Jomi F. Hübner†, Rafael H. Bordini⋆ 
Gauthier Picard†

† ENS Mines Saint Etienne, France 
{hubner,picard}@emse.fr

⋆ University of Durham, UK 
r.bordini@durham.ac.uk

Dagstuhl Seminar on Programming Multi-Agent Systems
1 Introduction
2 Design
3 Implementation
4 Conclusions
Agent Contest 2008
the Cows and Herders scenario
Objectives of our participation

• In 2006: program our agents using plans
  ⊲ reactive agents
• In 2007: program our agents using goals
  ⊲ goal directed agents
• In 2008: program our agents using organisation
  ⊲ join agent and system levels
  ⇔ use Jason for the agents
  ⇔ use MOISE+ for the organisation

• Test and improve Jason and MOISE+ software
• Evaluate the use of organisational constructors in the development of the team
Objectives of our participation

- In 2006: program our agents using plans
  - reactive agents
- In 2007: program our agents using goals
  - goal directed agents
- In 2008: program our agents using organisation
  - join agent and system levels
    - use Jason for the agents
    - use Moise$^+$ for the organisation
- Test and improve Jason and Moise$^+$ software
- Evaluate the use of organisational constructors in the development of the team
1 Introduction
   - context
   - objectives

2 Design
   - specification
   - dynamics
   - goals

3 Implementation
   - MOISE+
   - Jason
   - Java
   - tools

4 Conclusions
   - results
   - discussion
Our agents are organised in two types of groups:

- Exploration group: find cows
- Herding group: push cows into the corral
Exploration group

- Three instances of exploration group
- Each group is allocated to an area of the scenario
  - One explorer: decides where to go
  - One scouter: follows explorer
Exploration group

- Three instances of exploration group
- Each group is allocated to an area of the scenario
- One explorer: decides where to go
- One scouter: follows explorer
Herding group

- Created when a member of the team sees some cow
- As many instances as the number of clusters of cows
  - One herder: defines the team formation for the cluster
  - scouters: be in formation
  - spatial coordination
Herding group

- Created when a member of the team sees some cow
- As many instances as the number of clusters of cows
- One herder: defines the team formation for the cluster
- Scouters: be in formation
- Spatial coordination
Herding group

- Created when a member of the team sees some cow
- As many instances as the number of clusters of cows
- One herder: defines the team formation for the cluster
- Scouts: be in formation
- Spatial coordination
Start exploring

Exploring

explorer

scouter
Creation of herding group

Exploring
- explorer
- scouter

Herding
- herder
- herdboy
Merge two herding groups

Structural Dynamics
Dissolve herding group

Herding

<table>
<thead>
<tr>
<th>herder</th>
</tr>
</thead>
<tbody>
<tr>
<td>herdboy</td>
</tr>
</tbody>
</table>

Exploring

| explorer |
| scouter |

...
Explorer (the leader) is obligated to mission / Scouter is obligated to s
Herder (the leader) is obligated to mission \( l \)
Herdboy is obligated to \( b \)
## Goals

<table>
<thead>
<tr>
<th>Role</th>
<th>Goal</th>
<th>Goal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>explorer</strong></td>
<td>find_scouter</td>
<td>find agent nearby to play scouter</td>
</tr>
<tr>
<td></td>
<td>change_to_herding</td>
<td>change to a herding group</td>
</tr>
<tr>
<td></td>
<td>goto_near_unvisited</td>
<td>go to the nearest unvisited location</td>
</tr>
<tr>
<td><strong>scouter</strong></td>
<td>share_seen_cows</td>
<td>share information about cows</td>
</tr>
<tr>
<td></td>
<td>follow_leader</td>
<td>follow the leader of the group</td>
</tr>
<tr>
<td><strong>herder</strong></td>
<td>recruit</td>
<td>recruit more herdboys</td>
</tr>
<tr>
<td></td>
<td>release_boys</td>
<td>release some herdboys</td>
</tr>
<tr>
<td></td>
<td>define_formation</td>
<td>compute the formation of the group</td>
</tr>
<tr>
<td></td>
<td>be_in_formation</td>
<td>go to the place allocated to the agent</td>
</tr>
<tr>
<td></td>
<td>merge</td>
<td>merge two herding groups</td>
</tr>
<tr>
<td></td>
<td>change_to_exploring</td>
<td>change to an exploring group</td>
</tr>
<tr>
<td><strong>herdboy</strong></td>
<td>share_seen_cows</td>
<td></td>
</tr>
<tr>
<td></td>
<td>be_in_formation</td>
<td></td>
</tr>
</tbody>
</table>
1. Introduction
   - context
   - objectives

2. Design
   - specification
   - dynamics
   - goals

3. Implementation
   - \texttt{MOISE^+}
   - Jason
   - Java
   - tools

4. Conclusions
   - results
   - discussion
...<group-specification id="team">
  <sub-groups>
    <group-specification id="exploration_grp" min="0" max="3">
      <roles>
        <role id="explorer" min="1" max="1"/>
        <role id="scouter" min="0" max="1"/>
      </roles>
    </group-specification>
  </sub-groups>
</group-specification>
...

Tools to run the organisation:

- \(S-\text{MOISE}^+\): organisational infrastructure
  - manage the state of the organisation
- \(J-\text{MOISE}^+\): integration with \textbf{Jason}
  - library of organisational actions
  - organisational architecture
Agents are informed about their obligations

\[ \text{new goal event} \]

Plan to handle a new goal — maintenance goal pattern

- the goal is annotated with the group and role that generated the obligation

```java
+!define_formation[group(G),role(R)]
<- ... <the code> ...

// wait for the next cycle
.wait("+pos(X,Y,Cycle)" confusion);

// achieve that goal again
!define_formation[group(G),role(R)].
```
Agents are also informed by changes in the organisation
∴ change the belief base
∴ produce events

Examples

+play(Me, herder, G)
  : .my_name(Me)
  <- +group_leader(G, Me);
  .broadcast(tell, group_leader(G, Me)).

-group(Type, GroupId)
  <- .drop_intention(_[group(GroupId)]).
The achievement of organisational goals is implemented in Jason

**Goal: merge herding group**

**plan** merge

let $g_i$ be my herding group

forall herding group $g_j$ such that $g_i > g_j$ do

let $S_i$ be the set of cows of $g_i$'s cluster
let $S_j$ be the set of cows of $g_j$'s cluster

if $S_i \cap S_j \neq \emptyset$ then

remove group $g_j$ from the organisation
ask all agents of $g_j$ to adopt the role *herdboy* in $g_i$

// new role → new goals
Agent Oriented Programming II

Code in **Jason**

```jason
+!merge
    : .my_name(Me) &
    play(Me, herder, Gi) & // I play role herder
    current_cluster(MyC) // MyC is the list with my cows

<- // for all other groups
    for (group_leader(Gj, L) & Me < L) {
        .send(L,askOne,current_cluster(_),current_cluster(TC));
        .intersection(MyC,TC,I);
        if (I \== []) {
            .send(L, achieve, change_role(herdboy,Gi))
        }
    }
).
```

When the leader of other group change the role, he will ask his herdboys to also change the group
Object Oriented Programming

The following components were implemented in Java:

- Integration with the contest simulator
- Agent perception and action
- Find paths: A*
- Compute the formation (a lot of vector calculations)
- ...

# 130 lines of code in MOISE+
# 696 lines of code in Jason
# 4218 lines of code in Java
The following components were implemented in **Java**

- Integration with the contest simulator
- Agent perception and action
- Find paths: A*
- Compute the formation (a lot of vector calculations)
- ...

# 130 lines of code in **MOISE**+
# 696 lines of code in **Jason**
# 4218 lines of code in **Java**
Useful tools — Mind inspector

- Rules

\[
\text{random\_pos}(X,Y) :- \\
(\text{pos}(AgX,AgY,\_418) \& \text{(jia.random}(RX,40) \& ((RX > 5) \& ((X = ((RX-20)+AgX)) \& ((X >}

- Intentions

<table>
<thead>
<tr>
<th>Sel</th>
<th>Id</th>
<th>Pen</th>
<th>Intended Means Stack (hide details)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16927</td>
<td>suspended-self</td>
<td>+!be_in_formation[scheme(sch_herd_sch_12),mission(help)]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+!be_in_formation[scheme(Sch),mission(Mission)]</td>
<td></td>
</tr>
</tbody>
</table>
Useful tools — Moise$^+$ GUI

jason-cowboys (Organisational Entity)

Agents
gaacho3; gaacho4; gaacho5; gaacho6; gaacho1; gaacho2;

Groups

- gr_team_01
  - gr_exploration_grp_02 players (2): gaacho3 (explorer); gaacho6 (scouter);
  - gr_herdng_grp_05 players (4): gaacho1 (herder); gaacho2 (herdboy); gaacho4 (herdboy); gaacho5 (herdboy);
Typical screen
Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
  - watch the result in the contest scenarios
  - read and analyse long logs, minds dumps, traces, performance...
    - note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $S$-$MOISE^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again
Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...
  note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $S$-$MOISE^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again
Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...
  - note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $S$-$\text{MOISE}^+$, ...), start again
  - tuning of parameters (the cluster size?), start again
  - give up the idea, start again
Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...
  note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $S$-MOISE$^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again
Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...
  note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $S$-$\text{MOISE}^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again
Team

- agents are autonomous to
  - adopt roles
  - decide how to achieve goals
- coordination is essentially spacial
  (follow leader and formation)
- communication is used to share information
  (speech act based)
Summary II

- Jason
  - declarative and goal oriented programming
  - goal patterns (maintenance goal)
  - meta-programming (*.drop.intention(_.[group(g1)])
  - customisations (integration with the simulator and the organisation)
  - internal actions (code in Java)
  - good programming style
Summary III

- \textit{MOISE}^+
  - definition of groups and roles
  - allocation of goals to agents based on their roles
  - to change the team, we (developers) ‘simply’ change the organisation
  - global orchestration
  - team strategy defined at a high level
Good points

- New scenario of the contest
- Use of 3 programming paradigms
- Improve several issues of Jason, $\textit{MOISE}^+$, and their integration
  - New type of goal in $\textit{MOISE}^+$ (maintenance goal)
  - More suitable for collaborative systems (group deletion)
Weak points

- Too much time in ‘debug, test, and tuning mode’
  we rather prefer analysis and programming
- The organisation dynamics is specified inside the agents
  it is coded and mixed in the agent’s plans
  → new language to define it from a global perspective
- The functional dimension of the team is quite simple
  it allows the definition of global plans useful to achieve shared goals
  → more complex team strategies
  → changes in the scenario
- It is quite difficult to map an idea into different levels of analysis
  what is organisation and what is agent planning; what is MOISE+, Jason, or Java
Weak points

- Too much time in ‘debug, test, and tuning mode’
  we rather prefer analysis and programming

- The organisation dynamics is specified inside the agents
  it is coded and mixed in the agent’s plans
    → new language to define it from a global perspective

- The functional dimension of the team is quite simple
  it allows the definition of global plans useful to achieve shared goals
    → more complex team strategies
    → changes in the scenario

- It is quite difficult to map an idea into different levels of analysis
  what is organisation and what is agent planning; what is \texttt{MOISE}+, \texttt{Jason}, or Java
Weak points

- Too much time in ‘debug, test, and tuning mode’
  we rather prefer analysis and programming
- The organisation dynamics is specified inside the agents
  it is coded and mixed in the agent’s plans
  → new language to define it from a global perspective
- The functional dimension of the team is quite simple
  it allows the definition of global plans useful to achieve shared goals
  → more complex team strategies
  → changes in the scenario
- It is quite difficult to map an idea into different levels of analysis
  what is organisation and what is agent planning; what is \textit{MOISE}^{+}, \textit{Jason}, or Java
Weak points

- Too much time in ‘debug, test, and tuning mode’
  we rather prefer analysis and programming
- The organisation dynamics is specified inside the agents
  it is coded and mixed in the agent’s plans
  → new language to define it from a global perspective
- The functional dimension of the team is quite simple
  it allows the definition of global plans useful to achieve shared goals
  → more complex team strategies
  → changes in the scenario
- It is quite difficult to map an idea into different levels of analysis
  what is organisation and what is agent planning; what is Moise+, Jason, or Java
http://moise.sf.net

http://jason.sf.net
(the code of our agents is available there)