Multi-Agent Systems

Agents' Architectures

Olivier Boissier
Olivier.Boissier@emse.fr
SMA/SIMMO
ENS Mines Saint-Etienne
18 Octobre 2001

Outline

✓ Definitions
  ■ Analysis grid
  ■ Autonomous Agents
  ■ Communicative Agents
  ■ Social Agents
  ■ Agent's Architecture

Agent: external definition (1)

☐ A real or virtual entity, that evolves in an environment, that is able to perceive this environment, to act on this environment, to interact with other agents, and that exhibits an autonomous behaviour. [Demazeau 96]

☐ An intelligent Agent is a computer system capable of flexible and autonomous action in some environment. Flexible means: reactive, pro-active, social. [Wooldridge 98]

Agent: external definition (2)

☐ Reactivity: A reactive system is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it (in time).

☐ Proactiveness: A proactive system is one that generates and attempts to achieve goals; not driven solely by events; taking the initiative.

☐ Social Ability: A social system is one that is capable of interacting with other agents and perhaps to cooperate with others.
Agent: internal definition

- A real or virtual entity that encompasses some local control in some of its perception, communication, knowledge acquisition, reasoning, decision, execution, action processes [Demazeau 01]

- Our definition: a real or virtual entity that is equipped in different degrees with some capabilities to sense or to act on the environment, to communicate with the other agents, to manage relations with the others. This entity is capable of exerting local control on them.

Agent vs. Object

- An agent, like an object, encapsulates a state (attributes) and a behaviour (methods)

- **BUT:**
  - an agent controls its behaviour, i.e. decides what to do from what it receives or senses (an object controls only its state (access methods));
  - in a MAS, there are several flows of control as in concurrent objects BUT several types of control are possible (reactive, deliberative, social, ...)

Model, Theory, Architecture

- **Agent’s Model**: abstract description of an agent using behaviour and structure vocabulary.

- **Agent’s Architecture**: software or hardware architecture that manages the resources under the control of the agent.

- **Agent’s Theory**: language to specify, verify properties of an Agent’s model (often logical grounds)

Outline

- Definitions
- Analysis grid
  - Autonomous Agents
  - Communicative Agents
  - Social Agents
  - Agent’s Architecture
Reactive Agents [Demazeau 96]

- Organized Agents
  - Auto-organization
- Reproductive Agents
  - Replication mechanisms
- Cooperative Agents
  - Grouping mechanisms
- Coordinated Actions
  - Inhibition/activation architectures «subsumption»
- Stimulus/Response
  - Finite State Machines
  - Situated rules

Deliberative Agents [Demazeau 96]

- Organized Agents
  - Multiple perspectives, social laws
- Negotiating Agents
  - Negotiated conflict resolution
- Intentional Agents
  - Intentions, commitments, partial plans
- Cooperative Agents
  - Mutual representations, tasks allocations
- Communication Modules
  - Communication protocols
- Processus/Ac tors
  - Communication primitives

Other terms and classifications ...

- Autonomous Robots
- Personal Assistants, Desktop agents
- Softbots, Knowbots
- Mobile Agents
- Believable Agents
- Reactive Agents, Cognitive Agents, ...
- Intelligent Agents, Cooperative Agents, ...
- ...

Multi-Agent Systems

A new point of view: Agents
(methods+data)+autonomy+cooperation

Establish a contract for printing 10 in B&W
20 in color at the best price.
Model: Agent

Sources: Artificial Intelligence, Objects, Robotic, ...

- Knowledge
- Beliefs, Desires, Intentions
- Reasoning mechanisms, planning, learning, ...

Model: Organization

- Sources: sociology, social psychology, CSCW
  - Social laws
  - Organizational Structures, Contracts
  - Dependence Networks

Model: Interaction

- Sources: Speech Acts, Conversations
  - Interaction Languages
  - Protocols
  - Conversations

Model: Environment

- Sources: Simulation, Physics, GIS, ...
  - Dynamic of the environment in case of simulations
  - Action, Perception

O. BOISSIER (SMA/ENSM.SE)
Communicative Agent: \langle \alpha, a, e, i \rangle

Autonomous agent that is capable to communicate with other agents.

Social Agent: \langle \alpha, a, e, i, o \rangle

Communicative agent that is capable of managing relations with other agents.

Agent: decision view

Strength of the coupling to the environment
(also to the other agents)

- Reactive
Reactive Agent's Model

- The process cycle of an agent is a closed loop between "execute" and "see" (Stimulus/Response).
- Reaction to the evolution of the environment.
- No explicit representation of the environment, of the other agents, of its skills, ...
- Decisions are done without reference to the past (no history), to the future (no planning).

see

Agent

execute

Reactive agent's model (2)

- Reactive approach arises in opposition to the symbolic reasoning model (AI).
- Several approaches that are based on:
  - Behaviours
    - [Brooks 86], [Steels 89], (robotic)
    - [Drogo 93] (ethology)
  - Interactions
    - [Demazeau 93] (image analysis, cartography, ...)
    - [Bura 91] (games)
  - Situations
    - [Agre 87] (games)
    - [Wavish 90] (design, manufacturing)

Reactive agent's model (3)

Example of control cycle of a reactive agent (implemented as a set of condition/action rules):

condition-action rules

set of percepts

do {
  percepts := see();
  state := interpret_input(percepts);
  rule := match(state, rules);
  execute(rule[action]);
} while (1);

Agent: decision view

- Strength of the coupling to the environment (also to the other agents)
  - Reactive
  - Deliberative
Deliberative agent's model

- The process cycle of an agent introduces a "deliberate" function between "see" and "execute" in order to choose the "right" action.
- Explicit Representation of the environment, of the other agents, of its skills, ...
- History management, ...

![Diagram of agent's model]

Deliberative agent's model (2)

- Example of control cycle of a deliberative agent

```
s : state,
eq : event queue

s := initialize();
do {
    options := option_generator(eq,s);
    selected := deliberate(options, s);
    s := update_state(selected, s);
    execute(s);
eq := see();
} while(1);
```

Deliberative agent's model (3)

- Source: practical reasoning (process of deciding, moment by moment, which action to perform in the furtherance of goals: what goals to achieve, how to achieve it).
- Agent has:
  - a set of current beliefs (information the agent has about its current environment)
  - a set of current desires (possible course of actions available to the agent)
  - a set of current intentions (the agent's current focus)

Deliberative agent's model (4)

- Example of a control cycle of a BDI agent

```
b : beliefs, g : desires, i : intentions, eq : event queue

(b,g,i) := initialize();
repeat
    options := option_generator(eq,b,g,i);
    selected := deliberate(options, b,g,i);
    i := selected ∪ i;
    execute(i);
    eq := see();
    b := update_beliefs(b,eq);
    (g,i) := drop_successful_attitudes(b,g,i);
    (g,i) := drop_impossible_attitudes(b,g,i);
forever
```
Agent: decision view

- Strength of the coupling to the environment (also to the other agents)
  - Reactive
  - Hybride
  - Deliberative

✓ Functional and Decision views are orthogonal

Outline

- Definitions
- Analysis grid
  ✓ Autonomous Agents $\langle r_A, a, e \rangle$
  - Communicative Agents $\langle r_A, a, e, i \rangle$
  - Social Agents $\langle r_A, a, e, i, o \rangle$
  - Agent's Architecture

Subsumption architecture
[Brooks 86]

✓ Agent's decision making is realised through a set of tasks accomplishing behaviours. A behaviour continually takes perceptual inputs and maps it to an action to perform (finite state machines, no symbolic reasoning, no symbolic representation)

✓ Many behaviours can fire simultaneously. In order to choose between them, use of a subsumption hierarchy, with the behaviours arranged into layers. A high layer has priority on lower layers (inhibition)
Subsumption architecture (2)

Each layer can be incrementally added to the existing architecture.

Each layer is a set of modules (FSM) which sends messages to each other without central control. Inputs to modules can be suppressed and Outputs can be inhibited by wires terminating from other modules for a determined time. (subsumption)

Example: Distributed Robots

Problem: robots have to collect samples of precious rock (location of the rock samples is not known in advance) and bring it back to a mothership spacecraft.

- Cooperation without direct communication: indirect communication through the environment.
- Gradient field with a signal generated by the mothership
- Radioactive crumbs that can be picked up, dropped, and detected by passing robots.

Example: Distributed Robots (2)

Two sets of behaviours running in parallel:

- Handling behaviour
  - If I sense a sample and am not carrying one, I pick it up.
  - If I sense the vehicle-platform and am carrying a sample, I drop it.
  - If I carry a sample, I drop 2 crumbs.
  - If I carry no sample and crumbs are detected, I pick up one crumb.

- Movement behaviours organized along a subsumption hierarchy

Example: Distributed Robots (3)

If I sense an obstacle in front, I make a random turn.

If I am not carrying a sample and I sense crumbs, I move towards the highest concentration of crumbs.

If I am in exploration mode, I choose the direction of lowest gradient.

If I am in return mode, I choose the direction of highest gradient.

Choose randomly a direction to move. Move in that direction.
Example: Distributed Robots (4)

- Mode determination between Exploration and Return
  - If I am in exploration mode and I sense no lower concentration than the concentration in the cell on which I am located, I put myself in return mode.
  - If I am in return mode and I am at the vehicle platform I put myself in exploration mode.
  - If I am holding a sample, I am in return mode.

Agent réactif: MANTA [Drogoul 93]

- Domaine d’application: simulation de sociétés de fourmis afin d’étudier l’émergence d’une répartition de travail au sein de la société
- Chaque fourmi a un ensemble de tâches
- Tâche:
  - séquence d’actions primitives,
  - poids,
  - niveau d’activation.

MANTA: architecture

\[
W_i(t+1) = W_i(t)
\]

Primitives
- Tâche 1
- Tâche 2
- Tâche 3
- Tâche 4

Niveau d’activité d’une tâche
\[
A_i(t) = (W_i(t)/\Sigma W_j(t))V_i(x)
\]

Renforcement
\[
W_i(t+1) = W_i(t) + \text{delta}
\]

Choix

PRS [Georgeff 83]

- BDI dates back to [Bratman 88] (IRMA)
- PRS (Procedural Reasoning System (Georgeff and Lansky 1987)) uses and supports BDI
- Operational system: dMARS
- Applications: Space Shuttle (Diagnosis), Sydney Airport (air traffic control).

http://www.csc.liv.ac.uk/~mjw/pubs/rara/resources.html
**PRS (1)**

- Agent
- Monitor
- Data Base (Beliefs)
- KAS (Plans)
- Interpreter (Reasoner)
- Goals (Desires)
- PQueue (Intentions)
- Command Generator
- Effectors

**PRS (2)**

- The plan-recipes library (KAS) builds the procedural knowledge to satisfy the intentions.
- A plan-recipe (KA) is defined by:
  - a body
  - triggering condition to activate a plan (Desire)
  - a pre-condition (feasability)

**PRS (3)**

- Nouveaux Buts et Faits
- Unification of
- Dissectionnement
- Actions Primitives
- Exécution de l'Action
- Graph de l'Action
- Bibliothèque de Procédures
- Exécution de la Procédure
- Procédure choisie

**Touring Machine [Ferguson 94]**

- Constrained navigation in dynamic environments (populated with other agents for example)
- Consists of three activity producing layers: each layer produces suggestions for what actions the agent should perform.
Touring Machine (2)

- **Reactive layer**: reactive behaviour
  - Rule 1: *Avoid curb*
    - if $\text{is\_in\_front}(\text{curb}; \text{observer})$ and $\text{speed}(\text{observer}) > 0$ and $\text{separation}(\text{curb}; \text{observer}) < \text{curb threshold}$
    - then $\text{change\_orientation}(\text{curb avoidance angle})$
- **Planning-Layer**: Pro-active behaviour
- **Modeling Layer**: updating of the world, beliefs, predicts conflicts between agents, changes planning-goals
- **Control-subsystem**: Decides about who is active. Certain observations should never reach certain layers.

Outline

- Definitions
- Analysis grid
- Autonomous Agents $\langle x_A, a, e \rangle$
  - *Communicative Agents* $\langle x_A, a, e, i \rangle$
- Social Agents $\langle x_A, a, e, i, o \rangle$
- Agent ’s Architecture

Touring Machine (3)

Communicative Agents $\langle x_A, a, e, i \rangle$

<table>
<thead>
<tr>
<th>Reactive</th>
<th>Deliberative</th>
<th>Hybride</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ferrand 97] SMAALA</td>
<td>[Shoham 93] AOP</td>
<td>[Müller 95] InteRRaP</td>
</tr>
<tr>
<td>[Demazeau 91] PACO</td>
<td>[Burmeister 93] COSY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Steiner 93] IMAGINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Wittig 92] ARCHON</td>
</tr>
</tbody>
</table>
SMAALA [Ferrand 97]

- Based on the Coordination Patterns Model (PACO) [Demazeau 93]
- Used for solving spatial problems (ex: Decision Aid in Country planning)

SMAALA: example

Agent $i$

$p_i(x_i,y_i)$

$cc_i = \{a_{i-2}, a_{i-1}, a_{i+1}, a_{i+2}\}$

SMAALA Ferrand 97

Index of the environment

Environnement Map Values

Attributes of the environment

Population Other agents

State $X(t)$: $(p(t), A(t))$

Agent

Finite State Machine

Perception Filter: perceived regions selection

Perception field: Perceived positions and influence

Environment Map Values

relational Filter: acquaintances selection

Effect of the environment

Communication Field: Known Agents and influence

Parallel updating

Effect of the other agents

Other agents

State $X(t)$: $(p(t), A(t))$

Agent

Agent
SMAALA : example (2)

**Perception Field**

Agent $i$

$p_i = (x_i, y_i) \quad c_{ci} = \{a_{i-2}, a_{i-1}, a_{i+1}, a_{i+2}\}$

$rp_i = 20$

---

**External filtering**

Sensibility Map $X$ Perception Field = Used Information

---

**External Integration**

<table>
<thead>
<tr>
<th>Available Information</th>
<th>Perception Filter</th>
<th>Perceived Effect</th>
<th>weighting</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**SMAALA (4)**

Environment Map Values

Selection in the neighbourhood of the agent

Other agents

State $X(t): (p(t), A(t))$

Agent

Non Adaptive, Reactive Agent
AOP [Shoham 93]

- AOP: Agent Oriented Programming
- Three main components:
  - A formal language with a syntax and a semantics to describe mental states,
  - An interpreted programming language to program agents
  - Agentification process to convert native applications
- Agent: an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments, (...) What makes any hardware or software component an agent is precisely the fact that one has chosen to analyse and control it in these mental terms. [Shoham 93]

AOP: Agent0 (1)

- Agent specified in terms of:
  - A set of capabilities (things it can do)
  - A set of initial beliefs
  - A set of initial commitments (like intentions in BDI)
  - A set of commitment rules
- Key component, which determines how the agent acts, is the set of commitment rules. Each rule contains:
  - A message condition
  - A mental condition
  - An action

AOP: Agent0 (2)

- If the message condition matches a message the agent has received and the mental condition matches the beliefs of the agent, the rule fires.
- When a rule fires, the agent becomes committed to the action.
- Each action is either:
  - Private: an internal subroutine, or
  - Communicative: a message sent to other agents
AOP : Agent0 (3)

- Messages are constrained to be one of three types:
  - request: perform an action
  - unrequest: refrain from performing an action
  - inform: pass an information
- Request and unrequest messages typically result in modification of agent's commitments.
- Inform messages result in a change to the agent's beliefs.
- The operation of an agent is simply: (1) read all current messages, update beliefs and commitments (2) execute all commitments where capable of action (3) goto 1

AOP : Agent0 (4)

- Initialize
- Belief Update
- Commitments Update
- Execution
- Messages
- Beliefs
- Commitments
- Abilities
- Messages

AOP : Agent0 (5)

- Example:

  \[
  \text{COMMIT}(\text{agent, REQUEST, DO(time, action)}) , \\
  (B, [\text{now, Friend agent}] \text{ AND} \\
  \text{CAN(self, action) AND} \\
  \text{NOT [time, CMT(self, anyaction)]}) \\
  , \\
  \text{self, DO(time, action)}
  \]

  i.e.: if I receive a message from an agent requesting me to do action at time, and I believe that the agent is currently a friend and I can do the action and at time, I am not committed to doing any other action then commit to doing action at time.

InteRRap [Muller 95]

- Objective: to provide an agent's architecture for complex, dynamics problems (scheduling, robotics, ...)
- Based on the BDI Model
- Several evolutions and rewriting
Outline

- Definitions
- Analysis grid
- Autonomous Agents $\langle a,e \rangle$
- Communicative Agents $\langle a,e,i \rangle$
- Social Agents $\langle a,e,i,o \rangle$
- Agent 's Architecture

Social Agents $\langle a,e,i,o \rangle$

<table>
<thead>
<tr>
<th>Reactive</th>
<th>Deliberative</th>
<th>Hybride</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Boissier 93] ASIC</td>
<td></td>
<td>[Alloche 98] STAx</td>
</tr>
<tr>
<td></td>
<td>[Jennings 95] ADEPT</td>
<td></td>
</tr>
</tbody>
</table>

ADEPT [Jennings 95]

- Build on the ARCHON architecture (European project)
- Integration architecture
- Based on the BDI model
- Application : Business Process Reengineering
A DEPT (2)

AGENCY

Communication and negotiation with peers and subsidiaries

Responsile Agent

Service Execution Module (SEM)

Situation Assessment Module (SAM)

Interaction Management Module (IMM)

Communication Module (CM)

Communication with tasks

SUBSIDIARY AGENT

A DEPT (3)

Service Execution Module (SEM)

Situation Assessment Module (SAM)

Interaction Management Module (IMM)

Communication (CM)

Acquaintance Model

Execution Services

Preferences

Negotiation History

Self Model

Activity Schedule

Task Availability

A DEPT : IMM

AM
Agent's Groups
- Topology of the group
- Group's state
- Negotiation protocol
- Interaction history

SM
- Services’ descriptions
- Contracts
- Reservation values
- Cost functions

Evaluation Reasoner

Strategic Reasoner

Tactical Reasoner

Communications

CM

SAM

Emitted offers
- received offers
- strategies
- tactics
- Negotiation states
- service's starting and ending dates

ASIC [Boissier 93]

- Architecture for the study of social and individual control in a MAS
- Application: Computer Vision
**ASIC (2)**

- Decision layer
- Adaptation layer
- Command layer

**ASIC (3)**

- RS. evaluated
- KR. ind
- DS. possibilities
- evaluate
- reason
- decide
- RS. non evaluated
- CS. ind
- DS. chosen
- commit

**STAx [Allouche 98]**

- Temporal reasoning in a MAS: focus on the social reasoning aspects
- Application: industrial supervision

**STAx (2)**

- Mental State
- Task Description
- Environment Description
- Task Manager
- Domain
- Task

- Decide
- Planning

- Data Flows
- Control Flows
**Social Reasoning & Perception**

- Domain task
- External Desc agent: A1, A2, A3 ...
- Reception
- Communication
- Social Reasoning
- Decision
- Planning
- Task Manager

**Architectures**

- **Modules' Organization**
  - Horizontal architecture
  - Modular vertical architecture
  - Layered vertical architecture

- **Control flow**: one / several
- **Data flow**: broadcast, traduction
- **Control structure**: inhibition, hierarchy, ...

---

**Outline**

- Definitions
- Analysis grid
- Autonomous Agents $\langle x_a, a, e \rangle$
- Communicative Agents $\langle x_a, a, e, i \rangle$
- Social Agents $\langle x_a, a, e, i, o \rangle$
- Agent's Architecture

---

**Architectures (2)**

- **Horizontal**:
  - Simple ($n$ behaviours, $n$ layers)
  - Overall behaviour might be inconsistent
  - Interaction between layers: $m^n (m = \# actions per layer)$
  - Control-system is needed

- **Vertical**:
  - Only $m^2 (n - 1)$ interactions between layers
  - Not fault tolerant: If one layer fails, everything brakes down
Conclusion

- Several agents' architectures
- Does it exist one unique generic architecture?
- Several components to design the right architecture given the application?
- What about the MAS platforms: what are the agents' architectures available?
- Which Agent Model? For which application?

Bibliography (2)


Bibliography (3)

Bibliography (4)