Multi-Agent Oriented Programming
– Introduction –
The JaCaMo Platform

O. Boissier, R.H. Bordini, J.F. Hübner, A. Ricci

May 2014

Outline

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

MAS Concepts

1. BELIEFS
2. GOALS
3. PLANS
4. PERCEPTIONS
5. ACTIONS
6. EVENTS
7. AGENTS
8. MISSIONS
9. ROLES
10. DEONTIC RELATIONS
11. GROUPS
12. NORMS
13. SANCTIONS
14. REWARDS
15. ORGANISATIONS
16. RESOURCES
17. SERVICES
18. OBJECTS
19. ENVIRONMENTS
20. COMMUNICATION LANGUAGES
21. INTERACTIONS
22. TOPOLOGY
23. TOLERANCE
24. CHALLENGES

Challenges

• Numerous concepts, models and technologies!!!

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-Étienne, France (gauthier.picard@emse.fr)
- M. Hannoun, B. Gâteau, G. Danoy, R. Kitio, C. Persson, R. Yaich, ENS Mines St-Étienne, 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
MAS Dynamics & Coordinations

- Coordination may be programmed into one or several of these dimensions [Boissier, 2003]
- Agent / Environment / Interaction / Organisation Centred coordination depends of the purpose of the domain/problem

Challenges
- Numerous concepts, models and technologies!!!
- Rich scope of possible dynamics and coordinations!!!

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]

Multi-Agent Oriented Programming Platform have been/are proposed:
- 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], ...
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
3. MAOP Perspective: the JaCaMo Platform
4. MAOP Experiences
5. Conclusions and Perspectives

Introduction

MAOP

JaCaMo

Experiences

Conclusion

MAOP Meta-Model

A

E

O

Agent meta-model

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

Agent example I

Example (Giacomo Agent Code)

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

Example (companyX Agent Code)

```java
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
```
Introduction MAOP JaCaMo Experiences Conclusion

Agent & Agent Interaction meta-model

![Diagram of Agent & Agent Interaction meta-model]

Agent's dynamics

![Diagram of Agent's dynamics]

Environment meta-model

![Diagram of Environment meta-model]

Auction Artifact

![Diagram of Auction Artifact]

Example

```java
public class AuctionArt extends Artifact {
    @OPERATION void init(String taskDs, int maxValue) {
        defineObsProperty("task", taskDs); // task description
        defineObsProperty("maxValue", maxValue); // max. value
        defineObsProperty("currentBid", maxValue); // current best bid (lower service price)
        defineObsProperty("currentWinner", "no_winner");
    }

    // places a new bid for doing the service for price p
    @OPERATION void bid(double bidValue) {
        ObsProperty opCurrentValue = getObsProperty("currentBid");
        ObsProperty opCurrentWinner = getObsProperty("currentWinner");
        if (bidValue < opCurrentValue.intValue()) {
            opCurrentValue.updateValue(bidValue);
            opCurrentWinner.updateValue(getOpUserName());
        }
    }
}
```
**Introduction**

**MAOP**

**JaCaMo**

**Experiences**

**Conclusion**

---

**MAOP Meta-Model**

**A**

**E**

**O**

---

### Giacomo Agent Code I

```prolog
!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).
```

---

### Giacomo Agent Code II

```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
  <- print("Waiting the bids for 5 seconds...");
  .wait(5000); // use intern deadline of 5 sec to close auctions
+!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)
  }.
```

---

### companyA Agent Code I

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

---

```prolog
/* 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 */
```
Introduction

MAOP

JaCaMo

Experiences

Conclusion

MAOP Meta-Model

A

E

O

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

Outcomes of A & E Integration

- 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

Example: Organisation Structural Specification

Simplified MOISE meta-model [Hübner et al., 2009a]

Graphical representation of MOISE Struct. Spec.
Introduction

MAOP

JaCaMo

Experiences

Conclusion

MAOP Meta-Model

A

E

O

Example: Organisation Functional Specification

Graphical representation of MAISE Func. Spec.

Example: Organisation Normative Specification

Simplified representation of MAISE Norm. Spec.

A & E & O Interaction meta-model

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]
A & O Integration (2)

- 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)

```plaintext
+!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 [Piantiti 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]

```plaintext
/* 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/adoptions/violations 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

A MAOP meta-model

JaCaMo Platform [Boissier et al., 2011], based on Cartago [Ricci et al., 2009b], Jason [Bordini et al., 2007], MOISE [Hübner et al., 2009a] meta-models
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 OM: 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-MOISE+ [Hübner et al., 2007] can be reused.

**Open to integrate other Multi-Agent 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

Arduino Platforms
- building “Web of Things” Applications

Semantic Technologies
- JaSA: Semantically Aware Agents

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
- **Arduino Platforms**
  - building “Web of Things” Applications
- **Semantic Technologies**
  - JaSA: Semantically Aware Agents

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

M2M Infrastructure for Smart Cities (ETSI view)

- **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.

To get a urban M2M infrastructure for smart cities:

- 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 heterogeneous Applications
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 (→ artifacts operations)
- Interact with the other agents
- Adapt the global governance strategy (→ 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, ...)

M2M Agile Governance Example

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

- Vertical Organisations: based on each application SLA deployed on the Urban M2M Infrastructure agents participate to multiple organisations
M2M Agile Governance Example

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

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

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

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)
Monitoring and governance of the M2M infrastructure take place at different levels embracing an increasing broader view:

- **Artifacts < Agents & Interactions < Organisations**

~ Modularity / Lisibleity 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 deployement

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

Decentralized Trust Management

- 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

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 ...

Outline

- Introduction
- Multi-Agent Oriented Programming (MAOP)
- MAOP Perspective: the JaCaMo Platform
- MAOP Experiences

Conclusions and Perspectives

MAOP Open Issues & Perspectives

\[ \text{Co} \]
- 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 dimension)

\[ \text{En} \]
- Shift from MAS to MAOS (Multi-AgentOrganization Systems)

\[ \text{Eng} \]
- Debugging, Performance, ...
- Life cycle of MAS (from requirement to maintenance)
- software engineering tools and methods
- Shift from Agent-Oriented Software 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, R.H. Bordini, J.F. Hübner, A. Ricci

May 2014

Bibliography


