Multi-Agent Oriented Programming
– Introduction –
The JaCaMo Platform

O. Boissier\textsuperscript{1} R.H. Bordini\textsuperscript{2} J.F. Hübner\textsuperscript{3} A. Ricci\textsuperscript{4}

1. Ecole Nationale Supérieure des Mines (ENSMSE), Saint Etienne, France
2. Pontificia Universidade Catolica do Rio Grande do Sul (PUCRS), Porto Alegre, Brazil
3. Federal University of Santa Catarina (UFSC), Florianópolis, Brazil
4. University of Bologna (UNIBO), Bologna, Italy

September 2015
In collaboration with

- **J.S. Sichman**, Universidade de São Paulo - LTI-PCS, São Paulo, Brazil (jaime.sichman@poli.usp.br)

- **G. Picard**, ENS Mines St-Etienne, France (gauthier.picard@emse.fr)

- **M. Hannoun, B. Gâteau, G. Danoy, R. Kitio, C. Persson, R. Yaich**, ENS Mines St-Etienne, France, **L. Coutinho** Brazil

- **M. Piunti, A. Santi**, Università degli studi di Bologna - DEIS, Bologna, Italy (a.ricci@unibo.it)

- **A. Ciortea, A. Sorici**, Politehnica University of Bucharest, Romania
Outline

1 Introduction
2 Multi-Agent Oriented Programming (MAOP)
3 MAOP Perspective: the JaCaMo Platform
4 MAOP Experiences
5 Conclusions and Perspectives
MAS Conceptual framework / Dimensions

- **Agents**: abstractions for the definition of the decision/reasoning entities architectures
- **Environment**: abstractions for structuring resources, processing entities shared among the agents
- **Interaction**: abstractions for structuring interactions among entities
- **Organisation**: abstractions for structuring and ruling the sets of entities within the MAS

A rich set of abstractions for capturing applications complexity!

Each dimension has its own dynamics

Dynamics may be interlaced into bottom-up / top-down global cycles

Coordination of these dynamics may be programmed into one or several dimensions [Boissier, 2003]

~ A rich palette of possible dynamics & coordination!!
In these approaches, some dimensions lose their control & visibility!

Integrating the dimensions into one programming platform is not so easy!

Examples of Multi-Agent Oriented Programming Platforms:
- Volcano platform [Ricordel and Demazeau, 2002],
- MASK platform [Occello et al., 2004],
- MASQ [Stratulat et al., 2009],
- extending AGRE and AGREEN,
- Situated E-Institutions [Campos et al., 2009], ...

Agent Oriented Programming [Shoham, 1993]

Environment Oriented Programming [Ricci et al., 2011]

Interaction Oriented Programming [Huhns, 2001]

Organisation Oriented Programming [Pynadath et al., 1999]
MAS Programming

- **Agent Oriented Programming** [Shoham, 1993]
- **Environment Oriented Programming** [Ricci et al., 2011]
- **Interaction Oriented Programming** [Huhns, 2001]
- **Organisation Oriented Programming** [Pynadath et al., 1999]

**Challenge**

Shifting from an A/E/I/O oriented approaches to a **Multi-Agent** Oriented approach

- **keeping alive** the concepts, dynamics and coordinations of the A, E, I and O dimensions
Outline

1. Introduction

2. Multi-Agent Oriented Programming (MAOP)
   - MAOP Meta-Model
   - Focus on Agent meta-model
   - Focus on Environment meta-model
   - Focus on Organisation meta-model
   - Synthesis

3. MAOP Perspective: the JaCaMo Platform

4. MAOP Experiences

5. Conclusions and Perspectives
Seamless Integration of A & E & I & O

JaCaMo Meta-model [Boissier et al., 2011], based on Cartago [Ricci et al., 2009b], Jason [Bordini et al., 2007], MOISE [Hübner et al., 2009a] meta-models
Agent meta-model

Based on Jason meta-models [Bordini et al., 2007]
Agent example I

**Example (Giacomo Agent Code)**

```plaintext
!have_a_house.  // Initial Goal
/* Plan */
+!have_a_house <- !contract;
   !execute.
```

**Example (companyX Agent Code)**

```plaintext
my_price(300).  // initial belief
/* plans for contracting phase */
// there is a new value for current bid
+currentBid(V)
   :  not i_am_winning(Art) & // I am not the current winner
      my_price(P) & P < V // I can offer a better bid
   <- .bid( P ). // place my bid offering a cheaper service
```
Agent & Agent Interaction meta-model

Agent Dimension

External Action → Internal Action

Action → Plan

Plan → Trigger event

Trigger event → Belief

Belief → Goal

Interaction Dimension

Message

Content → SpeechAct
Agent’s dynamics

1. Percepts
   - perceive

2. BUF

3. Messages
   - checkMail

4. SocAcc
   - Beliefs to Add and Delete

5. SE

6. Unify Event
   - Relevant Plans
   - Plans
   - Beliefs

7. Check Context
   - Events
   - Internal Events

8. SI
   - Intended Means
   - New Intention
   - New
   - Suspended Intentions (Actions and Msgs)

9. Selected Intention
   - New
   - Messages

10. Execute Intention
    - Push New Plan
    - New Intention
    - Updated Intention
    - Messages

Plan Library

Agent

Percepts

Actions

Messages
Environment meta-model

Based on A&A meta-model [Omicini et al., 2008]
Example

public class AuctionArt extends Artifact {

    @OPERATION void init(String taskDs, int maxValue) {
        defineObsProperty("task", taskDs); // task description
        defineObsProperty("maxValue", maxValue); // max. value
        // current best bid (lower service price)
        defineObsProperty("currentBid", maxValue);
        // current winning agent ID
        defineObsProperty("currentWinner", "no_winner");
    }

    // places a new bid for doing the service for price p
    // (used by company agents to bid in a given auction)
    @OPERATION void bid(double bidValue) {
        ObsProperty opCurrentValue = getObsProperty("currentBid");
        ObsProperty opCurrentWinner = getObsProperty("currentWinner");
        if (bidValue < opCurrentValue.intValue()) {
            opCurrentValue.updateValue(bidValue);
            opCurrentWinner.updateValue(getOpUserName());
        }
    }
}
A & E Interaction meta-model

Environment Dimension
- Workspace
- Environment

Artifact
- Manual

Operation
- has

Agent
- has
- create, dispose, link, unlink
- focus, unfocus

Observable Property
- create, join, quit
- focus, unfocus

Observable Event
- generate
- focus, unfocus

Agent Dimension
- External Action
- Internal Action

Manual
- consult

Action
- Plan
- Trigger event
- Belief
- Goal

Environment Dimension
- focus, unfocus

Focus, unfocus

Update, generate

Has

Has
Example

!have_a_house.  // Initial Goal
/* Plans */
+!have_a_house <- !contract; !execute.
+!contract <- !create_auction_artifacts; !wait_for_bids.
+!create_auction_artifacts
   <- !create_auction_artifact("SitePreparation", 2000);
   !create_auction_artifact("Floors", 1000);
   !create_auction_artifact("Walls", 1000);
   !create_auction_artifact("Roof", 2000);
   !create_auction_artifact("WindowsDoors", 2500);
   !create_auction_artifact("Plumbing", 500);
   !create_auction_artifact("ElectricalSystem", 500);
   !create_auction_artifact("Painting", 1200).
Example

```prolog
+!create_auction_artifact(Task, MaxPrice)
    <- .concat("auction_for_", Task, ArtName);
    makeArtifact(ArtName, "tools.AuctionArt", [Task, MaxPrice], ArtId);
    focus(ArtId).
-!create_auction_artifact(Task, MaxPrice)[error_code(Code)]
    <- .print("Error creating artifact ", Code).
+!wait_for_bids
    <- println("Waiting the bids for 5 seconds...");
    .wait(5000); // use intern deadline of 5 sec to close auctions
    !show_winners.
+!show_winners
    <- for ( currentWinner(Ag)[artifact_id(ArtId)] ) {
            ?currentBid(Price)[artifact_id(ArtId)]; // check current bid
            ?task(Task)[artifact_id(ArtId)]; // and task it is for
            println("Winner of task ", Task," is ", Ag," for ", Price)
        }.
```
Example

my_price(1500). // initial belief
!discover_art("auction_for_Plumbing"). // initial goal
i_am_winning(Art) :- .my_name(Me) &
    currentWinner(Me)[artifact_id(Art)].

/* plans for contracting phase */
+!discover_art(ToolName)
    <- joinWorkspace("HouseBuildingWsp");
    lookupArtifact(ToolName,ToolId);
    focus(ToolId).

// there is a new value for current bid
+currentBid(V)[artifact_id(Art)]
    : not i_am_winning(Art) & // I am not the current winner
        my_price(P) & P < V // I can offer a better bid
        <- bid(math.max(V-150, P))[artifact_id(Art)].

/* plans for execution phase */
...
Environment’s dynamics

Artifact life-cycle
- Creation/Deletion
- Activation/Execution/Fail or Success/Deactivation of an Operation
- Linking / Unlinking

Workspace life-cycle
- Creation/Deletion of a workspace
- Creation/Deletion of Artifacts
- Creation/Deletion & Entry/Exit of Agents
Agents with dynamic action repertoire, extended/reshaped by agents themselves

Uniform implementation of any mechanisms (e.g. coordination mechanism) in terms of actions/percepts
  - No need to extend agents with special purpose primitives

Exploiting a new type of agent modularity, based on externalization [Ricci et al., 2009a]
Organisation meta-model

Simplified MOISE meta-model [Hübner et al., 2009a]
Example: Organisation Structural Specification

Graphical representation of MOISE Struct. Spec.
Example: Organisation Functional Specification

Graphical representation of MOISE Func. Spec.
## Example: Organisation Normative Specification

<table>
<thead>
<tr>
<th>norm</th>
<th>modality</th>
<th>role</th>
<th>mission / goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>Obl</td>
<td>house_owner</td>
<td>house built</td>
</tr>
<tr>
<td>n2</td>
<td>Obl</td>
<td>site_prep_contractor</td>
<td>site prepared</td>
</tr>
<tr>
<td>n3</td>
<td>Obl</td>
<td>bricklayer</td>
<td>floors laid, walls built</td>
</tr>
<tr>
<td>n4</td>
<td>Obl</td>
<td>roofer</td>
<td>roof built</td>
</tr>
<tr>
<td>n5</td>
<td>Obl</td>
<td>window_fitter</td>
<td>windows fitted</td>
</tr>
<tr>
<td>n6</td>
<td>Obl</td>
<td>door_fitter</td>
<td>doors fitted</td>
</tr>
<tr>
<td>n7</td>
<td>Obl</td>
<td>plumber</td>
<td>plumbing installed</td>
</tr>
<tr>
<td>n8</td>
<td>Obl</td>
<td>electrician</td>
<td>electrical system installed</td>
</tr>
<tr>
<td>n9</td>
<td>Obl</td>
<td>painter</td>
<td>interior painted, exterior painted</td>
</tr>
</tbody>
</table>

Simplified representation of $\text{MOISE}$ Norm. Spec.
Based on Cartago [Ricci et al., 2009b], Jason [Bordini et al., 2007], MOISE [Hübner et al., 2009a] meta-models
A & O Integration

- Instrumenting Organisation Management by dedicated Organisational Artifacts
  - Mapping of the *organisational state* onto artifacts computational state
  - Encapsulation of organisational functionalities by suitably designed artifacts providing *organisational operations*

- Reification of organisation management actions/perceptions by actions/percepts on the artifacts

- Extensible set of organisational artifacts:
  - Openness Management Artifact [Kitio, 2011]
  - Reorganisation Artifact [Sorici, 2011]
  - Evaluation Artifact (kind-of reputation artifact) [Hübner et al., 2009b]
  - Communication management Artifact [Ciortea, 2011]
Exploit the uniform access to artifacts

Agents may be aware of the Organisation by the way of:
- organisational events
- organisational actions

Agents can reason on the organisation:
- to achieve organisational goals
- by developing organisational plans
Example (Adoption of Role)

... 

+!discover_art(ToolName)
  \<- joinWorkspace("HouseBuildingWsp");
  lookupArtifact(ToolName, ToolId);
  focus(ToolId).

+!contract("SitePreparation", GroupBoardId)
  \<- adoptRole(site_prep_contractor)
  focus(GroupBoardId).

+!site_prepared
  \<- ... // actions to prepare the site..
E & O Integration

- Env. Artifacts provide operations on shared resources
- Org. Artifacts provide organisational operations
- Both artifacts bound by count-as, enact constitutive rules [Piunti et al., 2009, de Brito et al., 2012]

→ Org-agnostic agents may indirectly act on the organisation
→ Environment can act on the organisation
→ Organisation is embodied, situated in the environment
Count-as rules [de Brito et al., 2012]

Example

/* If an auction "Art" is finished, its winner ("Winner") plays a role "Role", if it doesn’t adopted it yet */

*auctionStatus(closed)[source(Art)]
count-as
    play(Winner,Role,hsh_group)[source(hsh_group)]
in
    currentWinner(Winner)[source(Art)] &
    not(Winner==no_winner) &
    auction_role(Art,Role).

/* The occurrence of the event "prepareSite" means the achievement of organisational goal "site_prepared" */

+ prepareSite[agent_name(Ag),artifact_name(housegui)]
count-as
    goalState(bhsch,site_prepared,Ag,Ag,satisfied)[source(bhsch)].
Organisation’s dynamics (triggered by Agents, Environment)

**Organisation life-cycle**
- Entrance/Exit of an agent
- Creation/Deletion of an Organisation entity
- Change of Organisation specification

**Structural Organisation life-cycle**
- Creation/Deletion of a group
- Adoption/Release of a role

**Functional Organisation life-cycle**
- Creation/End of a schema
- Commitment/Release of a mission
- Change of a global goal state

**Normative Organisation life-cycle**
- Activation/De-activation of obligation
- Fulfilment/Violation/Sanction
Outcomes of A & E & O Integration

- Normative deliberative agents
  - possibility to define mechanisms for agents to evolve within an organisation/several organisations
  - possibility to define proper mechanisms for deliberating on the internalisation/adopter/violation of norms

- Reorganisation, adaptation of the organisation
  - possibility to define proper mechanisms for diagnosing/evaluating/refining/defining organisations

- “Deliberative” Organisations
  - possibility to define dedicated organisational strategies for the regulation/adaptation of the organisation behaviour (organisational agents)

- “Embodied” Organisation / Organisation Aware Environment
  - possibility to connect organisation to environment
Synthesis: MAOP meta-model

JaCaMo Meta-model [Boissier et al., 2011], based on Cartago [Ricci et al., 2009b], Jason [Bordini et al., 2007], MOISE [Hübner et al., 2009a] meta-models
1 Introduction

2 Multi-Agent Oriented Programming (MAOP)

3 MAOP Perspective: the JaCaMo Platform
   - Overview
   - Project definition
   - Development Tools
   - Technologies

4 MAOP Experiences

5 Conclusions and Perspectives
Agent execution and communication management infrastructures can be:

**Centralised**: all agents in the same machine,
one thread by agent, very fast

**Centralised (pool)**: all agents in the same machine,
fixed number of thread,
allows thousands of agents

**Distributed (jade)**: distributed agents, use of FIPA-ACL using the
Jade agent execution and communication platform

.... others defined by the user (e.g. AgentScape)

Environment execution can be:

**Centralised**: one centralised environment shared by the agents, is automatically included in case of no other specification

**Distributed**: multiple environments shared by the agents – specified by cartago("infrastructure")
Simple way of defining a multi-agent system within the JaCaMo Platform

Example (Building House Definition)

```plaintext
mas house_building {
    agent giacomo // the agent that wants to build a house
    agent companyA // builder agents (see their code for details)
    agent companyB
    agent companyC {
        instances: 5
    }
    agent companyD {
        instances: 13
    }
    agent companyE

    asl-path: src/agt, src/agt/inc
}
```
Eclipse JaCaMo plugin
Agent’s Mind **inspector**

**Inspection of agent orgmajordomo**

- **Beliefs**
  - commitment(italian,mitaly,"jacamoPlan")
  - commitment(french,mFrench,"jacamoPlan")
  - commitment(brazilian1,mBrazil,"jacamoPlan")
  - commitment(brazilian2,mBrazil,"jacamoPlan")
  - current_wsp(obj_1,"server","427dd8d5-408e-431a-a702-7b11ce674e09")
  - formationStatus(ok)
  - goalState("jacamoPlan",greetings[french],[french],satisfied)
  - goalState("jacamoPlan",greetings_uk_done,[italian],[italian],satisfied)
  - goalState("jacamoPlan",greetings_italy_done,[italian],[italian],satisfied)
  - goalState("jacamoPlan",greetings_brazil_done,[brazilian1,brazilian2],[brazilian1,brazilian2],satisfied)
  - goalState("jacamoPlan",greetings_france_done,[french],[french],satisfied)
  - groups(["jacamoTeam"])
  - my_group("jacamoTeam")
  - my_group_id(obj_2)
  - my_ssh("jacamoPlan")
  - my_ssh_id(obj_3)
Structural Specification

Roles

- `greeter` extends `soc`.
- `greetera` extends `greeter`.
- `greeterb` extends `greeter`.
- `greeteri` extends `greeter`.
- `greeterf` extends `greeter`.

Group `jacamoGr`:

Possible roles: `greeterb`, `greeteri`, `greeterf`.

Local links:
- `greeter` has a `communication` link to `greeter` (intra-group, does not extend to subgroups)

Constraint Formation:

- Cardinalities
  - Cardinality of `greeterb` is (2,2)  
  - Cardinality of `greeteri` is (1,1)  
  - Cardinality of `greeterf` is (1,1)
Organization Structure

**jacamoTeam (group)**

- **Formation:**
  - ok

- **Players**
  - brazilian1 plays greeterb
  - brazilian2 plays greeterb
  - french plays greeterf
  - italian plays greeteri

- **Responsible for the following schemes:**
  - jacamoPlan

**History**
Functional Specification

**Scheme jacamoSch**

<table>
<thead>
<tr>
<th>goal</th>
<th>mission</th>
<th>type</th>
<th># ttf</th>
<th>description</th>
<th>arguments</th>
<th>plan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| greetings | mFrench   | achievement | all | greetings from France |
| greetings_france_done | mFrench   | achievement | all | Greetings from France |
| greetings_brazil_done  | mBrazil   | achievement | all | Greetings from Brazil |
| greetings_italy_done   | mItaly    | achievement | all | Greetings from France |
| greetings_uk_done      | mItaly    | achievement | all | Greetings from UK |

**History**

- created: obligation(french,n1,committed(french,mFrench,"jacamoPlan"),1411504910034)
- created: obligation(brazilian1,n2,committed(brazilian1,mBrazil,"jacamoPlan"),14115049)
- created: obligation(brazilian2,n2,committed(brazilian2,mBrazil,"jacamoPlan"),14115049)
- created: obligation(italian,n3,committed(italian,mItaly,"jacamoPlan"),14115049)
- created: obligation(french,n2,committed(french,mFrench,"jacamoPlan",greetings_france_done),achieved)
- created: obligation(brazilian1,ngoal("jacamoPlan",mBrazil,greetings_brazil_done),achieved)
- created: obligation(brazilian2,ngoal("jacamoPlan",mBrazil,greetings_brazil_done),achieved)
- created: obligation(italian,ngoal("jacamoPlan",mItaly,greetings_italy_done),achieved)
- created: obligation(italian,ngoal("jacamoPlan",mItaly,greetings_uk_done),achieved)
### jacamoPlan (scheme instance)

- **Formation:** ok
- **Responsible groups:** jacamoTeam.
- **Players**:
  - brazilian1 committed to mBrazil
  - brazilian2 committed to mBrazil
  - french committed to mFrench
  - italian committed to mItaly

<table>
<thead>
<tr>
<th>goal</th>
<th>state</th>
<th>committed/achieved by</th>
<th>arguments</th>
<th>plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>greetings</td>
<td>satisfied</td>
<td>[french]/[french]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>greetings_france_done</td>
<td>satisfied</td>
<td>[french]/[french]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>greetings_brazil_done</td>
<td>satisfied</td>
<td>[brazilian1,brazilian2]/[brazilian1,brazilian2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>greetings_italy_done</td>
<td>satisfied</td>
<td>[italian]/[italian]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>greetings_uk_done</td>
<td>satisfied</td>
<td>[italian]/[italian]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**History**

- created: obligation(french,n1,committed(french,mFrench,"jacamoPlan"),1411504910034)
- created: obligation(brazilian1,n2,committed(brazilian1,mBrazil,"jacamoPlan"),1411504910034)
- created: obligation(brazilian2,n2,committed(brazilian2,mBrazil,"jacamoPlan"),1411504910034)
- created: obligation(italian,n3,committed(italian,mItaly,"jacamoPlan"),1411504910034)
- created: obligation(french,n4goal("jacamoPlan",mFrench,greetings_france_done),achieved)
- created: obligation(brazilian1,n4goal("jacamoPlan",mBrazil,greetings_brazil_done),achieved)
- created: obligation(italian,n4goal("jacamoPlan",mItaly,greetings_italy_done),achieved)
- created: obligation(italian,n4goal("jacamoPlan",mItaly,greetings_uk_done),achieved)
Integration of Multi-Agent technologies

- **Agent:** Jason agents [Bordini et al., 2007]
- **Environment:** CArtAgO platform [Ricci et al., 2009b]
- **Organisation:** MOISE framework with the extended/refactored version of the MOISE OMI: ORA4MAS [Hübner et al., 2009a]
- **Interaction:** based on tight integration between Jason and KQML or ACL/FIPA

Dimensions are integrated with dedicated bridges:

- **A–E** (c4Jason, c4Jadex [Ricci et al., 2009b])
- **E–O** (count-as/enact rules [Piunti et al., 2009])
- **A–O** is for free (thanks to ORA4MAS). Strategies and reasoning capabilities from $J^+\text{MOISE}$ [Hübner et al., 2007] can be reused.

**Open to integrate other Multi-Agent Technologies**
Integration with other technologies

- **Web 2.0**
  - implementing Web 2.0 applications

- **Android Platforms**
  - implementing mobile computing applications on top of the Android platform

- **Web Services**
  - building SOA/Web Services applications
  - [http://cartagows.sourceforge.net](http://cartagows.sourceforge.net)

- **Arduino Platforms**
  - building “Web of Things” Applications
  - [http://jacamo.sourceforge.net](http://jacamo.sourceforge.net)

- **Semantic Technologies**
  - JaSA: Semantically Aware Agents
  - [http://cartago.sourceforge.net](http://cartago.sourceforge.net)
Outline

1. Introduction
2. Multi-Agent Oriented Programming (MAOP)
3. MAOP Perspective: the JaCaMo Platform
4. MAOP Experiences
   - MAOP Case: Agile Governance of M2M Infrastructure
   - Sharing data, Knowledge
   - Privacy & Trust
5. Conclusions and Perspectives
European Telecommunications Standards Institute (ETSI) view on M2M infrastructure

- **Device Domain**: smart devices (sensors and actuators) for collecting data and controlling the environment
- **Network Domain**: shared communication infrastructure (platforms and gateways) to connect applications to devices
- **Application Domain**: applications providing ubiquitous & added value services to citizens
In the context of this industrial project, our objective is to define an agile governance of the Urban M2M Infrastructure.

This is an ongoing research project in collaboration with OrangeLabs, France.

[http://www.senscity-grenoble.com](http://www.senscity-grenoble.com)
To get a urban M2M infrastructure for smart city:

- where costs and resources can be shared between several applications (e.g. Parking Management, Garbage Collection, Smart Metering, etc)
  - Shifting from “vertical” to “horizontal” M2M infrastructures
- where new Stakeholders (i.e. application/sensors/actuators) can be added/suppressed during the lifetime of the system
  - Openness
- and reacting to the changes of environmental conditions (e.g. increase in the number of collected data, number of messages)
  - Adaptation
- Definition of an agile and decentralized governance layer on top of the M2M Infrastructure [Persson et al., 2012]
SensCity platform

- composed of 47 types of components
- supporting access to physical devices
- and shared by several heterogenous Applications
MA Governance Layer: Governance Artifacts

- Monitoring and controlling tools that
- Compute direct / interpreted / aggregated status of the SensCity Platform’s components. Provide a uniform access of this status to the agents by observable properties or signals
- Offer the agents the possibility to act on the functioning of the M2M infrastructure by a repertoire of operations (Threshold definition, LoadBalancing, (De)activate, ...)
- Structured according to the ETSI domains
MA Governance Layer: Governance Agents

- Take local autonomous decisions given the prescriptions of the organisation and their own local strategies.
- Monitor the M2M infrastructure by focusing on artifacts (e.g. Failures, Overloads) updating their beliefs, goals.
- Apply their governance policies and local strategies (goals, beliefs, plans) to control M2M infrastructure by using their actions (\(\sim\) artifacts operations).
- Interact with the other agents.
- Adapt the global governance strategy (\(\sim\) may modify the organisation).
MA Governance Layer: Governance Organisations

- Expressed in terms of structure (groups, roles), functioning (missions, goals, plans) and norms, and reasoned on by the agents and monitored by the organisation management infrastructure.

- Horizontal Organisation: defines the nominal global functioning based on ETSI standards (Domains, Service Capabilities, Capability functionalities, ...).
MA Governance Layer: Governance Organisations

- Expressed in terms of structure (groups, roles), functioning (missions, goals, plans) and norms, and reasoned on by the agents and monitored by the organisation management infrastructure.

- Horizontal Organisation: defines the nominal global functioning based on ETSI standards (Domains, Service Capabilities, Capability functionalities, ...)

- Vertical Organisations: based on each application SLA deployed on the Urban M2M Infrastructure.

agents participate to multiple organisations.
SLA Definition

- **Service Level Agreement:** Translation of Application subscription to Devices in terms of Group, Roles, Missions, Norms
- **Validation:** If judged feasible, role adopted Else Rejected or New proposal
Contract Execution

- **Social Scheme activation:** when request is received, new scheme instance started
- **Execution Monitoring:** validation of application requests, monitoring of devices’ activity
M2M Agile Governance Example

Problem Detection
- Detection of possible norm violation, i.e. failure to comply with the SLA
- Platform Monitoring: find the source of the problem with the help of the artifacts
M2M Agile Governance Example

Infrastructure Adaptation

- Fix the problem by acting on the platform via the artifacts under their responsibility and their governance policies
- Coordinate with each other
Governance Strategy Adaptation

- Problem with the SLA: SLA might be too greedy, infrastructure adaptation not sufficient

~ SLA redefinition (eg. decrease message frequency, device subscription redefined)
Synthesis

- Monitoring and governance of the M2M infrastructure take place at different levels embracing an increasing broader view: Artifacts < Agents & Interactions < Organisations

~ Modularity / Visibility of the Governance Layer

- Coordination mechanisms installing Top-Down - Bottom-up loops (Synergie between Macro/Micro levels)

- Agents are in charge of the governance but also of the reorganisation process

- Current experiments:
  - Installing complex governance processes
  - Refactoring the Agents layer by making explicit dedicated coordination strategies expressed into coordination artifacts
  - Distributing and better managing the deployment
Smart Places

- Smart places where nomadic users participate to adhoc virtual communities (in collaboration with LaHC/UJM, LT2C/UJM-TSE)
Privacy preservation in open and decentralized communities by the definition of Privacy Enforcing Agent:

- Privacy Enforcement Norms checked by the agent Privacy Enforcement Layer
- Appropriateness laws (A-Laws) used by the agent Privacy violation detection layer
Open Innovation Communities where groups of individuals
- share common interests or objectives
- interact and share resources (documents, ideas, etc)

Management of Individual and Collective Trust Policies on behalf of the users [Yaich et al., 2012]
Decentralized Trust Management

Multi-Agent Architecture
Outline

1. Introduction
2. Multi-Agent Oriented Programming (MAOP)
3. MAOP Perspective: the JaCaMo Platform
4. MAOP Experiences
5. Conclusions and Perspectives
Conclusions

MAOP proposes a seamless integration of different abstractions that brings interesting features to Intelligent Environments:

〜 separation of concerns
   - using the best abstraction level and tools to tackle the specific dimensions, avoiding design pitfalls, such as using agents to implement either non-autonomous entities (e.g., a blackboard agent) or a collection of autonomous entities (group agent)

〜 openness and heterogeneity
   - E.g., heterogeneous agents working in the same organisation, heterogeneous agents working in the same environment, the same agent working in different and heterogeneous organisations, the same agent working in different heterogeneous environments

〜 programming features:
   - Each of the dimension can be addressed explicitly
   - Modularity, extensibility, reusability is possible
   - Extensible set of actions and tools provided to Agents (Reorganization Artifact, Organization Management Artifacts, ...)
   - ... code is cleaner and more understandable ...
Coordination
- Integration of Bottom-up AND Top-Down functioning within MAS ~⇒ integration of emergence AND Normative dynamics
- Management of Open Organisations, Multiple Organisations, Organisation of Organisations
- Management of Situated Organisations (Interactions between E and O dimensions)
- Shift from MAS to MAOS (Multi-AgentOrganization Systems)

Engineering
- Debugging, Performance, ...
- Life cycle of MAS (from requirement to maintenance) ~⇒ software engineering tools and methods
- Shift from Agent-Oriented Sofware Engineering to Multi-Agent Oriented Software Engineering where all the dimensions A, E, I, O may guide each step of the process
- Evaluation & Verification of MAO programmed applications,
- Integrating with other technologies
- Handle Scalability, Robustness
Multi-Agent Oriented Programming
– Introduction –
The JaCaMo Platform

O. Boissier\textsuperscript{1} \quad R.H. Bordini\textsuperscript{2} \quad J.F. Hübner\textsuperscript{3} \quad A. Ricci\textsuperscript{4}

1. Ecole Nationale Supérieure des Mines (ENSMSE), Saint Etienne, France

2 Pontificia Universidade Catolica do Rio Grande do Sul (PUCRS), Porto Alegre, Brazil

3. Federal University of Santa Catarina (UFSC), Florianópolis, Brazil

4. University of Bologna (UNIBO), Bologna, Italy

September 2015
Contrôle et coordination orientés multi-agents.
Habilitation à diriger des recherches, ENS Mines Saint-Etienne et Université Jean Monnet.

Multi-agent oriented programming with jacamo.
*Science of Computer Programming*, pages –.

*Programming Multi-Agent Systems in AgentSpeak using Jason*.

Formalising situatedness and adaptation in electronic institutions.

Modeling relationships for privacy preservation in virtual communities.
In Burdescu, D. D., Akerkar, R., and Badica, C., editors, WIMS, page 8. ACM.

Programming institutional facts in multi-agent systems.
In COIN-12, Proceedings.

From interactions to collective behaviour in agent-based systems.

Steps towards multi-agent oriented programming.
(slides Workshop) 1st International Workshop on Multi-Agent Systems, IWMAS’97, Boston.
Instrumenting Multi-Agent Organisations with Organisational Artifacts and Agents.
*Journal of Autonomous Agents and Multi-Agent Systems.*

Developing Organised Multi-Agent Systems Using the MOISE+ Model: Programming Issues at the System and Agent Levels.

Instrumenting Multi-Agent Organisations with reputation artifacts.

Interaction-oriented programming.


MASK: An AEIO toolbox to design and build multi-agent systems.

In et al., C., editor, Knowledge Engineering and Agent Technology, IOS Series on Frontiers in AI and Applications. IOS press, Amsterdam.


Artifacts in the A&A meta-model for multi-agent systems.


A jacamo-based governance of machine-to-machine systems.

Bibliography V

Embodying organisations in multi-agent work environments.
In IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2009), Milan, Italy.

Toward team-oriented programming.


Environment programming in multi-agent systems: an artifact-based perspective.
Environment programming in CArtAgO.

VOLCANO: a vowels-oriented multi-agent platform.

Agent-oriented programming.

Agile governance in an ambient intelligence environment.