Multi-Agent Oriented Programming

Olivier Boissier

MAS Dpt/G2I/EMSE
158 cours Fauriel,
42023 Saint-Etienne Cedex
France

Olivier.Boissier@emse.fr
Tel : 33 4 77 42 66 14
Fax : 33 4 77 42 66 66
Motivation and challenges

- Industries and Services:
  - Are still becoming global and complex, at the intersection of multiple networks, faced to a constant acceleration of changes
  - require to combine **efficiency** in enacting process that are at the heart of their organization, while insuring **flexibility** in their definition

Ex: E-Alliance [Bratu 00]

Ambient Intelligence [Vallée 04]
Today is being developed an **a software and material IT infrastructure**, in which:

- More and more business components are proposed (Web Services, …),
- Knowledge gains an increasing place (Semantic Web, …),
- Business process automation (Business Process, Workflow, …) is more and more present,
- Services, resources, teams, enterprises, communities are increasing.

Supported by and contributing to:

- Web Intelligence
- Ambient Intelligence
- Complex Systems
Objectives

- Development of intelligent software (reactive, proactive) that operate effectively in environments that:
  - Have no centralized control,
  - Are highly interconnected,
  - Are in constant state of flux,
  - Are highly unpredictable,
  - Involve multi-* (site, actors, expertise, viewpoints, decisions, ...),
  - Involve individually-motivated actors.

- Insuring QoS:
  - Reliability, Robustness, Security,
  - Flexibility, Adaptation, Interoperability,
  - Efficiency, Control, scalability.
Modeling Multi-Agent World

Agent for the management of the Alliance
Agent for the management of the Contracts on behalf of a Printshop
Agent for the management of Negotiations on behalf of a Printshop
Agent for the interaction with a Printshop

[Castellani 03]
Programming Multi-Agent World

- How do we engineer Environment, Interaction, Organization, Agent?
- What is the control flow between E, I, O and A?
Outline

• Introduction
• Case studies
• Multi-Agent Oriented Programming
• Conclusion
Outline

- Introduction
- Case studies:
  - STARS
  - E-Alliance
  - JojTeam
- Multi-Agent Oriented Programming
- Conclusion
Dynamic Systems Supervision

Terminus A

Terminus D

Matching

Outward

[Allouche 98]
Environment simulation

Temporal ACL Interaction Protocols

Interaction Services

Interaction

Outward

Terminus A

Closing

Opening

Terminus D

Return

Running

Loading

Unloading

Motor

Interaction Services

Temporal ACL Interaction Protocols

resp(a1)

resp(a2)

resp(a3)

resp(a4)

resp(a5)

resp(a6)
STARS: Agents Programming (1)

Interaction Services

Messages

Reception

Emission

Social Reasoning

Decision

Planning

Task Management

Perception

External Desc.
agent: A1 ...
agent: A2 ...
agent: A3 ...

Dependance networks

Mental State

Tasks
Environment Model

Domain Knowledge

Tasks

Environment Simulation

Control flow

Data flow

Domain Knowledge

Events

Environment Simulation
• **Temporal dependence Networks**

  ✓ **needs (N)**

  \[
  \text{needs}(a_u, a_v, p_i, p_j) \equiv p_i \in \text{resp}(a_u) \land p_j \in \text{resp}(a_v) \land p_j \notin \text{resp}(a_u)
  \]

  ✓ **helps (H)**

  \[
  \text{helps}(a_u, a_v, p_i, p_j) \equiv \text{opp}(p_i, p_j) \land p_j \in \text{resp}(a_u) \land p_j \notin \text{resp}(a_v)
  \]

  ✓ **competes (C)**

  \[
  \text{competes}(a_u, a_v, p_i) \equiv p_j \in \text{resp}(a_u) \land \text{resp}(a_v)
  \]

resp (a1) = {Outward, Return, TerminusA}
resp (a2) = {Outward, Return, TerminusD}
resp (a3) = {Outward, Return, Running}
resp (a4) = {Loading, Unloading}
resp (a5) = {Opening, Closing}
resp (a6) = {Motor}
**Outline**

- Introduction
- **Case studies:**
  - STARS
  - E-Alliance
  - JojTeam
- Multi-Agent Oriented Programming
- Conclusion
Aim: supporting out/insourcing within inter-organizational alliances

(negotiation and contracts fulfillment)
Propose a contract for 50K BW and 50K Binding, at the best price.

My schedule is too tight. I can’t accept !!!

Client

Printshop A

Printshop B

self

14h 10h
15h 18h

Printshop C
E-Alliance: out/in-sourcing

Multi-issues, Multi-parties, Multi-stages, And Open Negotiation

Client

Printshop A

Printshop B

Printshop C

Propose Job1
- type = BW
- size <= 50K
- cost <= 60€

Propose Job1
- type = BW
- size <= 50K
- cost <= 60€

Client

Ok

Proposal:
- Job1
  - type = BW
  - size <= 50K
  - cost <= 60€

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE
E-Alliance : out/in-sourcing

Accept job1 size=40k
cost=38€

Accept job1
Size=10k
cost=17€

Client

Printshop A

Printshop B

Printshop C
E-Alliance : negotiation

• Generic layer : collaborative building of negotiation graphs.
  • Posting and diffusing issues, alternatives,
  • Initiating, closing negotiations,
  • Persistence of negotiations.

• Strategic layer : negotiation decisions for building the graphs
  • Negotiation Object, Negotiation Framework, Tactics
• Manager:
  - Initiating Negotiation;

• NegF Agent:
  - Build and make offers
  - Verify constraints
  - Run Negotiation Protocols
  - Negotiation Coordination
  - Interaction with the Manager

• Negotiation Framework: How to negotiate

```plaintext
<ON> ::= <SON> | '('<name> <ON>* :dependencies <DDF>')'
<SON> ::= '('<name> :type <TDF> :candidates <CDF>
      <job <JDF> :dependencies <DDF>')'
<TDF> ::= BWP | CP | B
<CDF> ::= '('<CDF>* [<Relation>]')'|<Name>|'?'|'?*'
<Relation> ::= >> | <<
```

• Negotiation Object: What to negotiate

```plaintext
<ON> ::= <SON> |'
```

• Negotiation Tactic: Constraints on the negotiation process

```plaintext
outs rc-job @ partner(dest,job) @ insrc(dest,job) @
asablock(job) <--> transfer(job)
```
Interaction Programming in a Negotiation middleware

Xplore Protocol: **collaborative** construction of negotiation graphs

- **Connect** \((n, m)\): connection to a negotiation whose root node is \(n\).

- **Open** \((n, n_1, \ldots, n_p)\): creation of a new node \(n\) in a negotiation graph from nodes \(n_1, \ldots, n_p\).

- **Assert** \((n, v, i, t)\): decision that, in the negotiation node \(n\), the value of decision variable \(v\) must have the property \(t\) pertaining to issue \(i\).

- **Request** \((n, v, i)\): need to obtain information about issue \(i\) of decision variable \(v\) at node \(n\).

CooF: **Ready** \((n)\): satisfaction with the state of the negotiation at node \(n\).

- Diffusion of alternatives, issues.
- **Quit** \((n)\): closing of negotiation at node \(n\).
- Transactional properties.
E-Alliance: Interaction programming
Propose a job:
Size=10K  Cost<25
Delay<4, ....
Outline

• Introduction

• **Case studies:**
  • STARS
  • E-Alliance
  • JojTeam

• Multi-Agent Oriented Programming

• Conclusion
JojTeam

Teambots Simulator

[Hübner 03]
JojTeam : Structural Specification

Roles

- coach
- middle
- attacker
- leader
- back
- goalkeeper

Groups

- defense
- attack
- team

Links

- Marcos
- Lucio
- Edmilson
- Roque Jr.
- Cafu
- Gilberto Silva
- Juninho
- Ronaldinho
- Roberto Carlos
- Ronaldo
- Rivaldo

Organizational Entity

Structure 3-5-2
JojTeam: Functional Specification

**Mission:**

**Goal:**

- **Sequence:**
  - Get the ball (m1)
  - Go toward the opponent field (m1)
  - Be placed in the middle field (m2)
  - Be placed in the opponent goal area (m3)

- **Choice:**
  - Shot at the opponent’s goal (m3)
  - Kick the ball to the goal area (m2)
  - Go to the opponent back line (m2)
  - Kick the ball to the agent (m1)
  - Committed to m2

- **Parallelisme:**
  - m1, m2, m3
JojTeam: Deontic Specification

Permissions

Obligations

Organisational Entity
Lucio ----- m1
Cafu ----- m2
Rivaldo ----- m3
JojTeam: Programming Organization in Agents

- Organizational layer
- Deliberative layer
- Reactive layer
- Communication
- Action
- Perception

Other agents
MOISE+ Organization Specification
Environment
JojTeam: Programming Organization Middleware

MOISE+ Organization Specification

MOISE+ Agent

OrgManager

OrgWrapper

OrgWrapper

OrgWrapper

Organization Services S-MOISE+

Interaction Services

SACI platform

Organizational events
Outline

• Introduction
• Case studies
• Multi-Agent Oriented Programming
• Conclusion
Synthesis

• STARS:
  • Design and Programmation are centered around the Agent dimension.

• E-Alliance:
  • Design is centered around the Interaction dimension.
  • Programmation is mainly present in Agent, and in Interaction middleware (Negotiation middleware)

• JojTeam:
  • Design is centered around the Organization dimension.
  • Programmation is mainly present in Agent, and in an Organization middleware
MAS Analysis and Design

- Agent Centered Analysis and Design
  - STARS [Allouche], AOP [Shoham]

- Interaction Centered Analysis and Design
  - E-Alliance [Castellani], IOP [Huhns]

- Organisation Centered Analysis and Design
  - JojTEAM [Hübner], AGR [Ferber], TOP [Tambe]

- Environnement Centered Analysis and Design
  - Simulations [Bousquet, Cirad], MASSIVE
Multi-Agent Oriented Programming

- **Agent Oriented Programming** [Shoham 92]:
  - Models are programmed within the agents

- **Interaction Oriented Programming** [Singh, Huhns 00]:
  - Interaction Layer (Middleware) controlling interactions between agents

- **Organization Oriented Programming** [Tambe 01]:
  - Organization Layer (Middleware) controlling and imposing organizations on agents

- **Environment Oriented Programming**
Outline

• Introduction
• Case studies
• Multi-Agent Oriented Programming
• Conclusion
Multi-Agent Oriented Programming

- Not
  - Object-Oriented Programming (Objects + Message passing)
  - Logic nor Expert Systems Programming (Knowledge + Inference Mechanism)
  - Ontology-Oriented Programming (Knowledge + Problem Solving Methods)

- BUT:
  - Agent Oriented Programming
  - Interaction Oriented Programming
  - Organization Oriented Programming
  - Environment Oriented Programming
Multi-Agent Oriented Development

MAS Analysis & Design

MAS Generic Components
A,E,I,O: MAS level Components/Services
a,e,i,o: Agent Level Components

MAS Deployment & Observation

Agent Architectures

MAS Services
Organization Services
Interaction Services
Environment Simulation/Services

Middleware
What is the control flow between E, I, O and A?
Autonomous Agents in a Multi-Agent World

• What about autonomy of the Agent with respect to E, I, O when engineering a multi-agent world?

• An agent X is autonomous with respect to Y for P in a context C, noted is_autonomous (X, Y, P, C) if, the behaviour of X in C concerning P is not imposed by Y
  • Y – the influencer of autonomy: another agent, the organization, the environment, the user, etc.
  • P – the object of autonomy: the adoption of a goal (plan, action, etc.), the making of a decision, etc.
  • C – the context: the same agent can be autonomous in one situation and non-autonomous in another.

• How to manage/program constraints coming from other agents, from the organization, from the environment, from the user(s), from the designer, etc, to an agent?
What about Domain?

- MAS are not always A-centered!

- (((A + E) + I) + O) Robotics Science
- (((A + I) + O) + E) Social Science
- ((E + A) + (I + O)) Life Science
- (((I + O) + A) + E) Military Science
- (((O + I) + E) + A) Economic Science

[Demazeau]