Multi-Agent Oriented Programming
Introduction to Multi-Agent Systems

Olivier Boissier
Olivier.Boissier@emse.fr
Plan

1. Definitions
2. Action Domains
3. Positioning
4. Multi-Agent Perspective
5. Multi-Agent Engineering
6. Perspectives …
Multi-Agent Systems (MAS)

Definitions
Multi-Agent System (MAS) : set of **agents**, that **interact** with each other, situated in a common **environment**, eventually, building or participating to, an **organisation**
Agent (in a Multi-Agent World)

Agent: physical or software, autonomous entity that is pro-active, reactive, social, able to take part to an organised activity, in order to achieve its goals, by interacting with other agents and users.
Multi-Agent Systems Principles

• The **Agent perspective** (micro perspective)
  • Encapsulation of control
    • Reactivity, [Pro-activity, Social Ability & Organization awareness]
    • Autonomy: agents may exhibit activities that are not the one expected by the other agents in the system
  • Delegation/Adoption: agents may receive some control over their activities

• The **Multi-Agent System perspective** (macro perspective)
  • Distribution
    • knowledge, of resources, of reasoning/decision capabilities
  • Decentralization (loose coupling) of control, authority
  • Coordination
    • Agreement technologies, Coordination models and mechanisms to install coordination between the autonomous agents
  • Regulation
    • Emergent / Social order / Normative functioning / Regimentation
Autonomous Agent (in a Multi-Agent World) Definitions

• An agent $X$ is autonomous with respect to $Y$ for $O$ in situation $S$
  • $Y$ can be a user, another agent, a group of agents, an organisation, …
  • $O$ can be a goal, a plan, an action, a resource, a norm, a role, …

It means that:
• agent $X$ can decide locally of the adoption of $O$ in situation $S$
• And $Y$ has no certainty that $X$ is going to adopt $O$ in situation $S$

$\rightarrow$ Loose coupling between agents

Introduction to multi-agent systems © O. Boissier MiNES Saint-Etienne
Plan

1. Definitions
2. Action Domains
3. Positioning
4. Multi-Agent Perspective
5. Multi-Agent Engineering
6. Perspectives ...
MAS Action domains

- **Socio-technical Systems**
  - Integration of software applications, with humans, organizations and the physical world
  - Making them interoperate, interact, cooperate in a flexible and consistent manner with each other

- **Problem Solving**
  - Modeling and solving problems by cooperation between local solvers
  - Installing top-down and/or bottom-up (emergent) solving process

- **Simulation**
  - Modeling and reproducing complex phenomena of interacting entities in the real world in order to understand or to explain their behavior
Socio-Technical Systems (1)

- Industries, services, IT applications are getting global
  - Placed at the centre of multiple networks
  - Developing Knowledge intensive processes
  - Based on large scale underlying IT platforms such as Internet, Web, Internet of Things

- Industries, services, IT applications are situated in an ever-evolving environment
  - Requiring efficient collaboration processes
  - While keeping flexibility and agility

- Users are more and more at the centre of the cooperation and collaboration taking place in these socio-technical systems
Socio-Technical Systems (2)

Properties of the targeted applications:
- Absence of monolithic vision
- Incremental development, by different teams
- Multi-* (sites, expertise, domains, points of view, decisions, goals, motivations, …)
- Continuous execution and adaptation
- User-Centred

Main requirements:
- Openness, permeability, scalability in size or structure
- Distribution, no central control, control and interaction are local
- Autonomous Interacting entities loosely coupled with others or applications
- Knowledge Intensive processing and sharing
- Users may delegate their decisions to the application
Example (1/3)

Service Personnalisation

Tonight’s Suggested Viewing:

7pm World News Headlines
7:15 Personal Newsround
7:30 Selected highlights of today’s golf
7:50-8:00 Intermission (Video-call - it’s your brother’s birthday)
8:00-10:00 Film choice
   Jurassic Park (VR) OR Cyberspace 2
   (please select now)

Source CLIMATE Industrial Workshop 26/4/99
Example (1/3)

Tonight’s Suggested Viewing:
7pm World News Headlines
7:15 Personal Newsround
7:30 Selected highlights of today’s golf
7:50-8:00 Intermission (Video call - it’s your brother’s birthday)
8:00-10:00 Film choice
Jurassic Park (VR) OR Cyberspace 2
(please select now)

Source CLIMATE Industrial Workshop 26/4/99
Example (2/3)

User-Centred Service Coordination

(1) Emergency Assistance
(2) Telemonitoring & e-Inclusion
(3) Shopping Mall

Source: CASCOM
FP6-IST-2
Example (2/3)

Service Composition
Semantic Services
Description
Secured Execution and monitoring of services

Semantic Web
Agent Technology

Source: CASCOM
FP6-IST-2
Example (3/3)

Adaptation & optimisation

Planification, coordination, optimisation along a bottom-up approach:
- Responsibility Delegation
- Communication between the nodes
- Real time detection & reaction to changes
- Adaptation to changes & continuous optimisation

Planification, coordination, optimisation along a top-down approach:
- Centralised collect and processing of informations and events
- Propagation of plans & decisions
- No realtime decision.

Source Whitestein Agent Technology Conference 2004
MAS Action domains

- **Socio-technical Systems**
  - Integration of software applications, with humans, organizations and the physical world
  - Making them interoperate, interact, cooperate in a flexible and consistent manner with each other

- **Problem Solving**
  - Modeling and solving problems by cooperation between local solvers
  - Installing top-down and/or bottom-up (emergent) solving process

- **Simulation**
  - Modeling and reproducing complex phenomena of interacting entities in the real world in order to understand or to explain their behavior
Problem Solving

Properties of the targeted applications:
- Absence of global strategies, of global solving method
- Interaction between local strategies, between local solving methods
- Solution is the result of the interaction between local processes (points of view, decisions, goals, motivations, …)
- Continuous functioning and evolution

Main requirements:
- Decentralisation, local control, interactions
- Openness, permeability, scalability in size or structure
- Shared and dynamic environment
- Emergence of the solution
Example (1/2)

Action domains

Ferrand 97
Example (1/2)

Action domains

Ferrand 97
Example (2/2)

Design of Complex Systems

- Multi-Disciplinary Simulation & optimisation (ID4CS)
- Design of complex system:
  - Multi-level, Multi-disciplinary
  - Multi-methods
  - Multi-objectives, Multi-attributes
  - Uncertainty
- Cooperation methods between optimisation technics,
- Management of uncertainty
- Multi-* problem solving
- Emergence
MAS Action domains

• Socio-technical Systems
  • Integration of software applications, with humans, organizations and the physical world
  • Making them interoperate, interact, cooperate in a flexible and consistent manner with each other

• Problem Solving
  • Modeling and solving problems by cooperation between local solvers
  • Installing top-down and/or bottom-up (emergent) solving process

• Simulation
  • Modeling and reproducing complex phenomena of interacting entities in the real world in order to understand or to explain their behavior
Example (1/2)

In order to:

Understand, Explain
Discover, ..., Help,
Example (1/2)

**Action domains**

MANTA [Drogoul 93]

**Introduction to multi-agent systems**

© O. Boissier MINES Saint-Etienne
Example (2/2)

http://www.massivesoftware.com/

The Return of the King (2003)
The Two Towers (2002)
The Fellowship of the Ring (2001)

..., Entertainment

Action domains

Ratatouille (2007)

Conversational Zeno Robot
http://hansonrobotics.com/
Plan

1. Contexte
2. Definitions
3. Action Domains
4. Positioning
5. Multi-Agent Perspective
6. Multi-Agent Engineering
7. Perspectives ...
History – Major Steps

• **1980**: Agents in the Artificial Intelligence (AI) area
  - From AI to *Distributed* AI … to Multi-Agent Systems

• **1990**: Agents are invading other domains
  - Personal Assistants, avatars,
  - Mobile Agents,
  - Reactive Agents, ….

• **1995**: Agents spread in other domains, Application domains are enlarging
  - Artificial Life, Economic Agents, …,
  - Web, Ambient Computing, Internet of Things, …
History – Evolutions

• 1973 - 1980:
  • Hearsay II (1973): blackboard architecture for speech recognition
  • Actor Languages (1973): messages as control structures
  • Beings (1975), Society of Minds (1978)

• 1980 - 1990:
  • Contract Net (1980): hierarchical decentralized control
  • DVMT (1984): Distributed Interpretation
  • Subsumption architecture (1986): Reactive Robots
  • MACE (1987): multi-agent platforms

• 1990 - … :
  • Self-organisation, emergence, Interactions, organisations, reputation, trust, Agent Oriented Software Engineering, …
  • In 1995, first international conference ICMAS,
  • since 2002, Autonomous Agents + MAS -> AAMAS
Inter-Disciplinary Domain…

• Direct Links with:
  • Programming, Object Oriented Programming ...
  • Artificial Intelligence
  • Distributed Systems, Parallelism, Concurrent Programming

• But also:
  • Complex System (physics, …, ethology, ecology, …)
  • Artificial Life, Neural networks, …
  • Social Psychology, Sociology, Activity Theory, Economy, …
Direct Inheritance

• Object Oriented Programming:
  • Encapsulation, modularity: an object encapsulate data and methods that manage them (ex: C++, Java, Smalltalk),
  • Distribution: Distributed objects, CORBA, DCOM
  • → Actor Languages Development

• Artificial Intelligence:
  • Symbolic Reasoning Models (Expert systems, Knowledge Representation), logic, …
  • distribution: Blackboard Architectures

• Distributed Systems
Multi-Agent vs Objects

• An agent, as an object, encapsulates a state and behaviors

BUT:

• An agent encapsulates its control over its behaviors; an object has only control over its state
• Interactions among agents have a broader scope than the method calls between objects. Interactions consist in goals, plans, actions, hypothesis exchanges
• An agent may have different control cycles (data-directed, goal-directed, interaction-directed, …)
• A MAS has several control flows. An Object system has, a priori, only one control flow.
Mono-agent perspective of Artificial Intelligence is pushed away

Knowledge, Goals, Actions gain a social dimension
Ex. dependence networks
Multi-Agent \textbf{vs} Artificial Intelligence (3)

Social Knowledge

Ag3 external description:
\textbf{goals}: on(A,B), \textbf{actions}: clear
\textbf{resources}: A, \textbf{plans}: on(A,B):=clear(C), put_on(A,B)

Ag1 external description:
\textbf{goals}: on(C,Table), \textbf{actions}: put_on
\textbf{resources}: B \textbf{plans}: on(C,Table):=clear(C)

\ldots

action dép. put_on
Social Interaction

Hello Ag1, I need your action “put_on”
To set A on B, just do it!!

Command(Ag3, Ag1, put_on(A,B))
Multi-Agent vs Distributed Systems

• Both take into account interconnection and distribution

• In MAS, Interconnection and Distribution are concerned by:
  • The requirement of taking into account the agent autonomy, of developing synchronization and coordination mechanisms to coordinate their activities
  • The requirement to represent and take into account the user interests
  • The requirement to cooperate and to achieve agreements (or even compete) with other systems aiming at achieving their own interests.
A Large Domain!!

From *Autonomous Agents* to *Multi-Agent Systems*

- Autonomous Robots
- Personal Assistants
- Desktop Agents
- Softbots, Knowbots
- Mobile Agents
- Reactive Agents
- Intelligent Agents, Cooperative Agents, Conversational Agents
- Autonomous Agent in a multi-agent world
Plan

1. Definitions
2. Action Domains
3. Positioning
4. **Multi-Agent Perspective**
5. Multi-Agent Engineering
6. Perspectives …
Multi-Agent Perspective

• Several multi-agent models and concepts exist in the literature!

• One possible structuration of these models and concepts:
  • Along four dimensions ("Vowels" [Demazeau 95]):
    • Agent, Environment, Interaction, Organisation
  
    • Taking two main points of view:
      • global (System centred), local (Agent centred)
Agent Model

- To define the autonomous entities in the system (internal architectures, knowledge representation, reasoning mechanisms…)
  - BDI Agents (Belief Desire Intention) [Rao 95]
  - Rational Agents [Russel 91]
  - Situated Agents [Agre 87], [Maes 90]
- Sources: Artificial Intelligence, Objects, Robotics, ...

Global point of view
Environment Model

- To define the common space shared among the agents
  - Required explicit abstraction so that agents can be part of a MAS (execution space, coordination space, information space, etc)
  - Active entity that mediates the interaction among agents and provides access to resources

- Sources: Simulation, Physics, …
Reference Model [Weyns 06]

- Perception & Action Modalities,
- Resources, Coordination Artifacts
- Topology
- Proper Dynamic
CARTAGOGO [Ricci 07]

Global point of view

http://apice.unibo.it/xwiki/bin/view/CARTAGO/

Introduction to multi-agent systems

© O. Boissier MINES Saint-Etienne
Interaction Model

- To define and structure the **dynamic relation** between two or several agents through **reciprocal actions** that may draw patterns of activity (**conversations**)
  - Agent Communication Languages (ACL FIPA, KQML, …), Content Languages, Ontologies
  - Interaction Protocols, Conversations, …
- **Sources**: Speech Acts, Conversations

---

**Agent Architectures**

**MAS Services**

**Services Interaction**

**Environment Simulation/Services**

**Middleware**

---

Propose 10 NB

CFP 20 Couleur
Agent Communication Language

- High level language to share propositional attitudes, to collaborate, Negotiate, ...
  - FIPA-ACL [FIPA 02], KQML [Finin 97], etc.

- Set of performatives based on mental states
  - Inform, request, cfp, agree, understood

- Content Languages
  - Ex: KIF, FIPA-SL, FIPA-CCL, etc.
  - To express actions, objects, propositions
  - Based on ontologies, i.e. common vocabularies relative to a domain (weather-ontology, cinema-ontology, etc)
ACL FIPA Message Example

(inform communication act
  :sender A used for routing the message
  :receiver B used for routing the message
  :content (price (bid good02) 150) content
  :in-reply-to round-4
  :reply-with bid04
  :encoding 1000
  :language fipa-sl1 content language
  :ontology hpl-auction
  :reply-by 10 deadline for answering
  :protocol offer interaction protocol
  :conversation-id conv02 conversation id
)

Global point of view
Interaction Protocol Example

Global point of view
Organization Model

• To define the coordination pattern for achieving a given goal and To make these coordination patterns accessible to the agents
  • Organizational Structures: roles, groups, plans, scenes
• To control and regulate the autonomous activities of the agents
  • RBAC, Norms, Obligations, Permissions, Laws, …
• Sources: sociology, social psychology, CSCW,
Declarative definition of the agents organization expressing coordination patterns to achieve a common global goal
Ex: AGR [Ferber 98], Teamwork [Tambe 98], Islander [Esteva 01], Moise+ [Hubner 02], ...
Management of the organizational entity

- Within the agents ex: Jason-MOISE [Hubner 07],
- Within components accessible to the agents ex: Madkit [Gutknecht 00], Karma [Pynadath 03], Ameli [Esteva 04], S-Moise+ [Hubner 05], SYNAI [Gateau 06], …

http://moise.sourceforge.net
Introduction to multi-agent systems

Synthesis

- Common space shared among the agents (Signals, Influences, Dynamics, …)
- Autonomous entities (Internal Architectures, Knowledge, …)
- Interactions among the agents (ACL, Interaction Protocols, …)
- Coordination Patterns (Organization Modeling Languages, Norms, …)
- Environment
- Agent
- Interaction
- Organisation

Global point of view
Agent Model

**Agent**: software or hardware encapsulating processing mechanisms and data, which is able to control its decisions and actions (internal & external), to perceive and act on the environment, to interact with the other agents and to manage and reason on the relations with other agents and norms.

- **Situated Agents**
  - agents that reason on themselves and on the environment

- **Social Agents**
  - situated agents that reason also on the other agents

- **Organization Aware Agents**
  - social agents that also reason on the organizations/norms in which they are involved
Agent types wrt Control Architecture

Local point of view

• Strength of the coupling of the decision mechanism with external events (environment, interactions, organisation/normes)
  
  • **Reactive agent**
Reactive Agent Model

- Control Cycle: closed loop involving two operations "execute" and "see" (Stimulus/Response)
- reaction to the evolutions of the environment
- No explicit representation of the environment, of the other agents, of its skills, ...
- Decision taken using neither some past history nor future (no planning)
Reactive Agent Model (2)

- Reactive approaches (developed in opposition to AI symbolic reasoning)
- Different approaches based on:
  - behaviours
    - [Brooks 86], [Steels 89], (robotic)
    - [Drogoul 93] (ethology)
  - interactions
    - [Demazeau 93] (image analysis, cartography, …)
    - [Bura 91] (games)
  - situations
    - [Agre 87] (games)
    - [Wavish 90] (design, manufacturing)
Control Cycle Example (implemented using production rules)

\[\text{condition-action rules}\]
\[\text{set of percepts}\]

do {
    \text{percepts} := \text{see}();
    \text{state} := \text{interpret_input} (\text{percepts});
    \text{rule} := \text{match} (\text{state}, \text{rules});
    \text{execute} (\text{rule}[\text{action}]);
} \text{while} (1);
Agent types wrt Control Architecture

- Strength of the coupling of the decision mechanism with external events (environment, interactions, organisation/normes)
  - Reactive agent
  - Deliberative agent
Deliberative Agent Model

1. Introduction of “deliberate” function in the control cycle between the "see" and "execute" functions in order to reason and choose the proper action.

2. Explicit Representation of the environment, of the other agents, of its skills, ...

3. History management, planning.

Local point of view

Diagram:
- Agent
- Deliberate
- See
- Execute
- State
- Environment

© O. Boissier MINES Saint-Etienne
Deliberative Agent Model (2)

Local point of view

• Example of control cycle

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 Model (3)

- **BDI Control Cycle**

  - $b$: beliefs, $g$: desires, $i$: intentions, $eq$: event queue

  $$(b,g,i) := initialize();$$

  **repeat**

  - $options := option_generator(eq,b,g,i);$  
  - $selected := deliberate(options, b,g,i);$  
  - $i := selected \cup i;$  
  - $execute(i);$  
  - $eq := see();$  
  - $b := update_beliefs(b,eq);$  
  - $(g,i) := drop\_successful\_attitudes(b,g,i);$  
  - $(g,i) := drop\_impossible\_attitudes(b,g,i);$  

  **forever**
Agent types wrt Control Architecture

- Strength of the coupling of the decision mechanism with external events (environment, interactions, organisation/normes)
  - Reactive agent
  - Hybrid agent
  - Deliberative agent
Situated Agent

- Action,
- Perception (ex: access to DataBase, internet, …, physical world, virtual world)
Social Agent

- Interpretation of ACL, of interaction protocols
- Reasoning on interactions: interaction strategies
- Conversation management

Local point of view
Organization Aware Agent

- Reasoning on norms, on organizations [Carabelea 04]
- Contracts, Dependence Networks [Sichman 94]
- Reputation, Trust [Muller 06]
Introduction to multi-agent systems

© O. Boissier MiNES Saint-Etienne

Synthesis

Skills
Motivations
Reasoning

Knowledge
Reasoning
Organisation

Perception
Beliefs
Desires, Goals
Intentions
Interaction

ACL interpretation
Interaction protocols
Conversation
Management

Event Perception Action on the environnement
Representation & Mngt of Organisations
Representation & Mngt of Norms,
Social Reasoning

Local point of view
Introduction à la mécanique des systèmes multi-agents

**Synthèse**

- **Évaluation**
  - RS évalué
  - KR ind.
  - DS possibilités

- **Raisonnement**
  - évaluation
  - KR soc.
  - CS ind.
  - DS choix

- **Décision**
  - raisonement
  - KR soc.
  - engagement
  - KR soc.

- **Engagement**
  - évaluation
  - CS soc.
  - KR soc.

- **Interprétation**
  - Interprétation
  - Dialogue
  - Interprétation

- **Exécution**
  - Exécution
  - Dialogue
  - Exécution

**Mécanismes de traitement**

- PS
  - capteur
  - récepteur

- CS soc.
  - KR soc.

- Org.

- ES
  - émetteur
  - effecteur

**Représentation**

- ACL, Pr

- Flux de données

**Connaissances**
Introduction to multi-agent systems

© O. Boissier MINES Saint-Etienne
Synthesis

Compiled from AgentCore.java

public abstract class mast.core.AgentCore extends java.lang.Thread implements java.io.Serializable {

    public boolean RUNNING;
    java.io.File log;
    java.lang.StringBuffer logString;
    long timeout;
    java.lang.String name;
    mast.facet.Facet[] facet[];
    java.util.Hashtable expectingAnswer;
    java.util.Hashtable eventQueue;
    java.util.Hashtable facetParams[];
    public mast.core.Facet core()
    public mast.core.AgentCore() {
    public mast.core.AgentCore(java.lang.String);
    public mast.core.AgentCore(java.lang.String,long);
    public mast.core.AgentCore(java.lang.ThreadGroup,java.lang.String,long);
    public mast.core.AgentCore(java.lang.String);
    public mast.core.AgentCore();
    }
    public mast.core.AgentCore(java.lang.ThreadGroup,java.lang.String,long).throws
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.AgentCore();
    public mast.core.Ag
Introduction to multi-agent systems

Synthesis

Skills
Motivations
Reasoning

Representation & Mngt of Organisations
Representation & Mngt of Norms,
Social Reasoning

Knowledge
Reasoning
Organisation

ACL interpretation
Interaction protocols
Conversation Management

Interaction

Perception
Beliefs Desires, Goals Intentions
Action

Event Perception
Action on the environnement
Representation of the environment

© O. Boissier MINES Saint-Etienne
Plan

1. Definitions
2. Action Domains
3. Positioning
4. Multi-Agent Perspective
5. Multi-Agent Engineering
6. Perspectives …
Multi-Agent Engineering

• Developing multi-agent applications is often a difficult task
  • implementation, distribution, communications, …

• There exists
  • Multiple technologies focused on particular points of a MAS
  • Multiple agent programming languages, dedicated or general purpose based on existing programming languages
  • Multiple multi-agent programming platforms, involving often a specific agent architecture, proposing or not, first order abstractions for the environment, the organisation, the interaction.
  • Multiple standards
  • Multiple software engineering methods for the analysis and design of MAS.
  ➔ Multiple languages, platforms, methods are available …
  ➔ But often limited to a very focused set of applicative domains.

  ➔ Which one to choose? How to choose? How to compare?
Multi-Agent Technologies

- Agent Architectures and Theories
- Coalition formation mechanisms
- Multi-Agent Planning
- Agent Communication Languages, Interaction Protocols
- Auction mechanisms
- Negotiation strategies and mechanisms, Argumentation
- Electronic Institutions, Organisations, Norms
- Reputation, Trust
- Mono & multi-agent Learning
- Self-organisation, emergence, …
Declarative Approach

Languages

- CLAIM (Computational Language for Autonomous Intelligent and Mobile Agents)
  - Cognitive Agent Programming Language
  - Belonging to Himalaya Framework (Hierarchical Intelligent Mobile Agents for building Large-scale and Adaptive sYstems based on Ambients)
  - Based on process algebra in order to represent concurrency and agent mobility
  - Based on SyMPA platform implemented in JAVA respecting the MASIF standard

- FLUX:
  - Cognitive Agent Programming Language
  - Fluent Calculus implementation (Action representation formalism)
  - http://www.fluxagent.org
Imperative Approach

- JACK Agent Language (JAL)
  - Developed by Agent Oriented Software
  - Based on PRS, BDI model (similar to hybrid languages such as Jason, 3APL, Jadex)
  - JAL is an extension of Java allowing to create plans, beliefs base, …
  - Possibility to use team of agents, organisation of agents

- http://www.agent-software.com
Hybrid Approach

- **3APL (An Abstract Agent Programming Language « triple-a-p-l »)**
  - Programming language for the development of cognitive agents:
    - By defining structures for beliefs, goals, plans, actions (internal, external or communication) and reasoning rules (modification of plan bases),
    - By reasoning methods to generate plans, revise plans, to achieve goals
    - Integration of Prolog and Java
    - [http://www.cs.uu.nl/3apl](http://www.cs.uu.nl/3apl)

- **Jason**: extended version of AgentSpeak(L) interpreter, agent oriented programming language based on logic. Introduced by Rao.
  - Communication between agents based on Speech-act (beliefs and goals annotated by the information sources)
  - Plans annotations
  - Functions for selecting, for trust computation,
  - Functions and agent architecture may be adapted (perception, belief-revision, inter-agent communication, acting)
  - Integration by the user of existing code by the way of internal
  - Implemented in java, bound to the organisational language MOISE, interfaced with the CARTAGO platform
  - [http://jason.sourceforge.net](http://jason.sourceforge.net)
Existing Platforms

• Platforms
  • FIPA compliant
    • FIPA-OS (http://sourceforge.net/projects/fipa-os/)
    • Jade/LEAP (http://jade.tilab.com/)
  • Others:
    • SACI Simple Agent Communication Infrastructure (http://www.lti.pcs.usp.br/saci/)

• Developping Environments
  • Madkit (www.madkit.org)
  • JADEX, BDI agent model based on JADE (http://sourceforge.net/projects/jadex)
  • JACK execution environment, compiler, BDI agent model based on Procedural Reasoning System (PRS) (http://www.agent-software.com)
  • AgentBuilder based on Agent Oriented Program (AOP) (http://www.agentbuilder.com/)
  • AgentTool (http://macr.cis.ksu.edu/projects/agentTool/agentool.htm)
  • ADELFE (http://www.irit.fr/ADELFE/)

• Have a look at Software Products for MAS, AgentLink, June 2002
Standards

• **Knowledge Sharing Effort** The DARPA Knowledge Sharing Effort

• **MASIF - OMG** (Object Management Group) : OMG effort to standardize mobile agents - middleware services and internal middleware interfaces
  - [www.omg.org](http://www.omg.org)

• **IEEE Computer Society FIPA Standards Committee** (Foundation for Intelligent Physical Agents)
  - [www.fipa.org](http://www.fipa.org)
Introduction to multi-agent systems

FIPA Plateform

Standards

Non agent software
Agent

Agent Management Specification

HTTP
IIOP
SMTP
e tc.

ACL = Agent Communication Language

Structure Specification
ACL Message

Library of communication acts

Interaction Protocols Specification

Content Language Specification SL

Agent Message Transport Specification

Message Transport Service

Plateforme Agent FIPA

ACL = Agent Communication Language
JADE (Java Agent DEvelopment Framework)

- Middleware for developing agent-based P2P application
  - On fixed platforms, smart phones, ...
- Two main products:
  - Agent Platform compliant to FIPA specifications
  - API to develop agents in Java
- Open Source Project, LGPL License
- Controlled by Telecom Italia Lab, who owns the project
- Result of the joint effort of multiple actors belonging to the JADE Board (founded in 2003) which missions concern the promotion, the governance and implementation of the changes of the JADE platform

- Project portal: http://jade.tilab.com
Conteneurs

- Tout agent s’exécute dans un conteneur
- Un conteneur est un environnement d’exécution JADE :
  - Environnement complet d’exécution pour un agent
  - Exécution concurrente de plusieurs agents
  - Contrôle le cycle de vie des agents
  - Assure la communication entre agents
- Un seul conteneur héberge l’AMS, le DF, c’est le conteneur principal (main container)
  - AMS (Agent Management System)
    - Service de Pages Blanches : référence automatiquement les agents suivant leur nom dès leur entrée dans le système.
  - DF (Directory Facilitator)
    - Service de Pages Jaunes : référence à leur demande les agents suivant leur(s) service(s).
- Le conteneur principal peut être répliqué via des services de réplication
Caractéristiques

- Support à l’envoi de messages, transparent et multi-protocoles
  - Diffusion d’événements en local
  - Java RMI pour la diffusion interne à une plateforme
  - FIPA 2000 Message Transport Protocol
    - Protocole IIOP pour la diffusion inter-plateforme
    - Protocole HTTP et encodage des ACL en XML
- Modèle de concurrence à deux niveaux
  - Entre agents (pre-emptif, Threads Java)
  - Interne aux agents (co-opératif, classes de comportements “behaviour”)
- Mobilité des agents
  - entre plateforme, mobilité faible
- Framework orienté objet implémentant en Java les spécifications FIPA
- De nombreuses extensions :
  - **WSIG** : Web Service Integration Gateway
  - Semantics, Persistence, Sécurité, …
  - **WADE** : Workflows and Agents Development Environment
Caractéristiques

- JADE encapsule les spécifications de la FIPA :
  - La plateforme fournit : AMS, DF, MTS (ACC)
  - agent-management-ontology, constructeur d’un agent l’enregistre dans la plateforme (nom, adresse), classe DFServe permet un accès au DF
  - Transport et traitement des messages
  - Extension des protocoles standards d’interaction par les méthodes handle
- Système de gestion d’événements dans le noyau de la plateforme
  - Permet l’observation de la plateforme, des messages, du transport des messages, des agents
- Outils de gestion basés sur des agents
  - Agents spéciaux (RMA, Sniffer, Introspector) qui communiquent avec FIPA ACL
  - Extensions à l’ontologie fipa-management-ontology pour y inclure des actions spécifiques
  - Ontologie particulière pour l’observation jade-introspection
- Agents utilitaires
  - DummyAgent tool permet à des utilisateurs d’interagir avec les agents déployés sur la plateforme
  - Sniffer Agent agent utilisé pour observer les messages
Standards & Multi-Agent Systems

- Ontologies: DAML, OIL, OWL, ...
  - http://www.daml.org
  - http://www.ontoknowledge.org/oil/
  - http://www.w3.org/

- Other standards (De Facto)
  - Jini (www.sun.com/jini),
  - UPnP (www.upnp.org),
  - UDDI (www.uddi.org),
  - Salutation (www.salutation.org)
  - mobility: Aglets (www.trl.ibm.com/aglets/)
  - Web Services (http://www.w3.org/)
  - ...
Multi-Agent Methodologies

The engineering of Multi-Agent Systems needs to take into account two levels:

- **Multi-Agent System level (System-Centred)**
  - Number of agents, Agent Heterogeneity?
  - What is the common medium shared by the agents (Environment)?
  - What are the communication mechanisms between agents?
  - What are the communication languages, the ontologies, the interaction protocols used by the agents?
  - What is the organisation regulating the actions of the agents? How is it established?
  - How do the agents coordinate their actions? How to ensure a consistent behavior?

- **Agent level (Agent-Centred)**
  - What does an agent represent? What are the kinds of actions encapsulated into an agent?
  - How do the agents represent the environment, the organisation in which they are situated?
  - How do the agents process the interaction with other agents?
  - What is the agent architecture?
Tools supporting methods

- Software Engineering Tools supporting methodologies:
  - MASE AgentTool: macr.cis.ksu.edu/projects/agentTool/agentool.htm
  - ZEUS: sourceforge.net/projects/zeusagent
  - PASSI ToolKit: mozart.csai.unipa.it/passi/ptk.htm
  - INGENIAS: grasia.fdi.ucm.es/ingenias/
  - OPM: www.objectprocess.org

- Different ways to model applications:
  - Agent Oriented Software Engineering
  - Environment Oriented Software Engineering
  - Interaction Oriented Software Engineering
  - Organization Oriented Software Engineering
Plan

1. Definitions
2. Action Domains
3. Positioning
4. Multi-Agent Perspective
5. Multi-Agent Engineering
6. Perspectives …
To continue …

- **General references**
  - Les systèmes multi-agents, J. Ferber, InterEditions, 1995
  - Principes et architectures des Systèmes Multi-Agents, J.P. Briot, Y. Demazeau, IC2, Hermès, 2001

- **Some standards**
  - Knowledge Sharing Effort http://www.cs.umbc.edu/kse/
  - FIPA http://www.fipa.org
  - W3C http://www.w3.org

- **Some general addresses**
  - AgentLink : http://www.agentlink.org
  - AgentCities : http://www.agentcities.org
Multi-Agent Modeling

- Multi-model:
  - Articulation of different formalisms

- Multi-viewpoints:
  - Extern/intern, system centred/agent centred
  - Multiple views on a shared world

- Multi-levels
  - Via organisations, via the environment (MAS)

- Multi-scales
  - temporal, spatial, …
Multi-Agent Features

Artificial Intelligence
- Knowledge Sharing

Artificial Life
- Adaptation

Software Agent
- Delegation

Autonomy
- Decentralisation
- Interaction

Organisation
- Situated in Environment
- Openness

Situated in Environment
- Emergence

Distribution

Personnalisation

Intelligibility

Distributed Systems

Human Machine Interaction

Software Engineering

Intelligence
- Sharing

Adaptation

Delegation

Distribution

Personnalisation

Intelligibility
Scientific Challenges

Complexity
Applications

Short-term
Middle-term
Middle-term
Long-term

“closed” MAS
“inter-Organisational” MAS
“open” MAS
“massive” MAS

Source: AgentLink Roadmap

• a “same” goal
• adhoc coordination mechanisms
• Ad-hoc design
• Scalability in simulation

• different goals
• coordination mechanisms
• Multi-Agent Design
• Important number of agents, scalability

• generic coordination mechanisms
• Standards for the design of MAS
• Scalability in domains,
• Robustness,
• Openness

• Learning of coordination mechanisms
• Emergent Coordination
• Scalability
Applicative Challenges
Domain Overview (1/2)

• International Conferences
  – International Conference on Multi-Agent System (ICMAS) de 1995 à 2000,

• French Conferences
  – Journées Francophones SMA (http://www.cerv.fr/jfsma08/)
  – Collège SMA de l’AFIA (http://sma.lip6.fr/)

• European Projects
  – AgentLink (réseau d’excellence www.agentlink.org), Roadmap (www.agentlink.org/ roadmap)

• Some “Success Stories”
  – Brahms (agentsolutions http://agentsolutions.com/home.htm) @ NASA Ames Research Center
  – Living Systems (Whitestein technologies http://www.whitestein.com) @ ABX Logistics
  – eSTAR (http://www.estar.org.uk/) intelligent robotic telescope network
  – CalicoJack (http://www.calicojack.co.uk/)
Domain Overview (2/2)

- Standards
  - FIPA (Foundation for Intelligent Physical Agents) (http://www.fipa.org/)

- Competitions
  - http://www.robocup.org/
  - http://www.rescuesystem.org/robocuprescue/
  - http://tac.eecs.umich.edu/association.html
  - http://www.lips.utexas.edu/art-testbed/
Domain Overview (3/3)

- Journals
  - Autonomous Agents and Multi-Agent Systems
  - Artificial Intelligence
  - Knowledge Engineering Review
  - International Journal of Agent-Oriented Software Engineering (IJAOSE)
  - Web Intelligence and Agent Systems An International Journal

- News
  - Agent List
    - http://www.cs.umbc.edu/agentslist/
  - Distributed Artificial Intelligence List
    - DAI-List-Request@ece.sc.edu
  - French list
    - sma@loria.fr
Bibliography


Bibliography


Bibliography


Bibliography

Bibliography


• [Muller 08] G. Muller, L. Vercouter, Trust and Reputation, cours EASSS 08


Introduction to multi-agent systems

© O. Boissier MINES Saint-Etienne

Bibliography

- [Terziyan 07] Vagan Terziyan, MIT Department, University of Jyvaskyla, AI Department, Kharkov National University of Radioelectronics http://www.cs.jyu.fi/ai/vagan/index.html
Bibliography

- E-Game: http://csres43.essex.ac.uk:8080/elearn/eg/