MASTER WEB INTELLIGENCE

Multi-Agent Systems

Agent Architecture

Cosmin Carabelea

Cosmin.Carabelea@emse.fr

SMA/SIMMO

ENS Mines Saint-Etienne

September 22nd, 2003

Outline

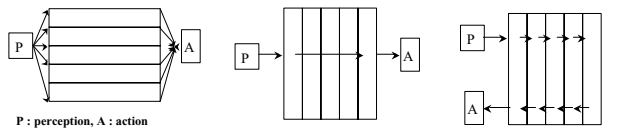
- Introduction
- Individual Agents
- Social Agents
- Organized Agents
- Conclusions

Model, Theory, Architecture

- **Agent model**: abstract description of an agent using behaviour and structure vocabulary.
- **Agent architecture**: software or hardware architecture that manages the resources under the control of the agent.
- **Agent theory**: language to specify and/or verify properties of an agent 's model (often logical grounds)

Agent Architectures

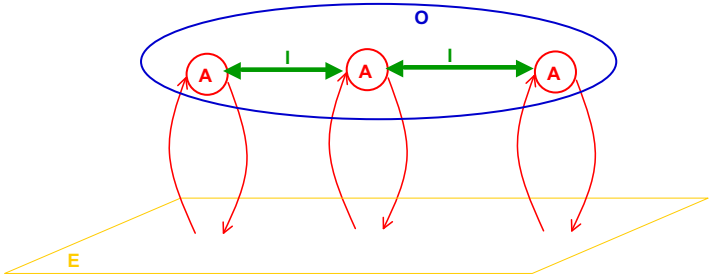
- Modules' Organization:



a) horizontal architecture b) modular vertical architecture one path c) layered vertical architecture two paths

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

Analysis grid



Individual agents reason about the agents in the system and the environment

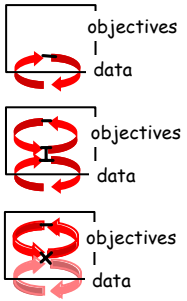
Social agents reason about the agents in the system, the environment and the interactions with other agents

Organized agents reason about the agents in the system, the environment, the interactions with other agents and the organizational structures

Analysis grid (2)

Strength of the coupling with external factors (the environment, other agents, the organization)

- Reactive agents
- Hybrid agents
- Deliberative agents



Outline

- Introduction
- **Individual Agents**
  - Subsumption architecture
  - PRS
  - Touring Machine
- Social Agents
- Organized Agents
- Conclusions

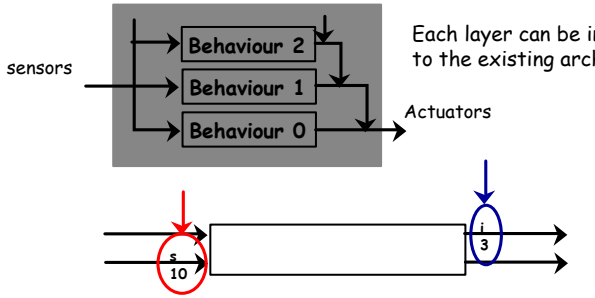
Individual, reactive agents

The subsumption architecture
[Brooks 86]

Subsumption architecture

- Agent's decision making is realised through a set of tasks accomplishing behaviours.
- A behaviour continually takes perceptual inputs and maps them 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)

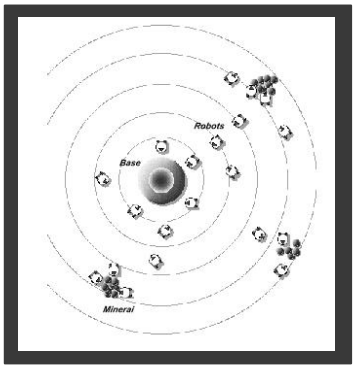
Subsumption architecture (2)



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 [Steels 89]



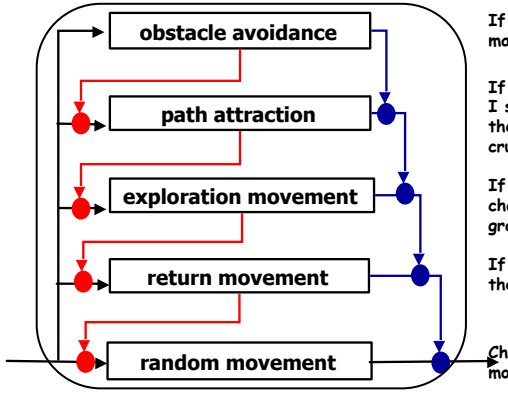
- Problem : robots have to collect samples of precious rock (location of the rock samples is not known in advance) and bring it back to a mothership spacecraft.
- Cooperation without direct communication : indirect communication through the environment.

➔ 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.

Individual, deliberative agents

PRS architecture
[Georgeff, Lansky 87]

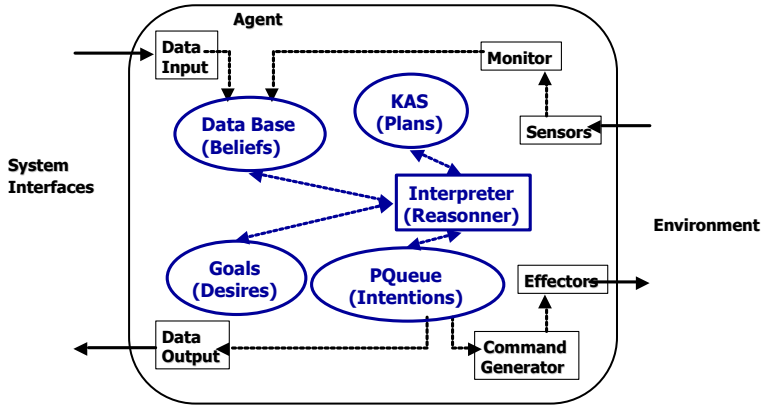
BDI Model

- [Georgeff 83], [Bratman 90] – the use of intentions in agent's design.
- [Rao, Georgeff 91] – the BDI model: an agent contains:
  - a set of beliefs about itself and the world;
  - a set of (possibly conflicting) desires
  - a set of non-conflicting intentions
  - reasoning mechanisms to update its beliefs, choose the desire(s) to pursue and generate new intentions

BDI Implementations

- [Georgeff, Lansky 87]: **P**rocedural **R**easoning **S**ystem uses and supports the BDI model.
- [Rao 95]: BDI-logics – modal operators for Beliefs, Desires and Intentions.
- BDI applications: Space Shuttle (Diagnosis), Sydney Airport (air traffic control).
- BDI Agents Platform: JACK, Zeus.

PRS



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)

Individual, hybrid agents

Touring Machine
[Ferguson 94]

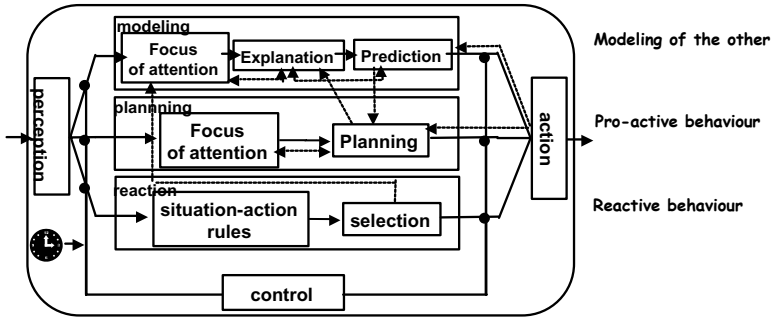
Touring Machine

- 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 *is_in_front*(*curb; observer*) and
    speed(*observer*) > 0 and
    seperation(*curb; observer*) < *curb threshold*
    then *change orientation*(*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.

Touring Machine (3)



Outline

- Introduction
- Individual Agents
- **Social Agents**
  - AOP
  - InteRRaP
- Organized Agents
- Conclusions

Social, deliberative agents

AOP

[Shoham 93]

AOP

- AOP : Agent Oriented Programming
- Three main components :
  - a formal language with a syntax and a semantic 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

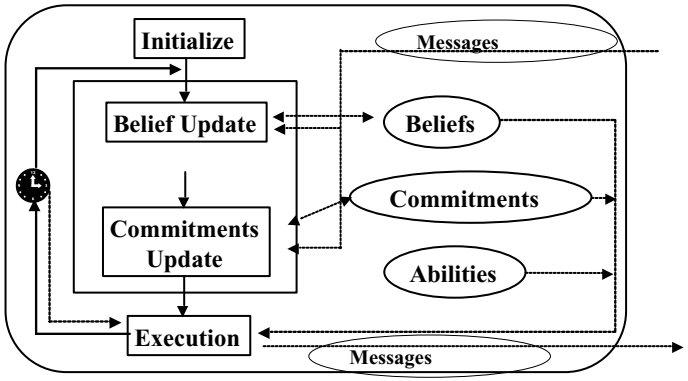
AOP : Agent0 (2)

- 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)



AOP : Agent0 (5)

- Example :

```
COMMIT( (agent, REQUEST, DO(time, action)),
        (B, [now, Friend agent] AND
            CAN(self, action) AND
            NOT [time, CMT(self, anyaction)]
        ),
        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.

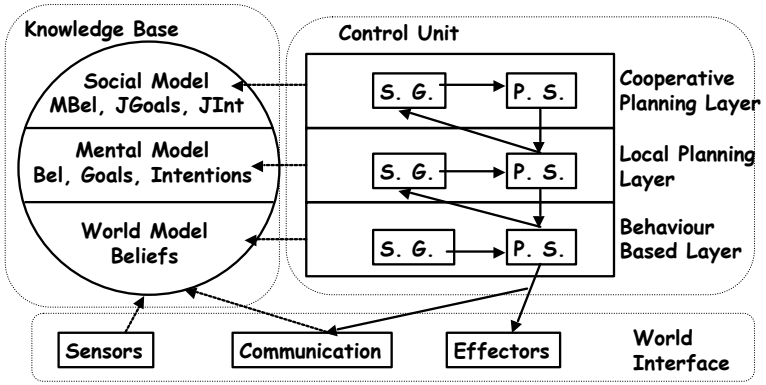
Social, hybrid agents

InteRRaP
[Muller 95]

InteRRap

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

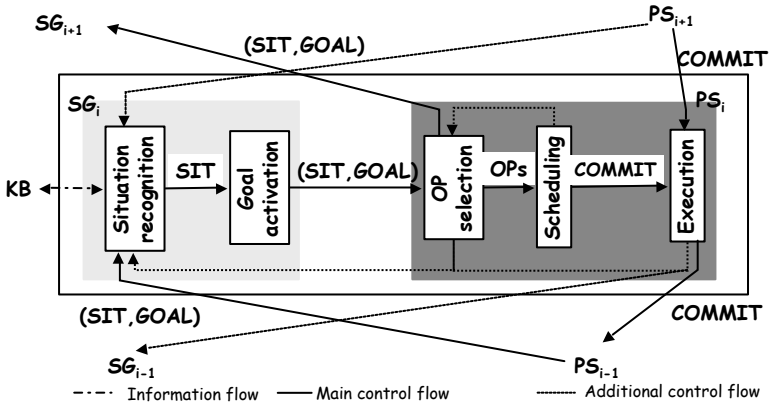
InteRRaP (2)



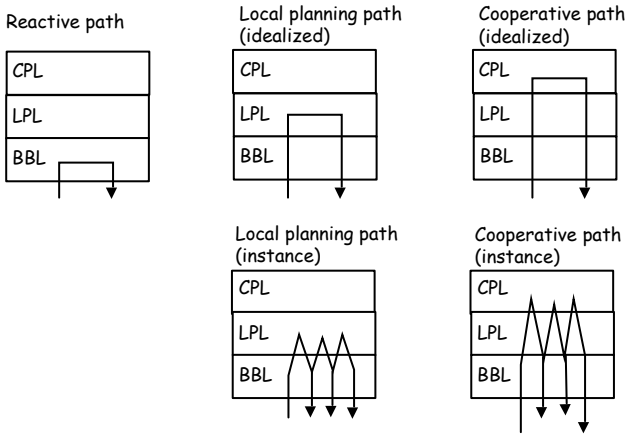
InteRRaP (3)

	BBL	LPL	CPL
Belief Revision	Generation and revision of beliefs (world model)	Abstraction of local beliefs (mental model)	Maintaining models of other agents (social model)
Situation recognition Goal activation	Activation of reactor patterns	Recognition of situations requiring local planning	Recognition of situations requiring cooperative planning
Planning Scheduling	Reactor POB : direct link from situations to action sequences	Modifying local intentions; local planning	Modifying joint intentions ; cooperative planning.

InteRRaP (4)



InteRRaP (5)



Outline

- Introduction
- Individual Agents
- Social Agents
- **Organized Agents**
  - B-DOING
- Conclusions

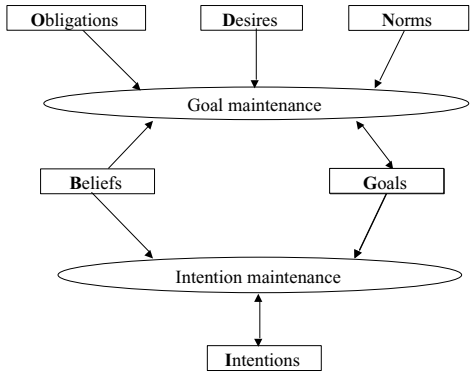
Organized, deliberative agents

B-DOING
[Dignum et al., 2001]

B-DOING

- Extends the BDI model.
- The agent's intentions are generated based on its current beliefs and a set of possibly conflicting goals.
- The goals are generated from:
  - a set of desires: what the agent wants;
  - a set of obligations: what other agents want;
  - a set of norms: what is good for the society.
- B-DOING logic: an extention of BDI-logic.

B-DOING (2)



Outline

- Introduction
- Individual Agents
- Social Agents
- Organized Agents
- **Conclusions**

Conclusion

- Several agent architectures.
- Is there a generic architecture?
- Several components to design the right architecture given the application?
- What about the MAS platforms: what are the agent architectures available?
- Given the application, what agent model?

Bibliography

- M.E. Bratman: *What is Intention?*. Intentions in Communication, MIT Press, 1990, p. 15-32.
- R. Brooks, J.H. Connel: *Asynchronous Distributed Control System for a mobile robot*, SPIE 727 Mobile Robots, 1986.
- Y.Demazeau: *From interactions to collective behaviour in agent-based systems*. In Proc. of the 1st European Conf. on Cognitive Science, Saint Malo, France, April, 1995, p. 117-132.
- F.Dignum, D.Kinny and E.Sonenberg: *Motivational Attitudes of Agents: On Desires Obligations and Norms*. In Proc. of the 2nd Int. Workshop of Central and Eastern Europe on MAS, 2001, p.61-70.
- I.Ferguson: *Integrating Models and Behaviors in Autonomous Agents: Some Lessons Learned on Action Control*. Intelligent Agents - Proc. of the Workshop on Agent Theories, Architectures, and Languages (ATAL'94). LNCS Series, Springer-Verlag, Berlin. 1995.
- M.P. Georgeff: *Communication and Interaction in Multi-Agent Planning*. AAAI, 1983, p. 125-129.

Bibliography (2)

- M.Georgeff, A. Lansky: *Reactive Reasoning and Planning*. In Proc. of the Conf. of the American Association of Artificial Intelligence, Seattle, WA, 1987, p. 677–682.
- J.P.Muller: *The Design of Intelligent Agents: A Layered Approach*. Springer-Verlag, 1996.
- A.S.Rao, M. Georgeff: *Modeling Rational Agents within a BDI-Architecture*. In J. F.Allen, R. Fikes and E. Sandewall (eds.), Proc. of the Int. Conf. on Knowledge Representation and Reasoning, Morgan Kaufmann, Cambridge, MA, 1991, p. 473–484.
- A.S.Rao: *Decision Procedures for Propositional Linear-Time Belief-Desire-Intention Logics*. In M. Wooldridge et al. (eds.), Intelligent Agents II – Proc. of the Workshop on Agent Theories, Architectures and Languages (ATAL-95), LNAI-890, Springer-Verlag, Berlin, 1995, p.1-39.
- Y. Shoham: *Agent Oriented Programming*, Artificial Intelligence 60, 1993, p. 51-92.
- L. Steels: *Cooperation between Distributed Agents through Self-Organization*. Decentralized AI Elsevier North-Holland, 1989.
- M.J.Wooldridge: *Intelligent Agents*, In G.Weiss (ed.): Multi-agent systems: A modern approach to Distributed Artificial Intelligence, MIT Press, 1999.