly adjust steering
- **Deliberative layer:** Handles strategic planning and navigation. For example:
 - Plans the route from origin to destination
 - Analyzes traffic conditions and selects the best path
 - Decides when to change lanes based on traffic patterns
 - Maintains a map and tracks the vehicle's position

The reactive layer ensures safety by responding instantly to immediate threats, while the deliberative layer handles the overall navigation strategy. The layers work together: the deliberative layer sets the general plan, and the reactive layer handles unexpected situations that require immediate action.

4.3.1 Examples of Hybrid Architectures

- **Procedural Reasoning System (PRS):** A BDI (Belief-Desire-Intention) architecture that combines deliberative reasoning with reactive capabilities. The agent maintains **Beliefs** (facts about the world expressed in first-order logic), **Desires** (system behaviors or goals), and **Intentions** (the current set of active plans). PRS includes a library of partially specified plans called Knowledge Areas (KAs), each with an activation condition. KAs can be activated by goals or by data, and can be reactive, allowing PRS to respond quickly to environmental changes while also reasoning about which plans to execute.
- **COSY (Cooperative System):** A BDI architecture that combines elements from both PRS and IRMA architectures. It has five main components: **Sensors** (receive perceptual inputs), **Actuators** (perform actions), **Communications** (send messages), **Cognition** (mediates between intentions and knowledge to choose actions), and **Intention** (contains long-term goals and control elements). COSY combines deliberative reasoning with reactive communication and action capabilities, making it suitable for interactive and collaborative environments.

4.4 Cognitive Architecture

A **cognitive architecture** is a computational framework that models the structure and processes of human cognition. It provides a unified theory of how the mind works, including perception, memory, reasoning, learning, and decision-making. Cognitive architectures can be defined as a hypothesis about the fixed structures that provide a mind, whether in natural or artificial systems, and how they work together—along with the knowledge and skills incorporated within the architecture—to produce intelligent behavior in a diversity of complex environments. Cognitive architectures aim to create artificial agents that can exhibit human-like intelligence and behavior.

Key characteristics:

- Models human cognitive processes and structures
- Provides unified framework for multiple cognitive functions
- Includes memory systems (short-term and long-term)
- Supports learning and adaptation
- Integrates perception, reasoning, and action
- Based on cognitive science and psychology principles

4.4.1 Examples of Cognitive Architectures

- **ACT-R:** A cognitive architecture developed primarily by John Robert Anderson at Carnegie Mellon University. The most important assumption of ACT-R is that human knowledge can be divided into two irreducible types of representations:
 - **Declarative knowledge:** Represented as **chunks** (vector representations of individual properties, each accessible from a labeled slot)
 - **Procedural knowledge:** Represented as production rules

Chunks are maintained and accessed through **buffers**, which are the front-end of **modules** (specialized and largely independent brain structures). ACT-R has two main types of modules:

- **Perceptual-motor module:** Manages interaction with the environment, handling the flow of perception and action to connect the agent with the world.
- **Memory module:** Divided into:
 - * **Long-term memory** (production memory): Contains production rules
 - * **Short-term memory** (working memory or declarative memory): Contains current facts about the world
- **SOAR:** A cognitive architecture created at Carnegie Mellon University by Laird, Newell, and Rosenbloom (1987). The ultimate goal of SOAR is to provide a foundation for a system capable of general intelligent behavior, supporting the full range of cognitive tasks, problem-solving methods, and knowledge representations. SOAR is both a theory of cognition and a computational implementation of that theory.

The design of SOAR is based on the hypothesis that all goal-oriented deliberate behavior can be understood as the selection and application of **operators** to a **state**:

- **State:** A representation of the current problem situation
- **Operator:** Transforms a state (performs changes in the representation)
- **Goal:** A desired result for the problem

SOAR runs continuously, attempting to apply the current operator and select the next operator (a state can have only one operator at a time) until the goal is achieved.

SOAR has separate memories with different representation modes:

- **Short-term memory:** Stores sensor data, intermediate inferences about current data, currently active goals, and active operators (actions and plans)
- **Long-term memory** (production memory): Maintains knowledge for responding to situations through procedures. It stores problem-solving knowledge, inference rules, and knowledge for selecting and applying operators in specific states

4.5 Other Architectures

- Three layer architecture
- Multilayer architecture
- Three tower architecture

5 Summary

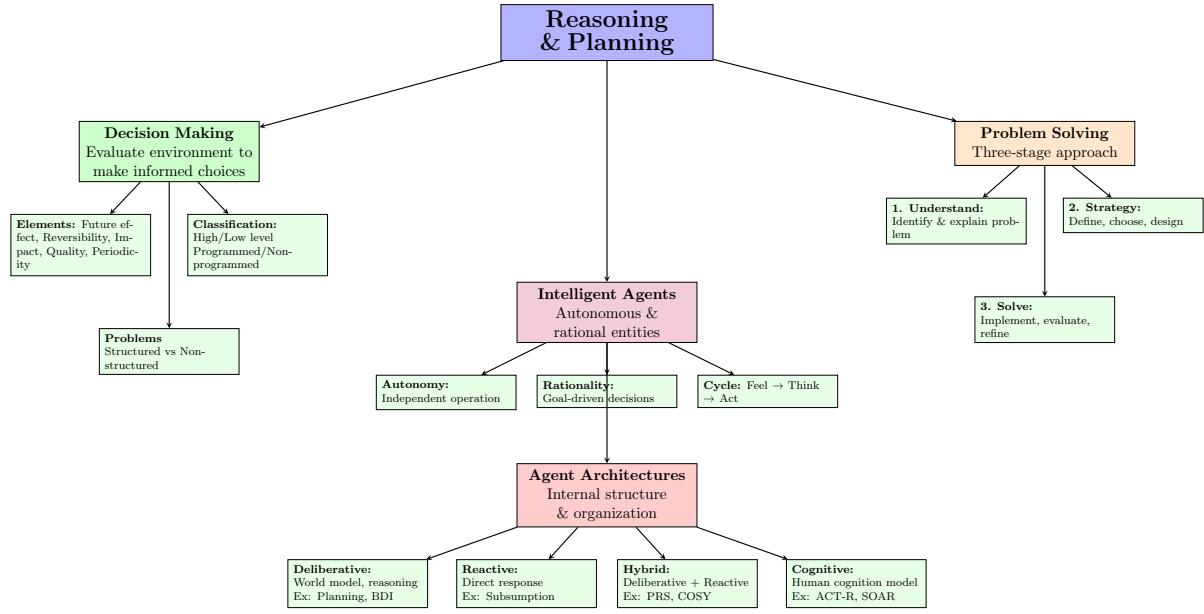


Figure 3: Conceptual map of the Reasoning and Planning course