Chapter 4: Agent Models and Architectures

I. Introduction

An agent perceives the environment using its sensors and it acts on its environment using its effectors. So, an agent can always be seen as a function linking its perceptions to its actions. We obtain a typology of agents if we answer the question: why does an agent act?

Answer: How to implement the actions efficiently according to the current situation of the environment or other agents.

Multi-agent systems are generally classified into two main families:

- cognitive systems
- and reactive systems.

II. Agent models

II.1. reactive agent

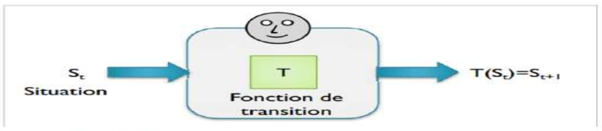
Because it reacts to external events. A reactive agent just reacts immediately to changes in the environment.

- they act according to a pattern of stimuli/reaction (response) to events coming from their environment.
- according to a reflection pattern by responding to signals from the environment

II.1.1. Characteristics of a reactive agent

- Individually, reactive agents are not "intelligent" (without anticipation, without planning)
- because of their large number, these reactive agents can solve problems described as complex
- No explicit representation of their environment or other agents.
- Situated actions (ants)
- Communication via the environment by propagating signals (the environment is fundamental for the coordination of actions between several agents)
- Work on these agents is more interested in modeling a society of agents
- areas of research: artificial life, ethology (anthill, termite nest, beehive), etc.

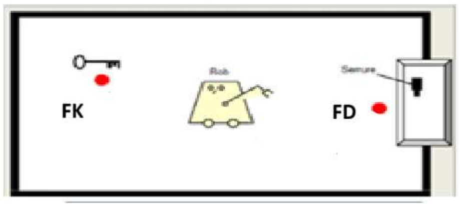
- it simply contents itself with acquiring perceptions and Reacting to them by applying certain pre-defined type rules (situation, action).
- it is likened to a transition function T: this function is an unordered set of type rules:
    - If condition then action
    - which are executed in an infinite and non-deterministic loop
    - behavior driven by perceptions



```
rules : règles condition-action
percepts : ensemble de percepts
Repeat
percept = perceive();
state := interpret_input(percept) ;
rule := match(state, rules) ;
execute(rule[action]) ;
forever
```

Example of a Reactive agent

- If FD and DO Then EXIT
- If FD and DC Then OPEN
- If FD and DL and K Then UNLOCK
- If FD and DL Then RANDOM-WALK
- If FK and K Then RANDOM-WALK
- If FK and Not K Then TAKE-KEY



K: Pop has the key
FD: in Front of the Door
FK: in Front of the Key
DO: Door Open
DC: Door Closed
DL: Door locked

Example of a Reactive agent

- A guard in a game
- As long as I see nothing, I follow my path
- If I see an enemy
- If he is not threatening and if I am not hurt, I attack him
- If he is threatening or if I am hurt, I sound the alarm, and I walk away

II.2. Cognitive agent

Each cognitive agent is specialized in a domain and can communicate with other agents. Agents are immersed in an environment in which they interact. Their structure revolves around three main functions: perceive, decide and act. It has a symbolic representation (memory capacities) of itself, its environment and other agents and is endowed with reasoning capacities. Each cognitive agent has a knowledge base including:

- various information related to their areas of expertise
- and the management of interactions with other agents and with its environment.

They have explicit goals and plans to accomplish their goals => intentional: Planning (human beings) intentional behavior (explicit goals and plans)

- Small groups of individuals can solve problems cooperatively, that need to coordinate their activity, and sometimes negotiate to resolve their conflicts

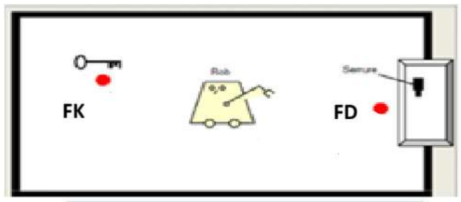
II.2.1. How a cognitive agent works

The cognitive agent has a plan P which it executes sequentially and deterministically and thus has sub-functions:

- revision of beliefs,
- cooperation (negotiation, coordination),
- learning.

Example of a cognitive agent

- GO-TO FD
- If DO Then EXIT
- Else If DC Then OPEN ;EXIT
- Else If DL Then If K Then UNLOCK;OPEN;EXIT
- Else GO-TO FK; TAKE KEY; GO TO FD; UNLOCK; OPEN; EXIT I



K: Pop has the key
FD: in Front of the Door
FK: in Front of the Key
DO: Door Open
DC: Door Closed
DL: Door locked

I.3. Deliberative agent

Deliberative agents are agents that perform some deliberation to choose their actions. Such a deliberation can be done based on the goals of the agent or on a certain utility function. It can take the form of a plan that reflects the sequence of actions that the agent must perform in order to achieve its goal

II.4. Hybrid agent

- Combination between the cognitive agent and the reactive agent.
- Each of these previous categories is appropriate for a certain type of problem.
- Nevertheless, for the majority of problems, neither a completely reactive architecture, nor a completely deliberative or cognitive architecture is appropriate,
- Agents must be able to react very quickly in certain situations (reflex behavior), while in others they must behave with more reasoning.

II.4.1. Reactive Agents vs Cognitive Agent

Cognitive Agent	**Reactive Agents**
Explicit representation of the environment	No explicit representation
Can take into account its past	No memorization of its history
Complex agent	Function stimulus/reaction
Small number of agents	Big number of agents

II.5. Purely communicating agent

(or software agent) an IT entity that:

- Exists in an open computing system (heterogeneous networks and systems or a collection of processors, software and resources that they can use and they interact with)
- can communicate with other agents,
- has skills (services) that it can offer to other agents,
- has a behavior, taking into account the resources and skills at its disposal and according to the communications it receives.
- Compared to the notion of agent in general, A purely communicating agent is distinguished:
    - it has no perception of other agents
    - it does not act in an environment (its development context is naturally that of computer networks).

II.6. Purely situated agent

- Purely situated agent: a physical entity that:

- is located at a position in an environment •
- able to perceive (but in a limited way) its environment
- has a behavior, taking into account the resources and skills at its disposal and the perceptions
- compared to the notion of agent in general, A purely situated agent is distinguished:
  - communications are generally not made directly, but indirectly through perceptions and their actions in the environment.

II.7. Mobile Agent

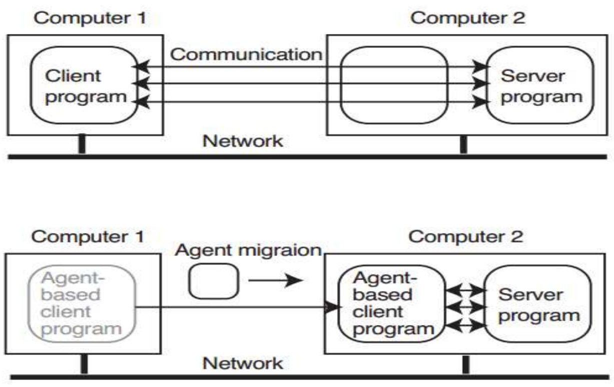
Initially introduced in 1994 with the Telescript environment. The mobile agent paradigm lies at the crossroads of two research areas: multi-agent systems and code mobility with more particularly process migration.

Mobile agents bring to these so-called stationary agents the ability to move on the different nodes of a network in order to work locally on the resources.

- an agent capable of migrating autonomously to the different sites of a network,
- first of all, the mobile agent code and data are hosted in computer A
- data and code are migrated to computer B.
- After the migration, the code is executed with the data and resources available on computer B.

They can provide a practical, efficient and robust framework for implementing distributed applications and intelligent environments such as grid computing for several reasons:

- the search for performance
- ease of adaptability and development of distributed applications.
- reduced communication costs
- reduce vulnerability
- asynchronous execution
- dynamic software deployment,
- has a behavior,
- They can reduce communication costs



III. Agent architectures

III.1. Agent structure

There are several ways to design agents, but regardless of the architecture adopted, an agent can always be seen as a function linking its perceptions to its actions. This function is realized and implemented by:

- agent program that runs on a
- architecture which also organizes the interface with the environment (perceptions, actions)

Agent = Architecture + Program (behaviour)

The characteristics of an architecture are:

- **physical:** a kind of processing equipment with physical sensors and effectors (ordinary PC, robotic car with several onboard computers, cameras, and other sensors) **or virtual** as in the case of software agents
- the architecture disposes the available perceptions of sensors to the program, executes the program, and puts the actions chosen, by the latter, to the effectors.

III.2. Agent program

The agent program implements the desired agent function. It is invoked for each new percept and returns an action each time: just takes the current percept from the sensors as input and returns an action to the effectors. The actions taken describe the behavior of the agent. An agent implements a behavior based on:

- what it can do (its skills),
- its perceptions,
- its representation of the world
- and the communications it has with others or with its environment. The principal of IA is:
    - how to write programs that produce rational behavior from a small amount of code rather than a large number of table entries
    - The agents have the same skeleton: acquiring percepts from an environment and generating actions using internal data structures, updated with each new percept arrived.

```
function SKELETON-AGENT(percept) returns action
  static:memory  //the agent's memory of the world
    memory←UPDATE-MEMORY(memory,percept)
    action←CHOOSE-BEST-ACTION(memory)
    memory←UPDATE-MEMORY(memory,action)
  return action
```

III.2.1. Agent Program Types

Russel and Norvig group agent programs into several types, namely:

- TABLE-DRIVEN "Table-lookup" agent example: automated taxi
- Simple reflex agents
- Reflex agents based on models (with internal model of the world)
- Goal-based agents
- Utility-Based Full Agents

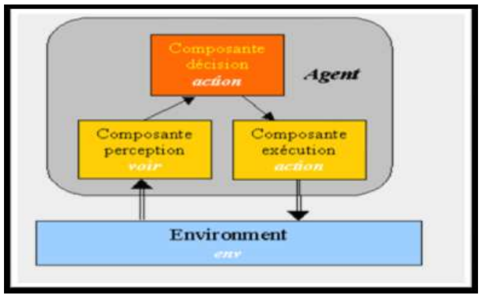
The reflex types of agents are considered reactive agents and the latter two types are considered deliberative agents.

III.3. Agent Architecture

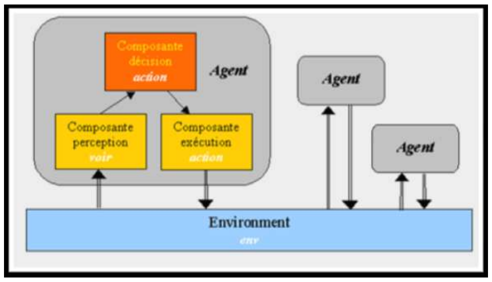
From a designer's point of view, the architecture of an agent describes the internal structure (software or hardware): how to assemble the different parts of an agent so that it performs the actions that are expected of it to from a set of inputs?

In other words, the architecture of an agent is a particular methodology to specify the construction of agents in a set of modules. The architecture of an agent is a description of its internal organization: the knowledge of the agent, the operations that can be performed and the control flow of operations. The choice of an architecture depends on:

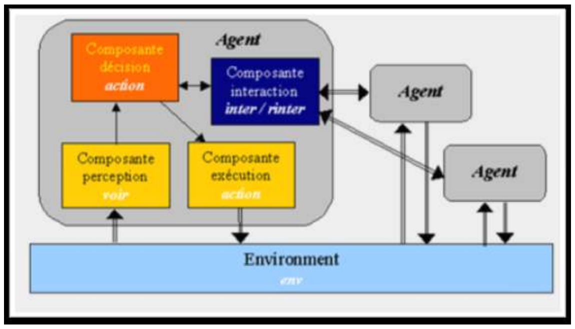
- designer's decision
- how perceptions relate to actions.


Composante décision action
Agent
Composante perception voir
Composante exécution action
Environment env

General structure of an agent (mono)


Composante décision action
Agent
Composante perception voir
Composante exécution action
Agent
Agent
Environment env

General structure of an interacting agent (communication via env)


Composante décision action
Agent
Composante interaction inter / rinter
Agent
Agent
Composante perception voir
Composante exécution action
Environment env

General structure of an interacting agent (communication via env and direct)

III.4. Types of architecture

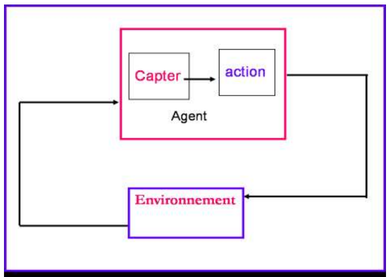
III.4.1. Abstract Architecture

- S = {s1, s2, …} set of states of the environment.
- The skills of an agent are represented by the set of actions it can perform A = {a1, a2, …},
- An agent can be seen as a function: Act: S*➔ A
- which maps a sequence (*) of states of the environment to actions.

III.4.1.1. Abstract Architecture (Purely Reactive Agent)

Agents decide what to do without reference to histories:

- Decision making is based only on the present without any reference to the past.
- sometimes called tropic agents because they simply respond directly to their environment.
- formally, the behavior of a purely reactive agent can be represented by the function: action: S ➔ A
- The abstract model must be refined (break down into subsystems)
- The first functional decomposition of an agent is made between perception and its decision process on actions
- Let P be the set of perceptions and "capture" the function which makes the states of the environment correspond to the perceptions
- capture: S ➔ P
- Act: P* ➔ A
- 2 states s1 and s2 are equivalent for the agent if it perceives them in a unique way: s1∈ S and s2 ∈ S and capture(s1)=capture(s2).

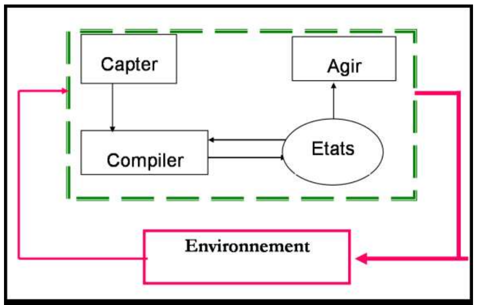


Sous-systèmes de perception et d'action.

III.4.1.2. Abstract Architecture (Agent with state)

To use its past, an agent can keep its history. Let "I" be the set of internal states of the agent, the decision process is expressed from I, to act: I ➔A

- This memorization process requires the accumulation of knowledge by the function: Compile (deliberate): I * P ➔ I
- therefore, the agents operate according to the new cycle: capture - compile - act.
- The agent perceives the world through capture, updates its state through compile, then chooses an action to perform through act,



Agents et états internes.

III.4.2. Concrete architecture

III.4.2.1. BDI architectures

A BDI architecture is central to the BDI theory of rational action, first proposed by Michael Bratman. The agent decides on the actions to be taken from its internal states which are expressed in the form of:

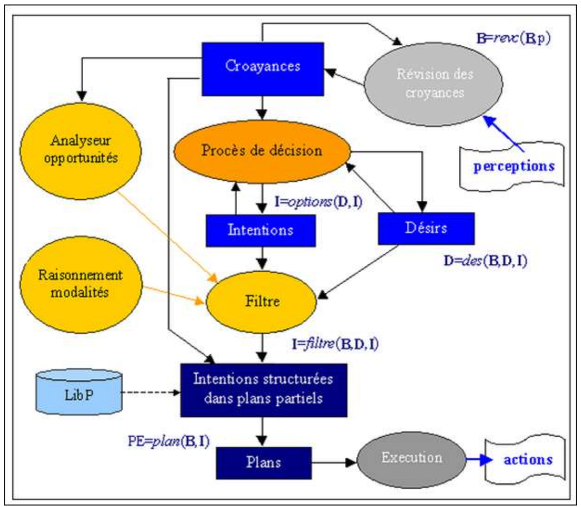
- **Belief**: knowledge of the world; the information that the agent has about the environment and about other agents, through its ability to perceive or through interaction with other agents, which is used to calculate the values of the desires (criteria).
- **Desire**: goals of the agent, represent the states of the environment, which the agent would like to see realized. are formalized as a set of criteria that will be used to evaluate plans to enable the agent to achieve its goals.
- **Intention**: are the desires that the agent has decided to accomplish or the actions that it has decided to do to accomplish its desires corresponds to the chosen instantiated plan The "classic" agent systems that have implemented the BDI architecture are: IRMA = Intelligent Resource-bounded Machine Architecture, and PRS = Procedural Reasoning System
- the control cycle of a BDI agent:

```
1 B:=B0
2 D:=D0
3 I:=I0
4 répéter
    4.1 obtenir nouvelles perceptions P
    4.2 B:=révision-croyance(B, P)
    4.3 D:=révision-désirs(B, D ,I)
    4.5 I:=filtre-intentions(B, D, I)
    4.6 PE:=plan(B, I)
    4.7 exécuter(PE)
  jusqu'à ce que l'agent soit arrêté

Où exécuter (PE)s'écrit comme :

tantque (PE<>{} et nonaccompli(I, B) et possible(I, B)) répéter
      4.7.1 x = première(PE);  exécuter(x);  PE = reste(PE)
      4.7.2 obtenir nouvelles perceptions p
      4.7.3 B = revc(B, p)

fin tant que
```



Example: How farmers improve cropping system using BDI agent-based architecture?

- **Proposed solution:** This model includes two types of entities:
- **agricultural agents**: has the following state variables: Plots, plans, intention, desires, beliefs (memory-last-productions, weather-last-days, price-crop, cost-crop), water, finance.
- **the base of desires (4 desires (criteria)):** Maximize profit, Minimize financial risks, Minimize workload, Maximize similarities with the last chosen plan
- **The following belief base (attributes):** memory-last-productions, weather-last-days, crop-price, crop-cost, water.
- **intention:** current cropping plan.

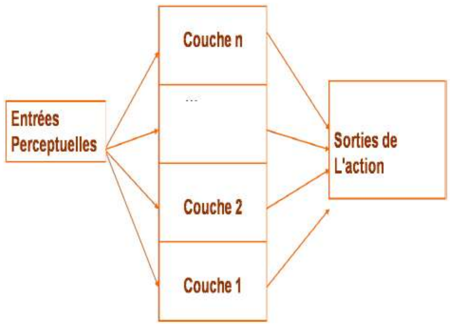
- **parcel agents**

III.4.2.2. Layered architectures

- Typically, at least 2 layers, one for reactive behavior and one for proactive behavior,
- Ability to design multiple layers
- Topology: information and flow control between several layers
- two types of architectures: horizontal and vertical.

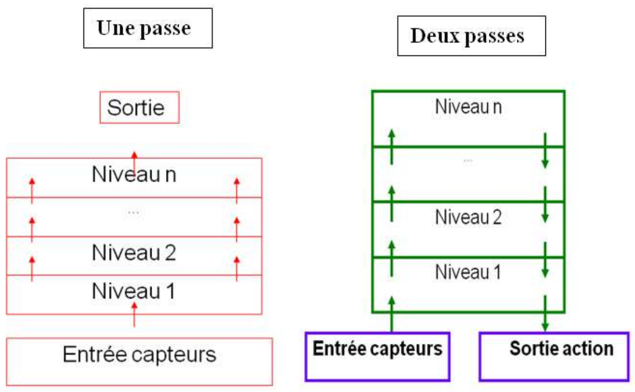
III.4.2.2.a. Horizontal layered architectures

- In this architecture, the software layers are linked to sensor inputs and action outputs.
- Each layer behaves like an agent and produces proposals on the type of action to be performed ➔independence and conceptual simplicity.
- for "n" different types of behavior, then we implement "n" different layers.
- competition between layers to suggest actions ➔ inconsistency problem of the global behavior of the agent.
- need for layer mediation: making decisions about which layer will have control over the agent at any time.



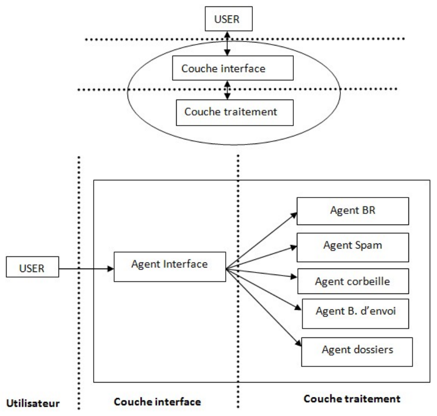
III.4.2.2.b. Vertical layered architectures

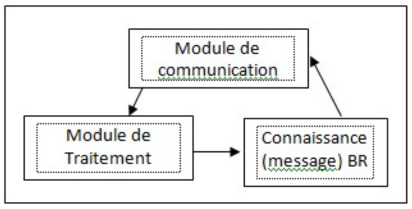
- In this architecture, sensor inputs and action outputs are supported by at most one layer.
- There are one-pass and two-pass vertical architectures.
  - ➢ **One pass:** in this architectural model, the information arrives from the sensors on a specialized layer, then passes through other layers in sequence until the last one which drives the output (action to be taken).
  - ➢ **Two passes**: in the two-pass architecture, the same layer provides the agent's input and output interface. The information follows the same path as in the previous mode, then descends the architecture in the opposite direction to return to the single front-end layer (interface) of the agent.



Example of a layered architecture

- development of intelligent electronic messaging (E-mail) by exploiting the agent paradigm to model such a system (PhD competition).
- the use of the agent paradigm for modeling this system because the system has autonomous components that act in parallel A global layer architecture of the system An architecture for the agents of the system

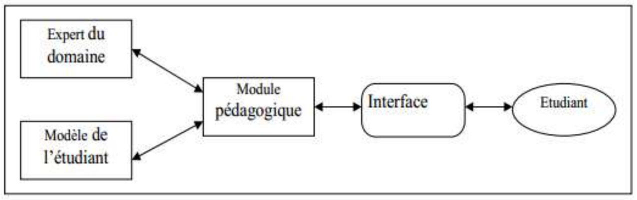


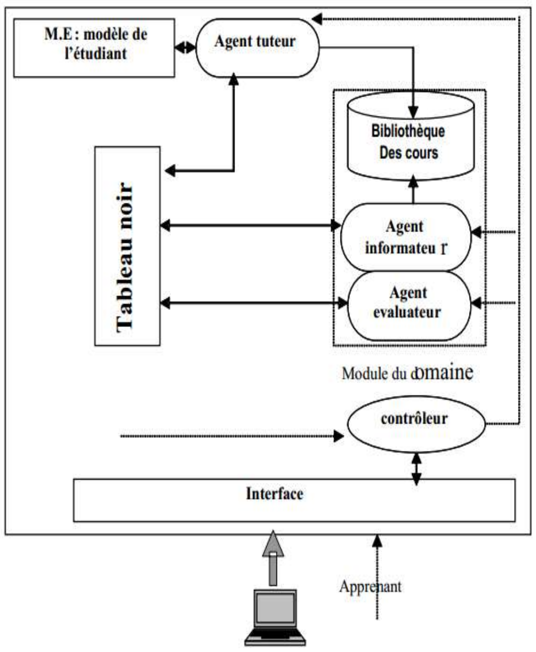


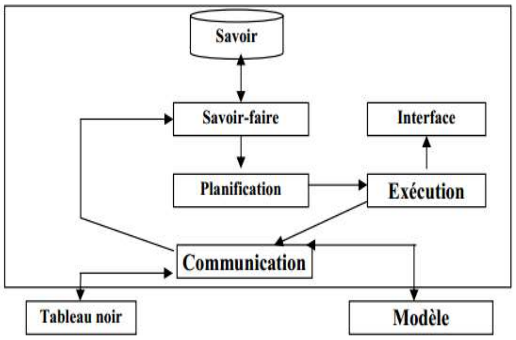
Example of a layered architecture

- development of an intelligent tutor (computer-assisted intelligent teaching system) using a multi-agent approach.
- The problem to be solved is that of teaching the main database concepts where the different types of knowledge come from three sources: the pedagogical expert, the expert in the field to be taught and the student.
- complex pbm, distribution and heterogeneity of different types of knowledge on a set of communicating agents
- the E.I.A.O system (intelligent tutors) is organized around three modules:
  - the domain (problem solving or expertise module)
  - the student model

- and the pedagogical expert (tutorial model)







Agent pédagogue