INTELLIGENT AGENTS

Intelligent Agents and Multiagent Systems
Josefa Z. Hernández, Nik Swoboda
{phernan,nswoboda}@fi.upm.es
Agent properties: Autonomy

- The main point about agents is they are *autonomous*: capable of acting independently, exhibiting control over their internal state.

- Thus:

  *an agent is a computer system capable of autonomous action in some environment in order to meet its design objectives*
Agent properties: Autonomy & Flexibility

- Trivial (non-interesting) agent:
  - thermostat

- An intelligent agent is a computer system capable of flexible autonomous action in some environment (Jennings et al., 98)

- By flexible, we mean:
  - reactive: on going interaction with the environment and on time response
  - pro-active: generating and attempting to achieve goals; not driven solely by events; taking the initiative
  - social: ability to interact with other agents
Other properties

- **Mobility**: the ability of an agent to move around an electronic network;

- **Veracity**: an agent will not knowingly communicate false information;

- **Benevolence**: agents do not have conflicting goals, and that every agent will therefore always try to do what is asked for;

- **Rationality**: agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved - at least insofar as its beliefs permit;

- **Learning/adaptation**: agents improve performance over time.
Environment properties

Accessible vs. inaccessible

Can the agent “see” everything?

Deterministic vs. non-deterministic

Do actions have guaranteed effect?

Static vs. dynamic

Does the environment change on its own?

Discrete vs. continuous

Is the number of actions and percepts finite?
Agents and Objects

- Are agents just objects by another name?

- Object:
  - encapsulates some state
  - communicates via message passing
  - has methods, corresponding to operations that may be performed on this state
Agents and Objects

Main differences:

- **agents are autonomous:** agents embody stronger notion of autonomy than objects, and in particular, they decide for themselves whether or not to perform an action on request from another agent.

- **agents are smart:** capable of flexible (reactive, pro-active, social) behavior, and the standard object model has nothing to say about such types of behavior.

- **agents are active:** a multi-agent system is inherently multi-threaded, in that each agent is assumed to have at least one thread of active control.

---

*Objects do it for free...*  
*agents do it because they want to*  
*agents do it for money*
Agents as intentional systems

- **Design Stance**: Use knowledge about design intention of an object to predict behavior (Example: Alarm Clock)

- **Physical Stance**: Observe $\rightarrow$ analyze function principles (induce general description) $\rightarrow$ predict future behavior (through deduction). (Example: Apple and Newton’s second law)

- **Intentional Stance**: Attributing attitudes (beliefs, desires, whishes) to systems whose precise internal function is unknown. (Controversial example: Light switch)

Most real world systems are **too complex** for physical or design stance. $\rightarrow$ Why not use **intentional stance** as means of **complexity reduction**?
Agent architectures

- Originally (1956-1985), pretty much all agents designed within AI were *symbolic reasoning* agents.

- Its purest expression proposes that agents use *explicit logical reasoning* in order to decide what to do.

- Problems with symbolic reasoning led to a reaction against this – the so-called *reactive agents* movement, 1985-present.

- From 1990-present, a number of alternatives proposed: *hybrid (cognitive) architectures*, which attempt to combine the best of reasoning and reactive architectures.
Deliberative/symbolic agents

- Key problems to face:
  - The transduction problem. Translate the real world into an accurate symbolic description (...vision, speech understanding, learning).
  - The representation problem. How can a symbolic description language represent information about the real world entities and processes (E.g. FOL)
  - The reasoning problem. How to get agents to reason with this information (symbolic descriptions) in time for the results to be useful (...knowledge representation, automated reasoning, automated planning).

- These problems are far from being solved

- Underlying problem: symbol manipulation algorithms are usually highly intractable.
A **deliverative agent** or agent architecture:
- contains an explicitly represented, symbolic model of the world
- makes decisions (e.g. about what actions to perform) via symbolic reasoning

How can an agent decide what to do using theorem proving?
- Basic idea is to use logic to encode a theory stating the best action to perform in any given situation

Let:
- $\rho$ be this theory (typically a set of deduction rules)
- $\Delta$ be a logical database which describes the current state of the world
- $Ac$ be the set of actions the agent can perform
- $\Delta \vdash_{\rho} \varphi$ mean that $\varphi$ can be proved from $\Delta$ using $\rho$
Deductive reasoning agents: agents as theorem provers

/* try to find an action explicitly prescribed */
for each $\alpha \in \text{Ac}$ do
    if $\Delta \vdash_{\rho} Do(\alpha)$ then
        return $\alpha$
    end-if
end-for

/* try to find an action not excluded */
for each $\alpha \in \text{Ac}$ do
    if $\Delta \not\vdash_{\rho} \neg Do(\alpha)$ then
        return $\alpha$
    end-if
end-for

/* no action found */
return null
Deductive reasoning agents: the vacuum world

- Domain predicates: \( \text{In}(x,y), \text{Dirt}(x,y), \text{Facing}(d) \)
- Possible actions: \( Ac = \{\text{turn}, \text{forward}, \text{suck}\} \) (\( \text{turn} = \text{turn right 90 degrees} \))
- Rules \( \rho \) for determining what to do:

  \[
  \begin{align*}
  \text{In}(0,0) \land \text{Facing(north)} \land \neg \text{Dirt}(0,0) & \rightarrow \text{Do(forward)} \\
  \text{In}(0,1) \land \text{Facing(north)} \land \neg \text{Dirt}(0,1) & \rightarrow \text{Do(forward)} \\
  \text{In}(0,2) \land \text{Facing(north)} \land \neg \text{Dirt}(0,2) & \rightarrow \text{Do(turn)} \\
  \text{In}(0,2) \land \text{Facing(east)} & \rightarrow \text{Do(forward)}
  \end{align*}
  \]

  …… and so on

- Using these rules (+ other obvious ones), starting at \( (0,0) \) the robot will clear up dirt

  **Problems:**
  - How to convert video camera input to \( \text{Dirt}(0,1) \)?
  - Decision making assumes a static environment: *calculative rationality*
  - Decision making using FOL is *undecidable*
Practical reasoning agents

- **Practical reasoning** is reasoning directed towards **actions** — the process of figuring out what to do (vs. **theoretical reasoning** — directed towards beliefs):
  - “Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes.” (Bratman)

- Human practical reasoning consists of two activities:
  - **Deliberation**: deciding **what** state of affairs we want to achieve
    Dynamic environments, computational costs of deliberation → deliberation cannot last indefinitely. Agent must control its deliberation-reasoning
    The outputs of deliberation are **intentions** (goals the agent has committed to)
  - **Means-Ends Reasoning (MER)**: deciding **how** to achieve these states of affairs
    Result of MER: plan → Then: commit to plan (or post-cond. of plan rsp.), execute plan → hope: goal reached
Intentions in Practical Reasoning: Advantages

- **Intentions drive means-ends reasoning.**
  If an agent have formed an intention then it will attempt to achieve the intention, which involves, among other things, deciding how to achieve it. Moreover, if one particular course of action fails to achieve an intention, then it will typically attempt others.

- **Intentions persist.**
  An agent will not usually give up on its intentions without good reason.

- **Intentions constraint future deliberation.**
  An agent will not consider options that are inconsistent with its current intentions.

- **Intentions influence beliefs upon which future practical reasoning is based.**
  If an agent adopt an intention, then it can plan for the future on the assumption that it will achieve the intention.
Practical reasoning agents: Deliberation

- **Beliefs, Desires, Intentions**: Symbolically represented:
  \[ Bel = \{B, B', B'', \ldots\} \quad Des = \{D, D', D'', \ldots\} \quad Int = \{I, I', I'', \ldots\} \]

- Deliberation = \(<\text{option}, \text{filter}>\)
  \[ \text{options} : 2^{Bel} \times 2^{Int} \rightarrow 2^{Des} \]
  \[ \text{filter} : 2^{Bel} \times 2^{Des} \times 2^{Int} \rightarrow 2^{Int} \]

- Option generation function \(\text{option}\) ➔ generates \(\text{desires}\) (goals)

- Filtering function \(\text{filter}\) ➔ selects \(\text{intentions}\) (commitments)

- Belief revision function \(\text{brf}\) ➔ updates \(\text{beliefs}\)
  \[ \text{brf} : 2^{Bel} \times Per \rightarrow 2^{Bel} \]
It’s the process of deciding how to achieve an end using the available means.

Basic idea is to give an agent:
- representation of goal/intention to achieve
- representation actions it can perform
- representation of the environment (agent’s beliefs)

And have it generate a plan to achieve the goal.
Set of predicates for world descriptions:
- $On(x, y)$: obj. $x$ on top of obj. $y$
- $OnTable(x)$: obj. $x$ is on the table
- $Clear(x)$: nothing is on top of obj. $x$
- $Holding(x)$: arm is holding $x$
- $ArmEmpty$: arm holds nothing

Here is a representation of the blocks world described above (agent’s belief):
- $Clear(A)$
- $OnTable(B)$
- $ArmEmpty$
- $On(A, B)$
- $OnTable(C)$
- $Clear(C)$

A goal is represented as a set of formulae (agent’s intentions). Example:
- $\{OnTable(A), OnTable(B), OnTable(C)\}$
Actions are represented using a technique developed in the STRIPS planner:

Each action has:
- a pre-condition list
  - list of facts which must be true for action to be executed
- a delete list
  - list of facts that are no longer true after action is performed
- an add list
  - list of facts made true by executing the action

Formally: An action $\alpha$ from a set of actions $Ac=\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is triple

$$\alpha=\langle P_\alpha, D_\alpha, A_\alpha \rangle$$

- $P_\alpha$: set of FOL formulae: Pre-conditions of $\alpha$
- $D_\alpha$: set of FOL formulae: Delete-set of $\alpha$
- $A_\alpha$: set of FOL formulae: Add-set of $\alpha$
Means-End Reasoning: STRIPS & The Blocks World

Blocks world operators (actions):

- **Stack(x, y)**
  
  $P_\alpha: \text{Clear}(y) \land \text{Holding}(x)$
  
  $D_\alpha: \text{Clear}(y) \land \text{Holding}(x)$
  
  $A_\alpha: \text{ArmEmpty} \land \text{On}(x, y)$

- **UnStack(x, y)**
  
  $P_\alpha: \text{On}(x, y) \land \text{Clear}(x) \land \text{ArmEmpty}$
  
  $D_\alpha: \text{On}(x, y) \land \text{ArmEmpty}$
  
  $A_\alpha: \text{Holding}(x) \land \text{Clear}(y)$

- **Pickup(x)**
  
  $P_\alpha: \text{Clear}(x) \land \text{OnTable}(x) \land \text{ArmEmpty}$
  
  $D_\alpha: \text{OnTable}(x) \land \text{ArmEmpty}$
  
  $A_\alpha: \text{Holding}(x)$

- **Putdown(x)**
  
  $P_\alpha: \text{Holding}(x)$
  
  $D_\alpha: \text{Holding}(x)$
  
  $A_\alpha: \text{Clear}(x) \land \text{OnTable}(x) \land \text{ArmEmpty}$
A planning problem is a triple \(<\Delta, O, \gamma>\)
- \(\Delta\): initial beliefs of agent
- \(O\): set of action (operator) descriptions
  \[ O = \{<P_\alpha, D_\alpha, A_\alpha> | \alpha \in Ac\} \]
- \(\gamma\): set of intentions

A plan is a sequence (list) of actions \(\pi = (\alpha_0, \alpha_1, \ldots, \alpha_n)\)

Start Goal
\[
\begin{array}{c|c|c}
A & B & C \\
\end{array} \quad \begin{array}{c|c|c}
A & B & C \\
\end{array}
\]

Plan: \textit{UnStack}(A,B), \textit{Putdown}(A)

A plan determines a sequence of \(n+1\) environment models \((\Delta_0, \Delta_1, \ldots, \Delta_n)\) with
\[
\Delta_0 = \Delta \quad \text{and} \quad \Delta_i = (\Delta_{i-1} \setminus D_\alpha_i) \cup A_\alpha_i \quad 1 \leq i \leq n
\]

A plan is acceptable iff \(\Delta_{i-1} \models P_\alpha_i\) \(1 \leq i \leq n\)

A plan is correct iff it is acceptable and \(\Delta_n \models \gamma\)
Means-End Reasoning: Planning

- In general, generating a plan can take exponential time on the set of operators.
- We do not need to generate the whole plan before acting, especially since our intentions might change in the meantime.
- In the mid 80’s, Chapman established some theoretical results which indicate that AI planners will ultimately turn out to be unusable in any time constrained system.
Implementing practical reasoning agents

Agent Control Loop Version 3
1.
2. \( B := B_0; \)
3. \( I := I_0; \)
4. while true do
5. \hspace{1em} \text{get next percept } \rho; \)
6. \hspace{1em} \( B := brf(B, \rho); \)
7. \hspace{1em} \( D := options(B, I); \)
8. \hspace{1em} \( I := filter(B, D, I); \)
9. \hspace{1em} \( \pi := plan(B, I); \)
10. \hspace{1em} \text{execute(}\pi\text{)} \)
11. end while
Commitment Strategies

Commitment implies *temporal persistence*. Critical issue: *How committed an agent should be to its intentions?*

“Some time in the not-so-distant future, you are having trouble with your new household robot. You say “Willie, bring me a beer.” The robot replies “OK boss.” Twenty minutes later, you screech “Willie, why didn’t you bring me that beer?” It answers “Well, I intended to get you the beer, but I decided to do something else.”

Miffed, you send the wise guy back to the manufacturer, complaining about a lack of commitment. After retrofitting, Willie is returned, marked “Model C: The Committed Assistant.” Again, you ask Willie to bring you a beer. Again, it accedes, replying “Sure thing.” Then you ask: “What kind of beer did you buy?” It answers: “Genessee.” You say “Never mind.” One minute later, Willie trundles over with a Genessee in its gripper. This time, you angrily return Willie for overcommitment.

After still more tinkering, the manufacturer sends Willie back, promising no more problems with its commitments. So, being a somewhat trusting customer, you accept the rascal back into your household, but as a test, you ask it to bring you your last beer. Willie again accedes, saying “Yes, Sir.” (Its attitude problem seems to have been fixed.) The robot gets the beer and starts towards you. As it approaches, it lifts its arm, wheels around, deliberately smashes the bottle, and trundles off.

Back at the plant, when interrogated by customer service as to why it had abandoned its commitments, the robot replies that according to its specifications, it kept its commitments as long as required — commitments must be dropped when fulfilled or impossible to achieve. By smashing the bottle, the commitment became unachievable.”

(Cohen, Levesque, 90)
Commitment Strategies

- The mechanism an agent uses to determine when and how to drop intentions is known as a commitment strategy.

- An agent has commitment both to ends (i.e., the wishes to bring about), and means (i.e., the mechanism via which the agent wishes to achieve the state of affairs).

- Currently, our agent control loop is overcommitted, both to means and ends.

- Modification: replan if ever a plan goes wrong.

Agent Control Loop Version 3

1. 
2. \( B := B_0; \)
3. \( I := I_0; \)
4. while true do
5. \( \text{get next percept } \rho; \)
6. \( B := \text{brf}(B, \rho); \)
7. \( D := \text{options}(B, I); \)
8. \( I := \text{filter}(B, D, I); \)
9. \( \pi := \text{plan}(B, I); \)
10. \( \text{execute}(\pi); \)
11. end while
Commitment Strategies

Agent Control Loop Version 4

1. \( B := B_0; \)
2. \( I := I_0; \)
3. while true do
4.   get next percept \( \rho; \)
5.   \( B := brf(B, \rho); \)
6.   \( D := \text{options}(B, I); \)
7.   \( I := \text{filter}(B, D, I); \)
8.   \( \pi := \text{plan}(B, I); \)
9.   while not empty(\( \pi \)) do
10.      \( \alpha := \text{hd}(\pi); \)
11.      \( \text{execute}(\alpha); \)
12.      \( \pi := \text{tail}(\pi); \)
13.      get next percept \( \rho; \)
14.      \( B := brf(B, \rho); \)
15.      if not sound(\( \pi, I, B \)) then
16.         \( \pi := \text{plan}(B, I) \)
17.      end-if
18.   end-while
19. end-while

Still overcommitted to intentions: Never stops to consider whether or not its intentions are appropriate

Modification: stop to determine whether intentions have succeeded or whether they are impossible
Commitment Strategies

Agent Control Loop Version 5
2. \( B := B_0; \)
3. \( I := I_0; \)
4. while true do
5. \( \text{get next percept } \rho; \)
6. \( B := brf(B, \rho); \)
7. \( D := \text{options}(B, I); \)
8. \( I := \text{filter}(B, D, I); \)
9. \( \pi := \text{plan}(B, I); \)
10. while not empty(\( \pi \))
   or succeeded(\( I, B \))
   or impossible(\( I, B \)) do
11. \( \alpha := \text{hd}(\pi); \)
12. \( \text{execute}(\alpha); \)
13. \( \pi := \text{tail}(\pi); \)
14. \( \text{get next percept } \rho; \)
15. \( B := brf(B, \rho); \)
16. if not sound(\( \pi, I, B \)) then
17. \( \pi := \text{plan}(B, I) \)
18. end-if
19. end-while
20. end-while

- It reconsiders its intentions when:
  - it has completely executed a plan to achieve its current intentions; or
  - it believes it has achieved its current intentions; or
  - it believes its current intentions are no longer possible.

- This is limited in the way that it permits an agent to *reconsider* its intentions

- *Modification:* Reconsider intentions after executing every action
Commitment Strategies

Agent Control Loop Version 6
1. \[ B := B_0; \]
2. \[ I := I_0; \]
3. while true do
   4. \[ B := brf(B, \rho); \]
   5. \[ D := options(B, I); \]
   6. \[ I := filter(B, D, I); \]
   7. \[ \pi := plan(B, I); \]
   8. while not (empty(\pi) or succeeded(\pi, B) or impossible(\pi, B)) do
      9. \[ \alpha := kd(\pi); \]
      10. \[ execute(\alpha); \]
      11. \[ \pi := tail(\pi); \]
   12. get next percept \( \rho \);
   13. \[ B := brf(B, \rho); \]
   14. \[ D := options(B, I); \]
   15. \[ I := filter(B, D, I); \]
   16. if not sound(\pi, I, B) then
      17. \[ \pi := plan(B, I) \] end-if
   18. end-while
21. end-while

- But intention reconsideration is **costly**! A dilemma:
  - an agent that *does not stop to reconsider* its intentions sufficiently often will continue attempting to achieve its intentions even after it is clear that they cannot be achieved, or that there is no longer any reason for achieving them
  - an agent that *constantly* reconsiders its intentions may spend insufficient time actually working to achieve them, and hence runs the risk of never actually achieving them

- **Solution:** incorporate an explicit *meta-level control* component, that decides whether or not to reconsider
Agent Control Loop Version 7

1. \[ B := B_0; \]
2. \[ I := I_0; \]
3. while true do
4.     get next percept \( \rho \);
5.     \[ B := brf(B, \rho); \]
6.     \[ D := options(B, I); \]
7.     \[ I := filter(B, D, I); \]
8.     \[ \pi := plan(B, I); \]
9.     while not (empty(\pi)
10.         or succeeded(\pi, I, B)
11.         or impossible(\pi, I, B)) do
12.         \[ \alpha := kd(\pi); \]
13.         execute(\alpha);
14.         \[ \pi := tail(\pi); \]
15.     get next percept \( \rho \);
16.     \[ B := brf(B, \rho); \]
17.     if reconsider(\pi, I, B) then
18.         \[ D := options(B, I); \]
19.         \[ I := filter(B, D, I); \]
20.     end-if
21.     if not sound(\pi, I, B) then
22.         \[ \pi := plan(B, I) \]
23.     end-if
24. end-while
Optimal Intention Reconsideration

(Kinny, Georgeff, 91) experimentally investigated effectiveness of intention reconsideration strategies in a BDI agent system.

Two different types of reconsideration strategy were used:

- **bold** agents never pause to reconsider intentions, and
- **cautious** agents stop to reconsider after every action.

*Dynamism* in the environment is represented by the *rate of world change*, $\gamma$.

Results (not surprising):

- If $\gamma$ is low (i.e., the environment does not change quickly), then **bold agents**
- If $\gamma$ is high (i.e., the environment changes frequently), then **cautious agents**

Different environment types require different intention reconsideration and commitment strategies.
Reactive agents

- There are many unsolved (some would say insoluble) problems associated with symbolic AI.

- These problems have led some researchers to completely reject symbolic approach and syntactic reasoning on symbolic representations.

- We review the work of one of the most vocal critics of mainstream AI: Rodney Brooks.
Reactive agents

Brooks has put forward three main theses:

- Intelligent behavior can be generated *without explicit representations* of the kind that symbolic AI proposes
- Intelligent behavior can be generated *without explicit abstract reasoning* of the kind that symbolic AI proposes
- Intelligence is an *emergent* property of certain complex systems

Two key ideas:

- **Situatedness and embodiment**: ‘Real’ intelligence is situated in the world, not in disembodied systems such as theorem provers or expert systems
- **Intelligence and emergence**: ‘Intelligent’ behavior arises as a result of an agent’s interaction with its environment. Also, intelligence is ‘in the eye of the beholder’; it is not an innate, isolated property
Reactive agents: Subsumption architecture

- Proposed by Brooks in (Brooks, 86) for robots and extended into a new view of AI (Brooks 91a, 91b)

- A subsumption architecture is a *hierarchy of task-accomplishing behaviors*

- *Behavior:* simple rule-like structure

  \[ \text{situation} \rightarrow \text{action} \]

- Multiple behaviors may fire at the same time
  - Layered hierarchy
  - *Lower layers* behaviors *inhibit higher level* ones (e.g. “avoid obstacles” lower layer than “drive to goal”)

- The resulting systems are, in terms of the amount of computation they do, *extremely simple*

- Some of the robots do tasks that would be impressive if they were accomplished by symbolic AI systems
Limitations of reactive agents

- Agents without environment models must have sufficient information available from local environment.

- If decisions are based on local environment, how does it take into account non-local information (i.e., it has a “short-term” view).

- Difficult to make reactive agents that learn (no memory).

- Building based on test-fault approach which implies a long and laborious process.

- It is hard to engineer agents with large numbers of behaviors (dynamics of interactions become too complex to understand).
Hybrid/cognitive agents

- Hybrid approach: Build an agent out of two (or more) subsystems:
  - a deliberative one, containing a symbolic world model, which develops plans and makes decisions
  - a reactive one, which is capable of reacting to events without complex reasoning
Hybrid Architectures

A key problem in such architectures is what kind of control framework to embed the agent’s subsystems in, to manage the interactions between the various layers:

- **Horizontal layering**
  Behaviors of layers may conflict → Mediator (overall control function) → must consider $m^n$ possible layer interactions → can be bottleneck

- **Vertical layering**
  Control flow: perceptions and action suggestions are passed up (one-pass) or up and then down (two pass) → not fault tolerant: failure in one layer implies failure of whole agent

$m$ possible actions suggested by each layer, $n$ layers

(a) Horizontal layering  
(b) Vertical layering (One pass control)  
(c) Vertical layering (Two pass control)
The TouringMachines architecture consists of *perception* and *action* subsystems, which interface directly with the agent’s environment, and three *control layers*, embedded in a *control framework*, which mediates between the layers.
The **reactive layer** is implemented as a set of situation-action rules, *a la* subsumption architecture

**rule-1: kerb-avoidance**

```plaintext
if is-in-front(Kerb, Observer) and speed(Observer) > 0 and separation(Kerb, Observer) < KerbThreshold then change-orientation(KerbAvoidanceAngle)
```

The **planning layer** constructs plans and selects actions to execute in order to achieve the agent’s goals
The *modeling layer* contains symbolic representations of the ‘cognitive state’ of other entities in the agent’s environment. Selects new goals for the planning layer.

**Control subsystem:** exceeds control (e.g. by suppressing information input to certain layers ("censorship") e.g. in order to prevent the triggering of certain actions.

```plaintext
censor-rule-1:
if
    entity(obstacle-6) in perception-buffer
then
    remove-sensory-record(layer-R, entity(obstacle-6))
```
Deciding the architecture: basic principles

- **Deliberative architectures** are adequate when *long-term planning* and *reasoning* is essential.

- **Reactive architectures** are adequate for agents situated in *highly dynamic environments* where *quick answers* are essential.

- **Hybrid architectures** are adequate for agents situated in *unknown and changing environments*. 
Multiagent Systems’ architecture

- Agents in a multiagent system tend to interact through a middleware layer.

- This middleware provides connectivity between agents, solving low-level connectivity issues:
  - Communication methods

- Sometimes this middleware is called *agent platform*.
Communication methods

- **Blackboard systems**
  - Agents communicate information through a common data structure, accessible by everybody
  - Problem: if there is no middleware to provide some concurrency, it tends to become a bottleneck.

- **Message passing**
  - Agents communicate directly by means of messages
  - The agent platform usually acts as message router
  - Common communication language (e.g. FIPA-ACL)
  - Common communication protocols (message format, steps in a communication)
FIPA Architecture for Agent Platforms

- **Agent Platform (AP):** physical infrastructure in which agents are developed which may be distributed among different computers.

- **Agent:** a program providing a list of services.

- **Directory Facilitator (DF):** an agent which provides a Yellow Pages service within the platform (knows the services that agents within the platform provide).

- **Agent Management System (AMS):** an agent controlling access and usage of the agent platform. It knows the platform and agents’ “addresses” and provides a White Pages service (knows the routing addresses for agents within and in other platforms).

- **Message Transport System (MTS):** is used to enable communication between agents in different platforms.
### Agent architectures: main features

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Main features</th>
</tr>
</thead>
</table>
| Deliberative Agents | - Centralized elements (planner)  
- Internal representation of the environment  
- Outline: *perception* → *symbolic_manipulation* → *response*  
- Not only agent + environment: Deliberative architectures can be used to coordinate behaviors  
- Complex agents                                                                 |
| Reactive Agents     | - No centralized elements  
- No internal representation of the environment  
- Outline: *perception* → *response*, with predefined response patterns  
- No memory  
- Very simple agents  
- Very simple interactions with other agents                                                                 |
| Hybrid Agents       | - Layered architecture  
- Reactive approach for low level activities and deliberative approach to generate complex behaviors |
### Agent architectures: comparison

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliberative Agents</td>
<td>- Familiar technologies&lt;br&gt;- Clear methodologies&lt;br&gt;- Several adequate theories&lt;br&gt;- Facilities to build cooperative systems with multiple autonomous agents&lt;br&gt;- Learning capability can be easily included&lt;br&gt;- Support the development of really autonomous agents</td>
<td>- Problems to update knowledge on time, mainly in highly dynamic environments with not available or limited resources&lt;br&gt;- Problems to translate environment information to a symbolic representation</td>
</tr>
<tr>
<td>Reactive Agents</td>
<td>- Good performance in highly dynamic environments&lt;br&gt;- Simplicity&lt;br&gt;- Robustness and fault tolerance&lt;br&gt;- Efficiency, very quick reactions to events&lt;br&gt;- Limited autonomy</td>
<td>- No consensus on the technology&lt;br&gt;- No methodology, only some isolated theories&lt;br&gt;- A lot of local information necessary&lt;br&gt;- Learning very difficult&lt;br&gt;- Development based on test-fault approach&lt;br&gt;- Agents can only be used for their original purpose&lt;br&gt;- Very simple interactions between agents&lt;br&gt;- Development of complex systems is extremely complicated</td>
</tr>
<tr>
<td>Hybrid Agents</td>
<td>- Combine the advantages of deliberative and reactive architectures</td>
<td>- Pragmatic but <em>ad-hoc</em> solution&lt;br&gt;- No clear consensus on the technology&lt;br&gt;- No methodology nor supporting formal theory&lt;br&gt;- Very difficult to generalize&lt;br&gt;- Very difficult to reproduce results in different domains</td>
</tr>
</tbody>
</table>
## Agent architectures: examples and applications

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Examples</th>
<th>Applications</th>
</tr>
</thead>
</table>
| Deliberative agents | Agentes BDI [Rao, Georgeff, 95]  
                      | GRATE [Jennings, 92]  
                      | IRMA [Bratman et al., 88]  
                      | ADEPT [Alty, 94]  
                      | PRS [Georgeff & Lansky, 86]  
                      | MECCA [Steiner et al, 95]  
                      | Agent0 [Shoham, 92, Shoham, 93]  
                      | SOAR [Rosembloom et. al., 91] | • Software agents  
                      | • Control of complex dynamic processes  
                      | • Information search  
                      | • Simulation |
| Reactive agents | Subsumpción [Brooks, 91]  
                      | Pengi [Agre, 87].  
                      | Situated Automata [Kaebling & Rosenschein, 91] | • Hardware and autonomous agents (robots)  
                      | • Games |
| Hybrid agents   | InteRRaP [Müller, 93]  
                      | Touring Machines [Ferguson, 92]  
                      | 3T [Bonasso, 96]  
                      | ARTIS [Botti, 99]  
                      | COSY [Burmeister & Sundermeyer, 92]  
                      | RAP [Firby, 87]  
                      | SIM_Agent [Sloman, 96]  
                      | dMARS[d’Inverno, 97]  
                      | AuRA [Arkin, 97] | • Robots  
                      | • Control systems  
                      | • Simulation  
                      | • Transport management  
                      | • Planning |