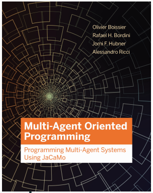
Multi-Agent Coordination



Olivier Boissier
Rafael H. Bordini
Jomi F. Hubner
Alessandro Ricci

Multi-Agent Oriented Programming

Programming Multi-Agent Systems Using JaCaMo

Introduction to Multi-Agent Systems and Multi-Agent Oriented Programming

Olivier Boissier, Luis Gustavo Nardin
Mines Saint-Etienne IMT – LIMOS UMR 6158 CNRS

CPS2 M2 – Fall 2024

Adapted version of Tutorial given at PFIA 2023 – Strasbourg

Table of Contents

1. Overview of Multi-Agent Oriented Programming
2. Autonomous Agents Situated in an Environment
3. Situated Autonomous Agents in Interaction
4. Coordinating and Regulating Autonomous Agents
5. Wrap-up and Next Steps
6. References

Overview of Multi-Agent Oriented Programming

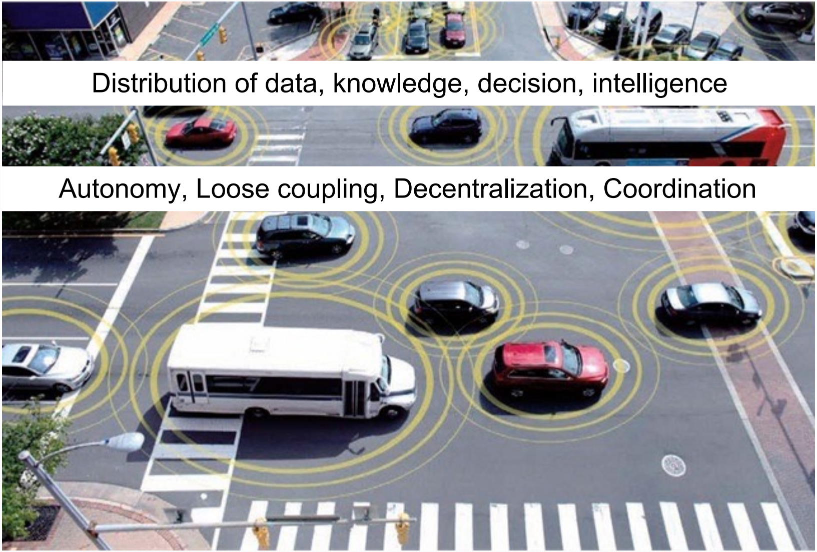
Motivations – Requirements

- Complex system are systems composed of many components which may interact with each otherÂăand present non-trivial relationships between cause and effect
  - each effect ⇝ multiple causes
  - each cause ⇝ multiple effects
  - feedback loops
  - non-linear cause-effect chains
- Complex cyber-physical social systems
  - Smart cities
  - Smart grids
  - Manufacturing
  - Mobility systems

Motivations – Requirements

Distribution of data, knowledge, decision, intelligence

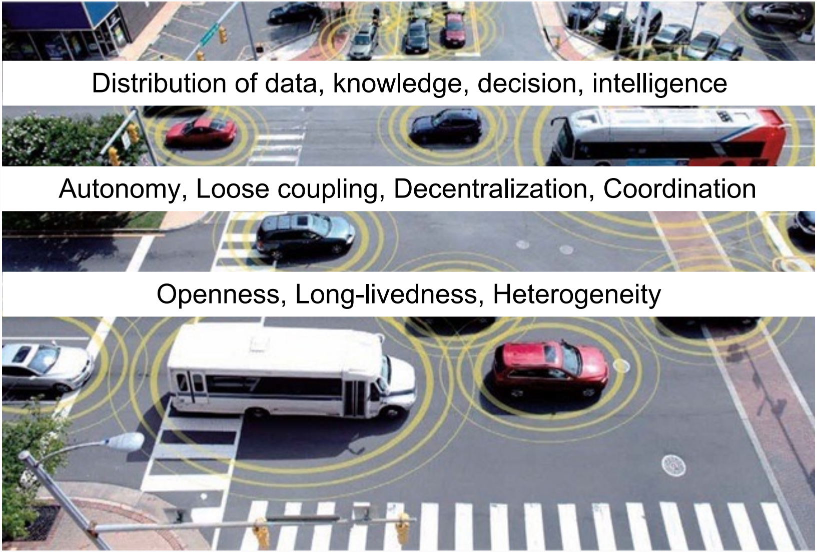
Motivations – Requirements



Distribution of data, knowledge, decision, intelligence

Autonomy, Loose coupling, Decentralization, Coordination

Motivations – Requirements

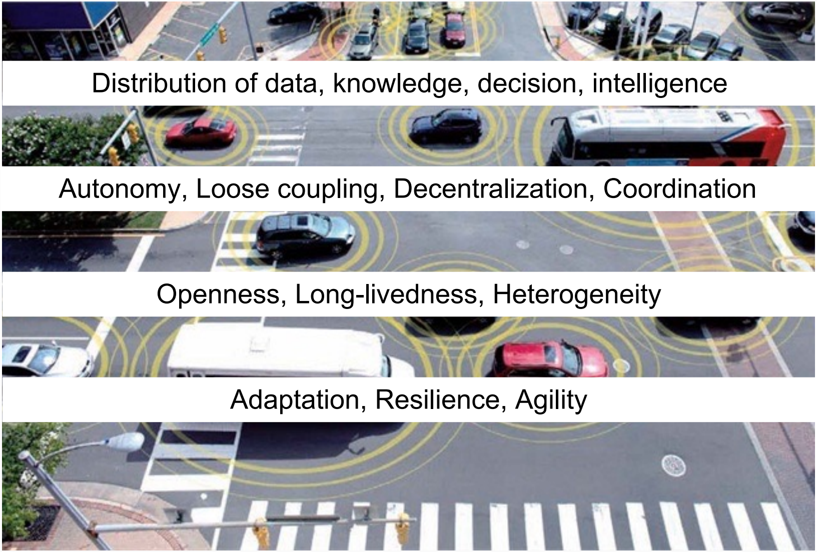


Distribution of data, knowledge, decision, intelligence

Autonomy, Loose coupling, Decentralization, Coordination

Openness, Long-livedness, Heterogeneity

Motivations – Requirements



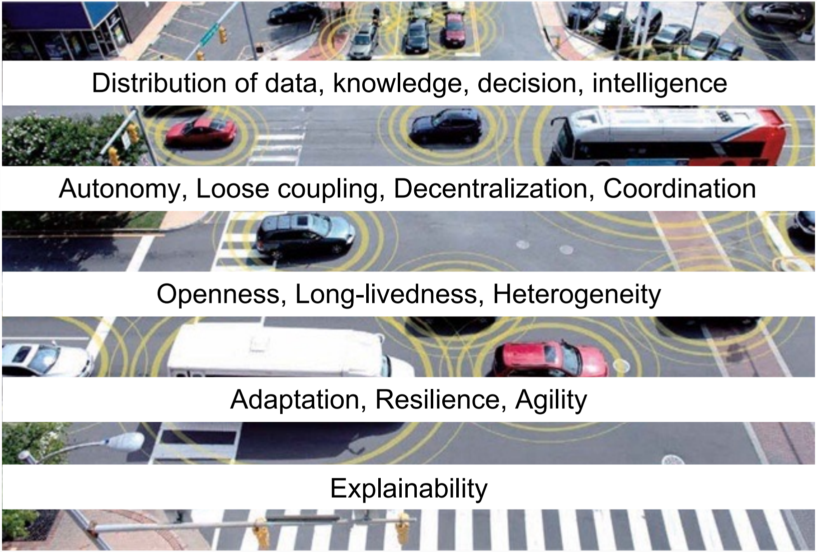
Distribution of data, knowledge, decision, intelligence

Autonomy, Loose coupling, Decentralization, Coordination

Openness, Long-livedness, Heterogeneity

Adaptation, Resilience, Agility

Motivations – Requirements



Distribution of data, knowledge, decision, intelligence

Autonomy, Loose coupling, Decentralization, Coordination

Openness, Long-livedness, Heterogeneity

Adaptation, Resilience, Agility

Explainability

Approaches

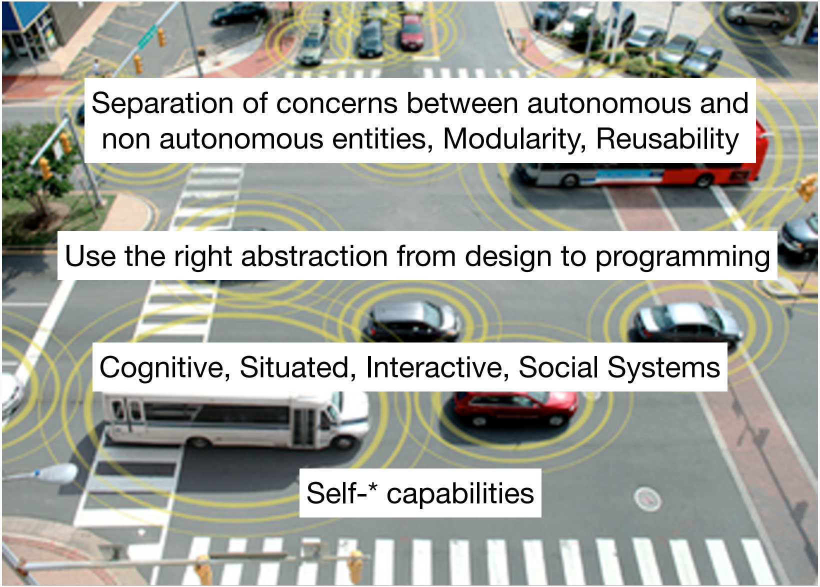
Multi-Agent Based Simulation: models are used to describe and simulate existing complex systems, either natural or artificial, to analyze their properties.

- Local representations of different points of view, decisions, goals motivations, behaviors, etc.
- Interaction between local strategies, behaviors and global and common strategies of control
- Continuous operation and evolution
- Solution is the result of interaction between local processes

Multi-Agent Based System Engineering: models are used to the design and development of systems and applications.

- No monolithic vision
- Multi-* (sites, expertise, domains, points of view, decisions, goals, motivations, ...)
- Incremental and collaborative development
- Continuous execution, adaptation
- Increasingly user-centric

Motivations – Engineering



Separation of concerns between autonomous and non autonomous entities, Modularity, Reusability

Use the right abstraction from design to programming

Cognitive, Situated, Interactive, Social Systems

Self-* capabilities

Multi-Agent System

A set of autonomous agents interacting with each other within a shared environment, eventually under one to multiple organisations

- Agents: autonomous decision-making entities able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them ⇝ (soft/hard)ware, (coarse/fine)-grain, (hetero/homo)geneous,
- Environment: shared medium providing the surrounding conditions for agents to exist and act ⇝ virtual/physical, passive/active, deterministic or not, ...

  e.g. communication and coordination infrastructure, topology of spatial domain, support of an action model
- Interaction: motor of dynamics and interoperability in the MAS ⇝ direct communicative / indirect actions through the environment
- Organisation: abstractions to declare and make accessible to agents their collective structure and functioning in a shared environment ⇝ pre-defined/emergent, static/adaptive, open/closed, ...

  e.g. coordination and regulation activities

Multi-Agent System

A set of autonomous agents interacting with each other within a shared environment, eventually under one to multiple organisations

- Agents: autonomous decision-making entities able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them ⇝ (soft/hard)ware, (coarse/fine)-grain, (hetero/homo)geneous,
- Environment: shared medium providing the surrounding conditions for agents to exist and act ⇝ virtual/physical, passive/active, deterministic or not, ...

  e.g. communication and coordination infrastructure, topology of spatial domain, support of an action model
- Interaction: motor of dynamics and interoperability in the MAS ⇝ direct communicative / indirect actions through the environment
- Organisation: abstractions to declare and make accessible to agents their collective structure and functioning in a shared environment ⇝ pre-defined/emergent, static/adaptive, open/closed, ...

  e.g. coordination and regulation activities

Multi-Agent System

A set of autonomous agents interacting with each other within a shared environment, eventually under one to multiple organisations

- Agents: autonomous decision-making entities able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them ⇝ (soft/hard)ware, (coarse/fine)-grain, (hetero/homo)geneous,
- Environment: shared medium providing the surrounding conditions for agents to exist and act ⇝ virtual/physical, passive/active, deterministic or not, ...

  e.g. communication and coordination infrastructure, topology of spatial domain, support of an action model
- Interaction: motor of dynamics and interoperability in the MAS ⇝ direct communicative / indirect actions through the environment
- Organisation: abstractions to declare and make accessible to agents their collective structure and functioning in a shared environment ⇝ pre-defined/emergent, static/adaptive, open/closed, ...

  e.g. coordination and regulation activities

Multi-Agent System

A set of autonomous agents interacting with each other within a shared environment, eventually under one to multiple organisations

- Agents: autonomous decision-making entities able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them ⇝ (soft/hard)ware, (coarse/fine)-grain, (hetero/homo)geneous,
- Environment: shared medium providing the surrounding conditions for agents to exist and act ⇝ virtual/physical, passive/active, deterministic or not, ...

  e.g. communication and coordination infrastructure, topology of spatial domain, support of an action model
- Interaction: motor of dynamics and interoperability in the MAS ⇝ direct communicative / indirect actions through the environment
- Organisation: abstractions to declare and make accessible to agents their collective structure and functioning in a shared environment ⇝ pre-defined/emergent, static/adaptive, open/closed, ...

  e.g. coordination and regulation activities

Multi-Agent System

A set of autonomous agents interacting with each other within a shared environment, eventually under one to multiple organisations

- Agents: autonomous decision-making entities able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them ⇝ (soft/hard)ware, (coarse/fine)-grain, (hetero/homo)geneous,
- Environment: shared medium providing the surrounding conditions for agents to exist and act ⇝ virtual/physical, passive/active, deterministic or not, ...

  e.g. communication and coordination infrastructure, topology of spatial domain, support of an action model
- Interaction: motor of dynamics and interoperability in the MAS ⇝ direct communicative / indirect actions through the environment
- Organisation: abstractions to declare and make accessible to agents their collective structure and functioning in a shared environment ⇝ pre-defined/emergent, static/adaptive, open/closed, ...

  e.g. coordination and regulation activities

Multi-Agent System

A set of autonomous agents interacting with each other within a shared environment, eventually under one to multiple organisations

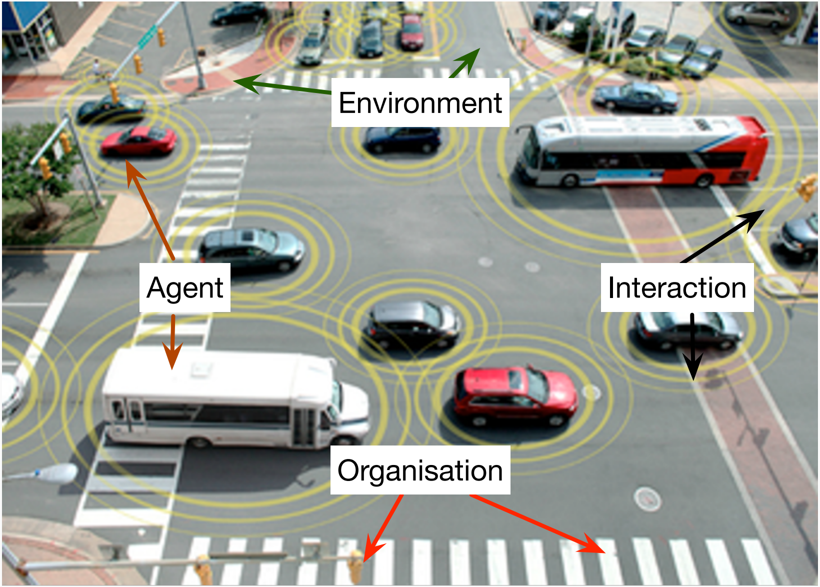
A Multi-Agent System **is more than** a simple set of agents

- Agents: autonomous decision-making entities able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them ⇝ (soft/hard)ware, (coarse/fine)-grain, (hetero/homo)geneous,
- Environment: shared medium providing the surrounding conditions for agents to exist and act ⇝ virtual/physical, passive/active, deterministic or not, ...

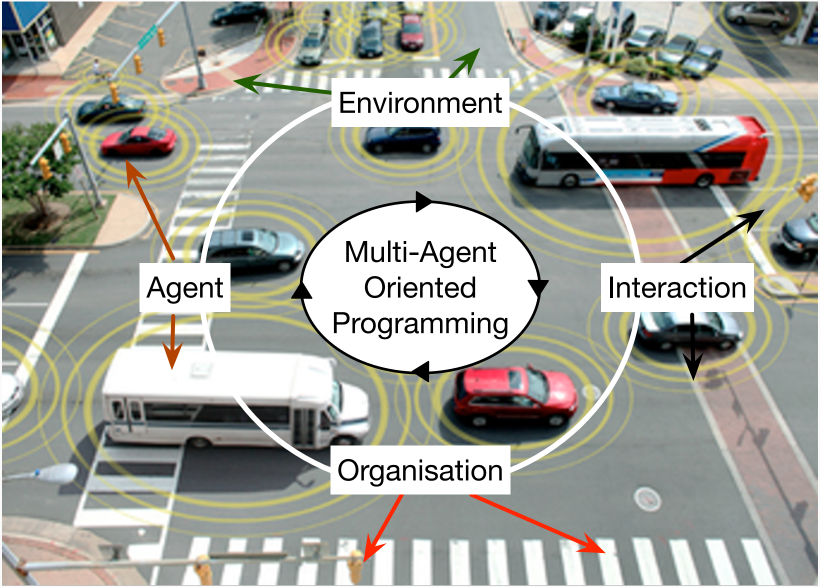
  e.g. communication and coordination infrastructure, topology of spatial domain, support of an action model
- Interaction: motor of dynamics and interoperability in the MAS ⇝ direct communicative / indirect actions through the environment
- Organisation: abstractions to declare and make accessible to agents their collective structure and functioning in a shared environment ⇝ pre-defined/emergent, static/adaptive, open/closed, ...

  e.g. coordination and regulation activities

Multi-Agent Oriented Programming



Multi-Agent Oriented Programming



Multi-Agent Oriented Programming (MAOP)

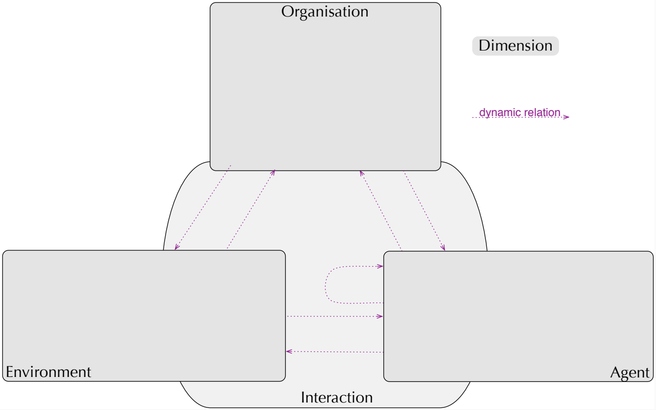
MAOP aims at engineering systems:

- as organisation of autonomous agents in interaction with each other within a shared environment,
- by keeping alive, from design to execution, concepts pertaining to each of the Agent / Environment / Interaction / Organisation dimensions as well as their control/life cycles.

Note: inspired by the VOWELS' perspective [Dem95]

⇝ Going beyond each of the AOP [Sho93], EOP [RPV10], IOP [Huh01], OOP [PTCC99] programming approaches

JaCaMo Meta-model – Multi-Agent Concepts

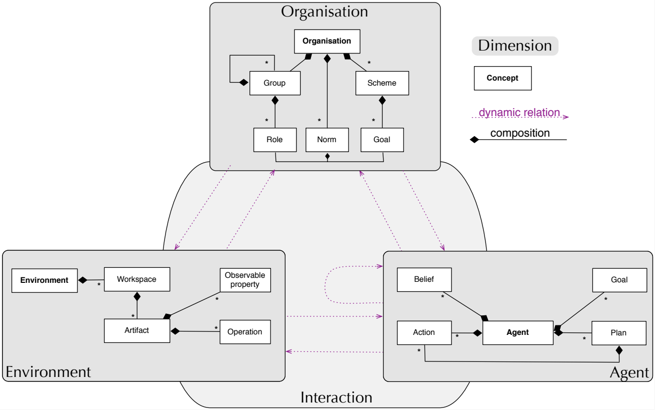


Simplified view of the whole meta-model [BBHR20, BBH+11]

seamlessly integrating the four dimensions based on Jason [BHW07b],

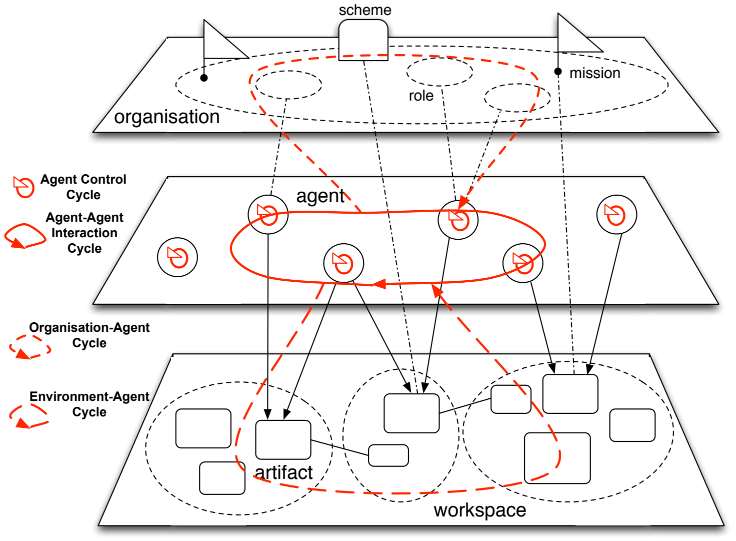
Cartago [RPVO09b], Moise [HBKR10] meta-models

JaCaMo Meta-model – Multi-Agent Concepts



Simplified view of the whole meta-model [BBHR20, BBH+11] seamlessly integrating the four dimensions based on Jason [BHW07b], Cartago [RPVO09b], Moise [HBKR10] meta-models

JaCaMo Meta-model – Multi-Agent Dynamics



Overview of Multi-Agent Oriented Programming

Practice

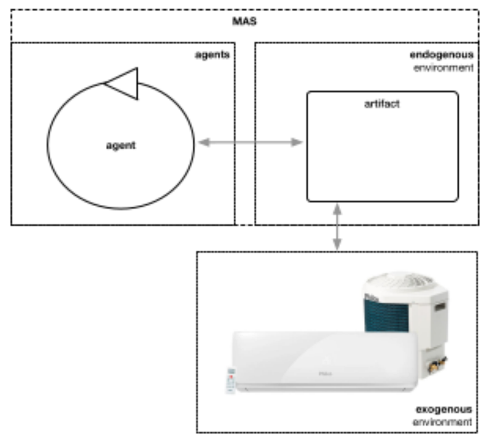
Smart Room Scenario

Develop one room controller agent to manage a "Heating, Ventilating and Air Conditioning" (HVAC) device to reach a desired temperature based on agents' preferences acting on behalf of users

Separation of concerns

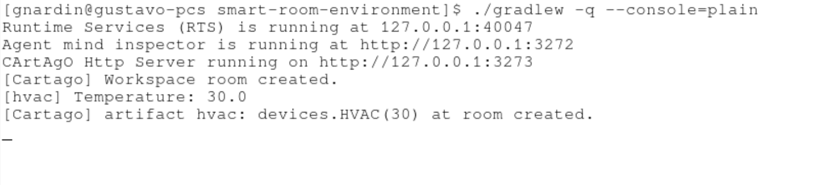
- integration and interoperability with the HVAC
    - **environment** modeling
- strategy to keep the right temperature
    - **agent** modeling

Smart Room Scenario



JaCaMo Application on Local Computer Instructions

1. Enter the directory `smart-room-environment`
   - `cd smart-room-environment`
2. Run the JaCaMo application
   - `./gradlew -q --console=plain`

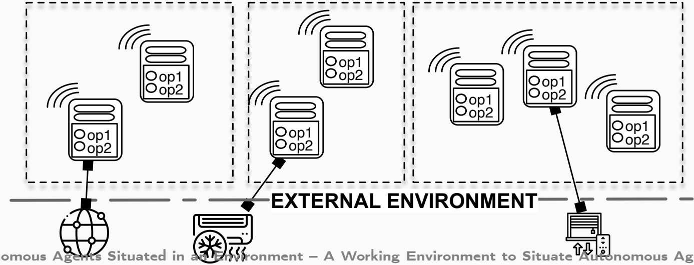


```
[gnardin@gustavo-pcs smart-room-environment]$ ./gradlew -q --console=plain
Runtime Services (RTS) is running at 127.0.0.1:40047
Agent mind inspector is running at http://127.0.0.1:3272
CArtAgO Http Server running on http://127.0.0.1:3273
[Cartago] Workspace room created.
[hvac] Temperature: 30.0
[Cartago] artifact hvac: devices.HVAC(30) at room created.
_
```

Require Java OpenJDK 17

Autonomous Agents Situated in an Environment

An Environment to Situate Autonomous Agents



Basic Concepts

- Environment: shared medium providing the surrounding conditions for resources to be exposed, for agents to execute, interact, observe and act on resources.

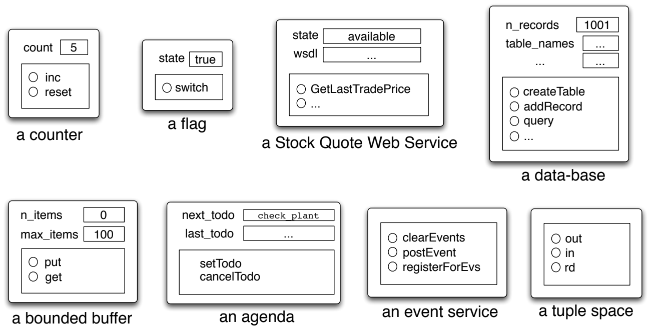
- Workspace: topological or symbolic container of artifacts and agents situated in the environment. All interactions between agents and artifacts happen in the context of a workspace whether or not explicit.

- Artifact: a real or conceptual environment resource that can be dynamically constructed, shared and used by other agents to support their activities. An artifact is a non-autonomous, function-oriented and stateful entity, exposing:
  - *operations* that agents can use to execute actions,
  - *properties* that agents can observe to acquire beliefs,
  - *signals* that agents can perceive while using or observing the artifact.

⇝ a uniform interface to heterogeneous set of resources

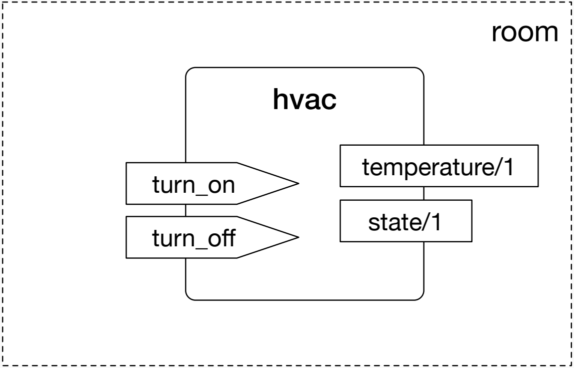
- Workspace and Artifact are not autonomous nor proactive.

A World of Artifacts

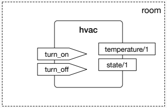


- Individual or personal artifacts: functionalities for a single agent use (e.g. agenda, library)
- Social artifacts: functionalities for structuring and managing the interaction (e.g. a blackboard, a game-board)
- Boundary artifacts: access external resources/services/IoT (e.g. a printer, a Web Service) or to represent devices enabling I/O with users (e.g GUI, console)

Environment Programming (in JaCaMo)



Environment Programming (in JaCaMo)



```
public class HVAC extends Artifact {
  private double temperature;

  void init(double temp){
    this.temperature = temp; // initial simulated value
    defineObsProperty("state","off");
    defineObsProperty("temperature",temperature);
    ...
  }
  @OPERATION void turn_on() {
    if (getObsProperty("state").stringValue().equals("off")) {
      getObsProperty("state").updateValue("on");
      this.execInternalOp("updateTemperatureProc",-1);
      log("HVAC on");
    }
  }
  @OPERATION void turn_off() {
    ...
  }
  @INTERNAL_OPERATION void updateTemperatureProc(double step){
    ObsProperty prop = getObsProperty("temperature");
    ObsProperty state = getObsProperty("state");
    while (!state.stringValue().equals("off")) {
      temp.updateValue(temp.doubleValue() + step);
      log("Temperature: " + temp.doubleValue());
      this.await_time(300);
    }
  }
}
```

Environment Dynamics

In the context of Environment life-cycle:

- Creation/Deletion of Workspaces

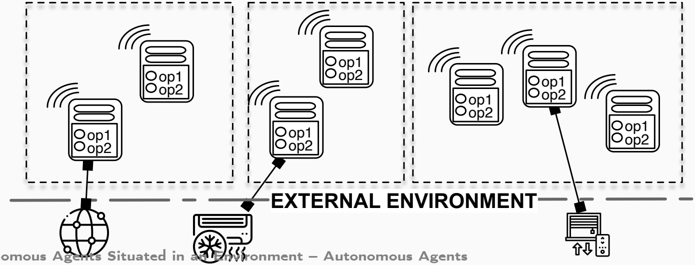
In the context of Workspace life-cycle:

- Creation/Deletion of Artifacts
- Creation/Deletion & Entry/Exit of Agents

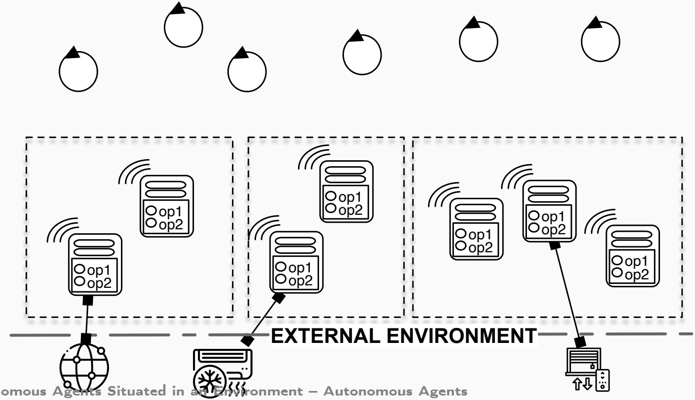
In the context of Artifact life-cycle:

- Atomic execution, Transactionality, Success/Failure, Activation/Deactivation of an operation
- Creation/Deletion/Update of Observable Properties
- Linking/Unlinking with other artifacts

Autonomous Agents situated in an Environment



Autonomous Agents situated in an Environment



Some literature on Agents

- Books: [BDDFS05, BDDFS09]
- Proceedings: ProMAS, DALT, LADS, EMAS, AGERE, ...
- Surveys: [BBD^{+}06, FBHT07] ...
- Languages of historical importance: Agent0 [Sho93], AgentSpeak(L) [Rao96], MetateM [Fis05], 3APL [HdBvdHM97], Golog [GLL00]
- Other prominent languages: *Jason* [BHW07a], Jadex [PBL05], 2APL [Das08], GOAL [Hin09], JACK [Win05], JIAC, AgentFactory
- But many other languages and platforms...

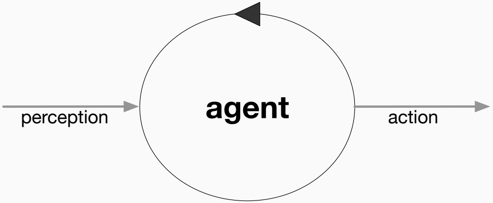
Some Languages and Platforms

Jason (Hübner, Bordini, ...); 3APL and 2APL (Dastani, van Riemsdijk, Meyer, Hindriks, ...); Jadex (Braubach, Pokahr); MetateM (Fisher, Guidini, Hirsch, ...); ConGoLog (Lesperance, Levesque, ... / Boutilier – DTGolog); Teamcore/ MTDP (Milind Tambe, ...); IMPACT (Subrahmanian, Kraus, Dix, Eiter); CLAIM (Amal El Fallah-Seghrouchni, ...); GOAL (Hindriks); BRAHMS (Sierhuis, ...); SemantiCore (Blois, ...); STAPLE (Kumar, Cohen, Huber); Go! (Clark, McCabe); Bach (John Lloyd, ...); MINERVA (Leite, ...); SOCS (Torroni, Stathis, Toni, ...); FLUX (Thielscher); JIAC (Hirsch, ...); JADE (Agostino Poggi, ...); JACK (AOS); Agentis (Agentis Software); Jackdaw (Calico Jack); *simpAL*, *ALOO* (Ricci, ...);

...

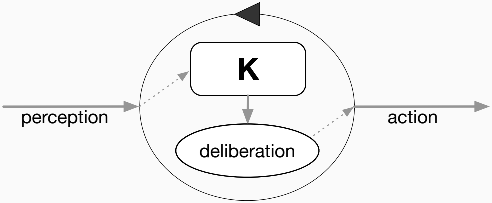
Autonomous Agent

Autonomous decision-making entity able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them



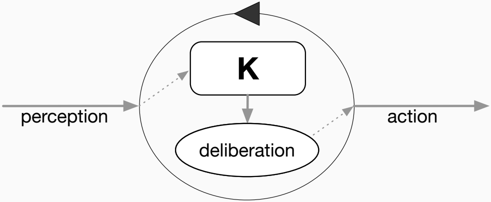
Autonomous Agent

Autonomous decision-making entity able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them



Autonomous Agent

Autonomous decision-making entity able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them

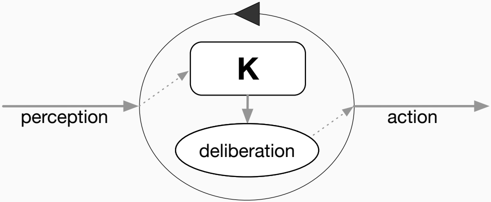


[reasoning cycle]

```
while true do
    $K \leftarrow K \pm perception()$
    $P \leftarrow deliberation(K)$
    $act(P)$
```

Autonomous Agent

Autonomous decision-making entity able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them

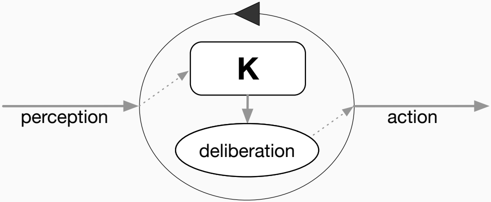


Sources of Knowledge:

- the programmers (i.e. initial knowledge of the agent),
- the environment (by perception),
- the other agents (by communication),
- the agent itself (i.e. reasoning or learning).

Autonomous Agent

Autonomous decision-making entity able to react to events while pursuing (pro-actively defined or delegated) goals and directing actions to achieve them



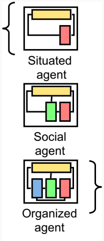
A note about "Autonomy":

- Agents encapsulate and control their knowledge
- Agents deliberate about their actions
- Agents influence the other agents (delegation - adoption of knowledge)

Types of Autonomous Agents

Combining external factors that agents reason about:

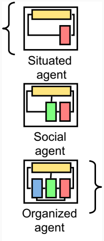
- *Situated* Agents: agents that reason about *themselves* and about their *environment*
- *Social* Agents: situated agents that reason about the *interactions* with other agents
- *Organized* Agents: agents that reason about the *organisations* (e.g. social structures, norms)



Types of Autonomous Agents

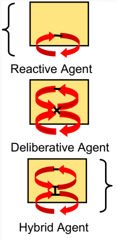
Combining external factors that agents reason about:

- *Situated* Agents: agents that reason about *themselves* and about their *environment*
- *Social* Agents: situated agents that reason about the *interactions* with other agents
- *Organized* Agents: agents that reason about the *organisations* (e.g. social structures, norms)



with the coupling of inputs to actions:

- *Reactive* Agent: tight coupling between perception of the external factors with action
- *Deliberative* Agent: deliberation on the actions to execute from their perception of the external factors and from their goals (loose coupling)
- *Hybrid* Agent: agents that are mixing reactivity and deliberation



Agent Knowledge (Main Language Constructs in JaCaMo)

- Beliefs: explicit representation of information available to the agent, on which the agent can reason (information about the environment, other agents or itself, organisation, ...)

```
temperature(20).          happy(bob).
```

- Goals: state of affairs that the agent would like to bring about

```
!temperature(20).          !happy(bob).
```

- Actions: internal, external, communicative or organisational means to change the state of environment/organisation, to interact with agents
- Plans: a recipe for action(s)
  - to achieve one of the possible goals that an agent might have

```
+!temperature(20) <- turn_on(fac).
+?happy(bob) <- .send(bob,askOne,happy(X)).
```

  - to react to perceived events or to mental state changes

```
+temperature(10)  <- !temperature(20).
-happy(bob) <- !happy(bob).
```

Note: identifiers starting with upper case denote variables

Agent Knowledge (Main Language Constructs in JaCaMo)

- Beliefs: explicit representation of information available to the agent, on which the agent can reason (information about the environment, other agents or itself, organisation, ...)

```
temperature(20).          happy(bob).
```

- Goals: state of affairs that the agent would like to bring about

```
!temperature(20).          !happy(bob).
```

- Actions: internal, external, communicative or organisational means to change the state of environment/organisation, to interact with agents
- Plans: a recipe for action(s)
  - to achieve one of the possible goals that an agent might have

```
+!temperature(20) <- turn_on(fac).
+?happy(bob) <- .send(bob,askOne,happy(X)).
```

  - to react to perceived events or to mental state changes

```
+temperature(10)  <- !temperature(20).
-happy(bob) <- !happy(bob).
```

Note: identifiers starting with upper case denote variables

Agent Knowledge (Main Language Constructs in JaCaMo)

- Beliefs: explicit representation of information available to the agent, on which the agent can reason (information about the environment, other agents or itself, organisation, ...)

```
temperature(20).          happy(bob).
```

- Goals: state of affairs that the agent would like to bring about

```
!temperature(20).          !happy(bob).
```

- Actions: internal, external, communicative or organisational means to change the state of environment/organisation, to interact with agents
- Plans: a recipe for action(s)
  - to achieve one of the possible goals that an agent might have

```
+!temperature(20) <- turn_on(fac).
+?happy(bob) <- .send(bob,askOne,happy(X)).
```

  - to react to perceived events or to mental state changes

```
+temperature(10)  <- !temperature(20).
-happy(bob) <- !happy(bob).
```

Note: identifiers starting with upper case denote variables

Agent Knowledge (Main Language Constructs in JaCaMo)

- Beliefs: explicit representation of information available to the agent, on which the agent can reason (information about the environment, other agents or itself, organisation, ...)

```
temperature(20).          happy(bob).
```

- Goals: state of affairs that the agent would like to bring about

```
!temperature(20).          !happy(bob).
```

- Actions: internal, external, communicative or organisational means to change the state of environment/organisation, to interact with agents
- Plans: a recipe for action(s)
  - to achieve one of the possible goals that an agent might have

```
+!temperature(20) <- turn_on(fac).
+?happy(bob) <- .send(bob,askOne,happy(X)).
```

  - to react to perceived events or to mental state changes

```
+temperature(10)  <- !temperature(20).
-happy(bob) <- !happy(bob).
```

Note: identifiers starting with upper case denote variables

Agent Knowledge (Main Language Constructs in JaCaMo)

- Beliefs: explicit representation of information available to the agent, on which the agent can reason (information about the environment, other agents or itself, organisation, ...)

```
temperature(20).          happy(bob).
```

- Goals: state of affairs that the agent would like to bring about

```
!temperature(20).          !happy(bob).
```

- Actions: internal, external, communicative or organisational means to change the state of environment/organisation, to interact with agents
- Plans: a recipe for action(s)
  - to achieve one of the possible goals that an agent might have

```
+!temperature(20) <- turn_on(fac).
+?happy(bob) <- .send(bob,askOne,happy(X)).
```

  - to react to perceived events or to mental state changes

```
+temperature(10)  <- !temperature(20).
-happy(bob) <- !happy(bob).
```

Note: identifiers starting with upper case denote variables

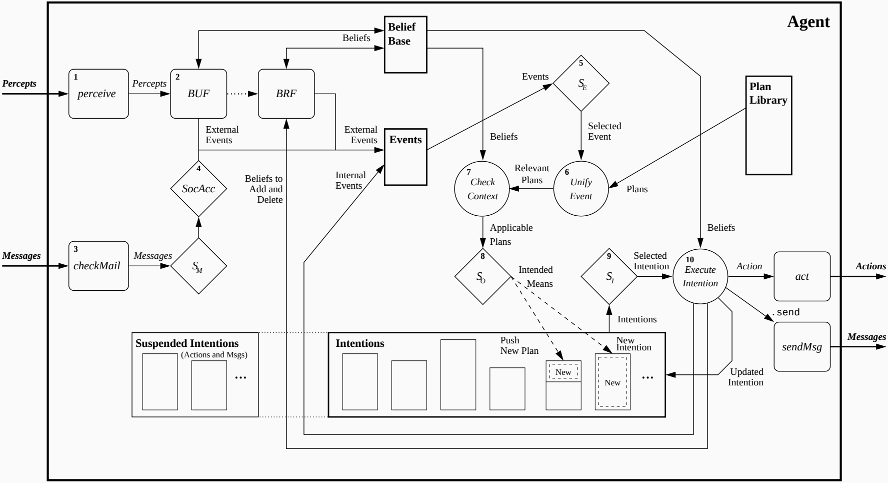
Agent Dynamics

Agent control cycle

1. Perceive the environment and update beliefs
2. Process new messages and update beliefs and goals
3. Select event
4. Select relevant plans
5. Select applicable plans
6. Create/update intention
7. Select intention to execute
8. Execute one step of the selected intention

- Events: consequences to changes in the agent's beliefs or goals
- Intentions: instantiated plans

Agent Dynamics: Agent control cycle



Autonomous Agents Situated in an Environment

Practice

Practice Agent Programming (in JaCaMo)

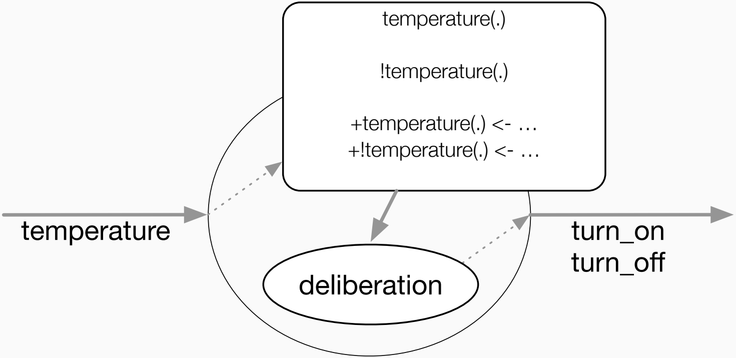
- Enter the folder `smart-room-agent`

Room temperature controller agent implemented with diverse complexity

1. Reactive agent to new belief
2. Reactive agent turns off HVAC
3. Context-based plans for new beliefs
4. Context-based plans for goals
5. `.jcm` set beliefs and goals

`./gradlew run --args="stepX.jcm"`, where `X` is the version of the room temperature controller agent implementation

Practice Agent Programming (in JaCaMo)



Practice Agent Programming (in JaCaMo)

JaCaMo `.jcm` configuration file

```
mas smart_room_agent {

  agent room_agent : room_agent_1.asl {
    focus: room.hvac
  }

  workspace room {
    artifact hvac: devices.HVAC(30)
  }
}
```

Practice Agent Programming (in JaCaMo)

```java
public class HVAC extends Artifact {
  private double temperature;

  void init(double temp) {
    this.temperature = temp;
    defineObsProperty("state", "off");
    defineObsProperty("temperature", this.temperature);
  }
  @OPERATION void turn_on() {
    if (getObsProperty("state").stringValue().equals("off")) {
      getObsProperty("state").updateValue("on");
      this.execInternalOp("updateTemperatureProc", -1);
    }
  }
  @OPERATION void turn_off() {
    if (getObsProperty("state").stringValue().equals("on")) {
      getObsProperty("state").updateValue("off");
    }
  }
  @INTERNAL_OPERATION void updateTemperatureProc(double step) {
    ObsProperty temp = getObsProperty("temperature");
    ObsProperty state = getObsProperty("state");
    while (!state.stringValue().equals("off")) {
      temp.updateValue(temp.doubleValue() + step);
      this.await_time(300);
    }
  }
}
```

Practice Agent Programming (in JaCaMo) i

Room temperature controller agent **reacts to a new belief to achieve the temperature goal**, but never turns the HVAC off

```
// reacting to a new belief and creating a new goal (proactivity)
+temperature(30) <- !temperature(20) .

// achieving to a new goal by acting
+!temperature(20) <- turn_on .
```

Practice Agent Programming (in JaCaMo) ii

Room temperature controller agent reacts to a new belief to achieve the temperature goal and **turns the HVAC off when the target temperature is reached**

```
// reacting to a new belief and creating a new goal (proactivity)
+temperature(30) <- !temperature(20) .

// reacting to a new belief and acting
+temperature(20) <- turn_off .

// achieving a new goal by acting
+!temperature(20) <- turn_on .
```

Practice Agent Programming (in JaCaMo) iii

Room temperature controller agent reacts to a preferred temperature and **select plans based on context instead of fixed values**

```
// initial belief, given by the developer
preferred_temp(20) .

// reacting to changes in the temperature
+temperature(T) : preferred_temp(P) & math.abs(P-T) > 0
  <- !temperature(P) .

+temperature(T) : preferred_temp(T)
  <- turn_off .

// achieving a new goal in a given context
+!temperature(P) : temperature(T) & T > P
  <- turn_on .
```

Practice Agent Programming (in JaCaMo) iv

Room temperature controller agent **does not react to beliefs** and has a **goal to maintain temperature** based on the context

```
// initial belief, given by the developer
preferred_temp(20) .

// initial goal, given by the developer
!keep_temperature .

// maintain goal pattern
+!keep_temperature : temperature(T) & preferred_temp(P) & math.abs(P-T) > 0
  <- turn_on ;
     !keep_temperature .

+!keep_temperature : temperature(T) & preferred_temp(P) & T <= P
  <- turn_off ;
     !keep_temperature .
```

Practice Agent Programming (in JaCaMo) V

JaCaMo `.jcm` configuration file **set the initial beliefs and goals** of the room temperature controller agent

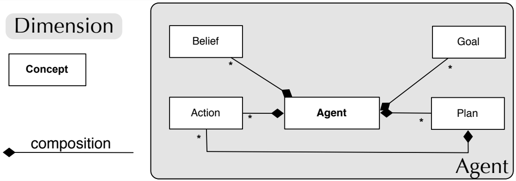
```
mas smart_room_agent {

  agent room_agent : room_agent_5.asl {
    beliefs: preferred_temp(20)
    goals: keep_temperature
    focus: room.hvac
  }

  workspace room {
    artifact hvac: devices.HVAC(30)
  }
}
```

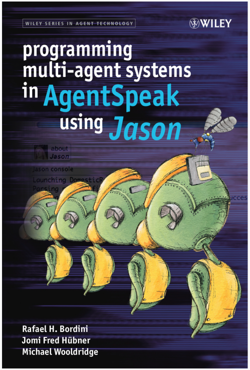
Practice Agent Programming (in JaCaMo) vi

Room temperature controller agent has a **goal to maintain temperature based on the context**

```
// maintain goal pattern
+!keep_temperature : temperature(T) & preferred_temp(P) & math.abs(P-T) > 0
  <- turn_on ;
     !keep_temperature .

+!keep_temperature : temperature(T) & preferred_temp(P) & T <= P
  <- turn_off ;
     !keep_temperature .
```

Autonomous Agent: Main Features

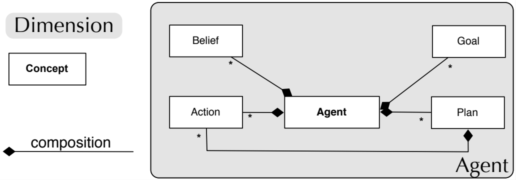


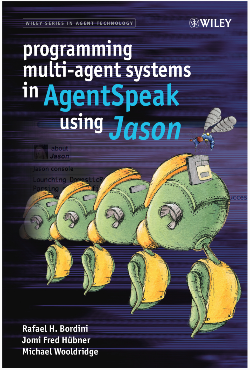


- Reactivity: reaction to new beliefs or goals
- Pro-activity: creation of new goals
- Long-term goals: commitment to achieve goals
- Context awareness: selection of plans based on the circumstances

- Transparency: the reasons for an action can be traced back
- sound theoretical background for agent architectures (practical reasoning [Bra87], intentions [CL87], BDI [RG95])

Autonomous Agent: Main Features

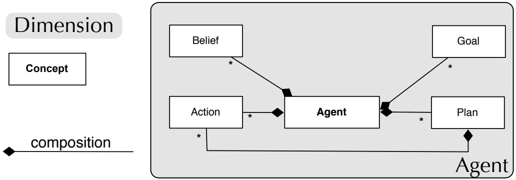


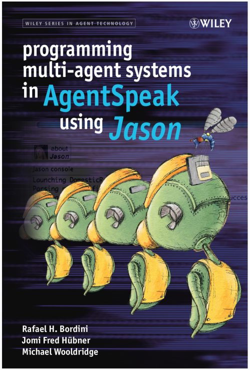

WILEY SERIES IN AGENT TECHNOLOGY
WILEY
programming multi-agent systems in AgentSpeak using Jason
Rafael H. Bordini
Jomi Fred Hübner
Michael Wooldridge

- Reactivity: reaction to new beliefs or goals
- Pro-activity: creation of new goals
- Long-term goals: commitment to achieve goals
- Context awareness: selection of plans based on the circumstances

- Transparency: the reasons for an action can be traced back
- sound theoretical background for agent architectures (practical reasoning [Bra87], intentions [CL87], BDI [RG95])

Autonomous Agent: Main Features

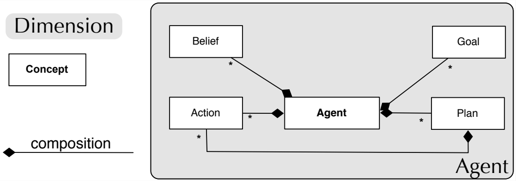


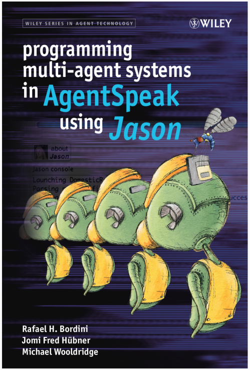


- Reactivity: reaction to new beliefs or goals
- Pro-activity: creation of new goals
- Long-term goals: commitment to achieve goals
- Context awareness: selection of plans based on the circumstances

- Transparency: the reasons for an action can be traced back
- sound theoretical background for agent architectures (practical reasoning [Bra87], intentions [CL87], BDI [RG95])

Autonomous Agent: Main Features



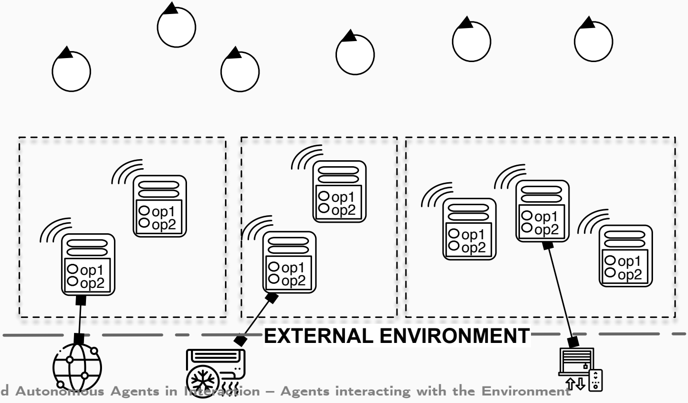


- Reactivity: reaction to new beliefs or goals
- Pro-activity: creation of new goals
- Long-term goals: commitment to achieve goals
- Context awareness: selection of plans based on the circumstances

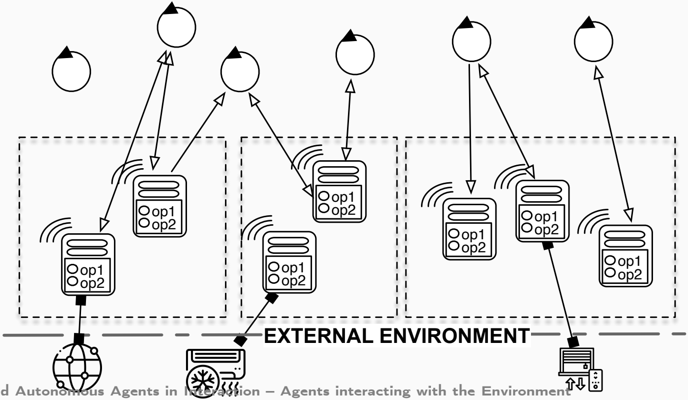
- Transparency: the reasons for an action can be traced back
- sound theoretical background for agent architectures (practical reasoning [Bra87], intentions [CL87], BDI [RG95])

Situated Autonomous Agents in Interaction

Autonomous Agents situated in an Environment



Autonomous Agents situated in an Environment



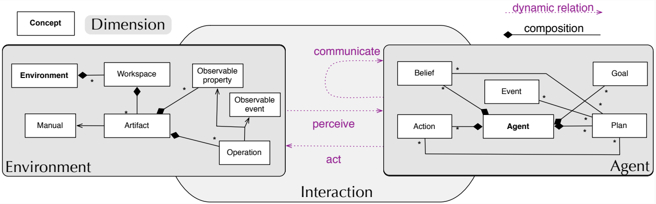
Interaction between Agents & the Environment

- Agents and artifacts are visible to each other within the workspace in which they are situated.

- Agents interact with the environment as well as indirectly with other agents by means of the environment.
  - Agents can perceive (i.e., observe or sense) the artifacts situated in their workspace and react to that perception.
  - Agents can also act upon the artifacts of their workspace to change their state.

- Agents are the ones who trigger the environment life cycle

Integrating Agent & Environment Dimensions



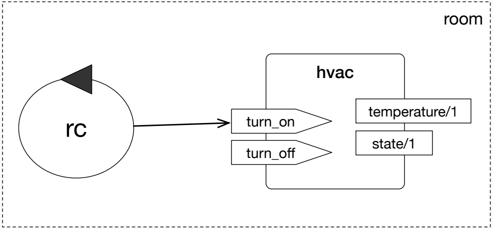
Mapping of:

- Artifacts' operations onto agents' actions
- Artifacts' observable properties onto agents' beliefs
- Artifacts' signals onto agents' belief-update events related to observable events

An agent has a dynamic repertoire of actions:

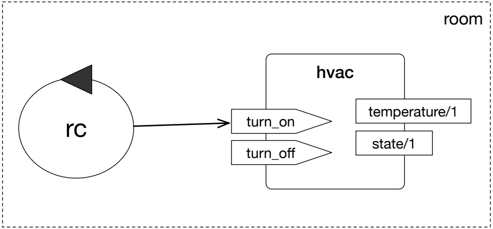
- Repertoire of operations from the set of artifacts available in the workspaces where the agent is situated
- Dynamic repertoire that can be changed by creating/disposing artifacts in the corresponding workspaces or by joining other workspaces

Interaction Model – Action Execution / Agent side



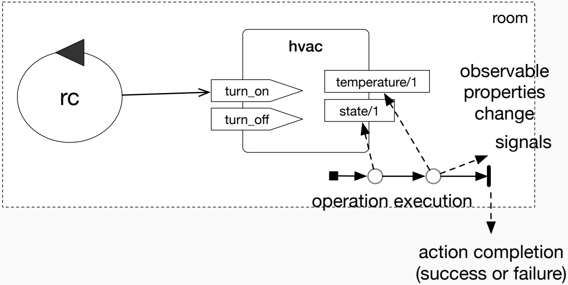
- Success/failure semantics of an action is defined from the operation semantics
- Executing an action suspends the intention of the agent until success or failure of the corresponding operation
  - Action success/failure events are generated by the artifact and perceived by the agent without explicit observation or reasoning by the agent.
- *The agent control cycle is not blocked*!
  - The agent can continue to process percepts and possibly execute actions of other intentions

Interaction Model – Act



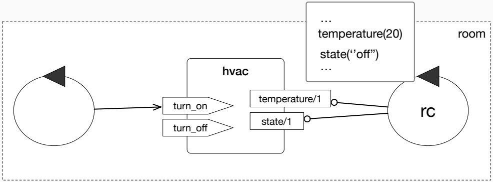
- Acting on Artifact's usage interface: triggering the execution of an operation
- Concurrency: Artifacts can be safely used by multiple agents
- Mutual exclusion enforced:
    - Only one operation in execution at a time in an artifact
    - If multiple operations have been triggered and are in execution, all but one are suspended (for instance, in awaiting some condition)

Interaction Model – Operation Execution / Artifact side



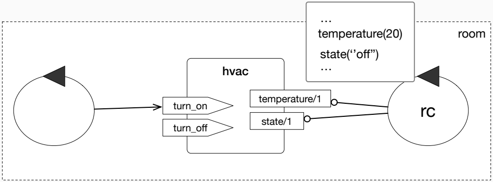
- Operation execution can be a process structured in one or multiple transactional steps
- Asynchronous with respect to agent
    - *the agent can proceed possibly reacting to percepts and executing actions of other plans/activities*
- Operation completion causes action completion
    - Action completion events with success or failure, possibly with action feedbacks

Interaction Model: Observation



- Agents can dynamically select which artifacts to observe: focus/stopFocus actions
- By focussing an artifact
    - observable properties are mapped into agent's beliefs
    - signals are mapped as beliefs related to observable events

Interaction Model: Observation (contd)



- Observation completeness (no events can be lost): every observable state generated by an artifact is perceived by every agent observing the artifact
- Event ordering
    - Observable states and events generated by an artifact are perceived by every agent observing the artifact in the generation order.
    - No order is defined between events generated by different artifacts.
- Atomic perception: observable properties changed in the same operation execution are a single percept in the same reasoning cycle.

Situated Autonomous Agents in Interaction

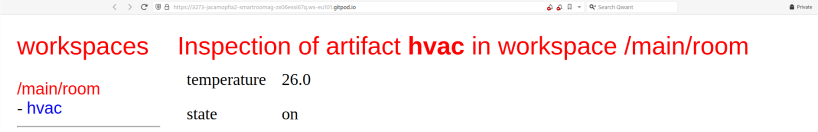
Practice

Inspecting Artifacts and Agents Mental States (in JaCaMo)

- JaCaMo allows inspecting Artifacts and Agents Mental states
- After running the application, you can open the Mind Inspector by
  - Selecting the URL and Ports

Port 3272	:	Agent Mind inspector
Port 3273	:	Artifact inspector

Inspecting Artifacts and Agents Mental States (in JaCaMo)



https://3273-jacamopfia2-smartroomag-zx06essi67q.ws-eu101.gitpod.io
Search Qwant
Private

workspaces

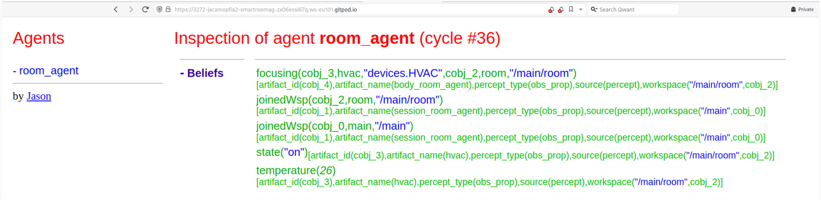
/main/room
- hvac

Inspection of artifact **hvac** in workspace /main/room

temperature	26.0
state	on

- hvac observable states
  - `state` = `on`
  - `temperature` = 26

Inspecting Artifacts and Agents Mental States (in JaCaMo)



https://3272-jacamopfia2-smartroomag-zx06essi67q.ws-eu101.gitpod.io
Search Qwant
Private

Agents

- room_agent

by Jason

Inspection of agent **room_agent** (cycle #36)

- Beliefs

```
focusing(cobj_3,hvac,"devices.HVAC",cobj_2,room,"/main/room")
[artifact_id(cobj_4),artifact_name(body_room_agent),percept_type(obs_prop),source(percept),workspace("/main/room",cobj_2)]
joinedWsp(cobj_2,room,"/main/room")
[artifact_id(cobj_1),artifact_name(session_room_agent),percept_type(obs_prop),source(percept),workspace("/main",cobj_0)]
joinedWsp(cobj_0,main,"/main")
[artifact_id(cobj_1),artifact_name(session_room_agent),percept_type(obs_prop),source(percept),workspace("/main",cobj_0)]
state("on")[artifact_id(cobj_3),artifact_name(hvac),percept_type(obs_prop),source(percept),workspace("/main/room",cobj_2)]
temperature(26)
[artifact_id(cobj_3),artifact_name(hvac),percept_type(obs_prop),source(percept),workspace("/main/room",cobj_2)]
```

- `room_agent` beliefs at cycle #36
  - `state` = `on`
  - `temperature` = `26`
- situated in the `room` workspace
- originating from `HVAC` device

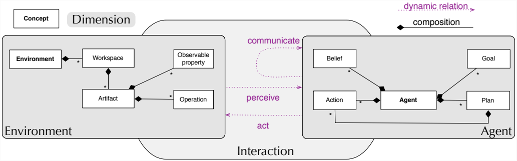
Inspecting Artifacts and Agents Mental States (in JaCaMo)

Exercise

- Open smart-room-agent folder
- Change line 43 of `src/env/devices/HVAC.java`
  From `this.await_time(300);`
  To `this.await_time(5000);`
- Run

  ```
  ./gradlew run --args="step2.jcm"
  ```
- Inspect the Artifact and Agent Mind states

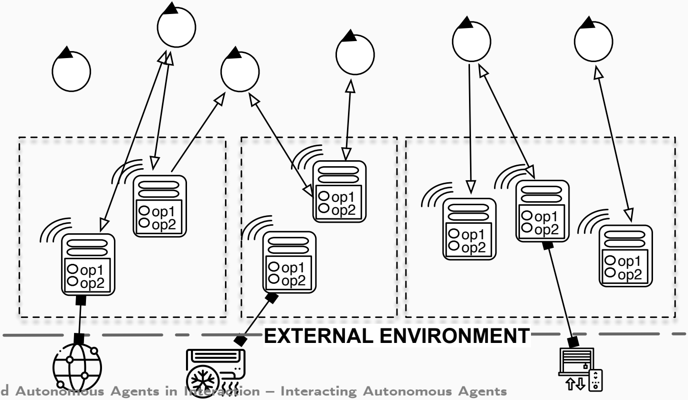
Autonomous Agents situated in an Environment: Main Features



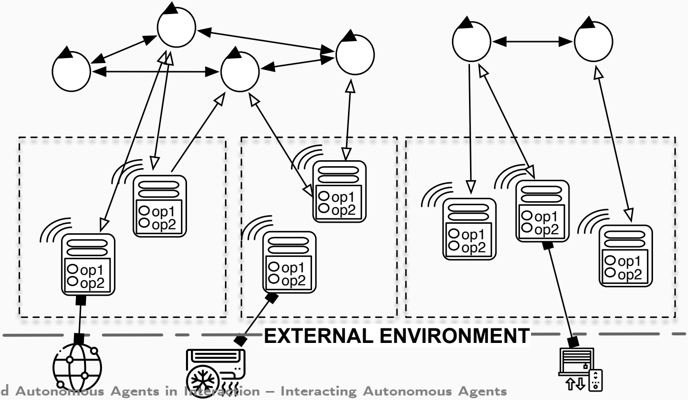
based on A&A [RPVO09a]

- Shared environment among agents ⇝ medium of indirect interaction between agents
- Situatedness: actions and perceptions of agents depend on the situation of the agent
- Agents can build/instrument their (virtual) environment
- Separation of concerns: autonomous entities (agents) vs non autonomous entities (tools/artifacts)
- Artifacts are not objects: they are conceived for agents with atomic and transactional operations
- Sound theoretical background (Activity Theory [Nar96])

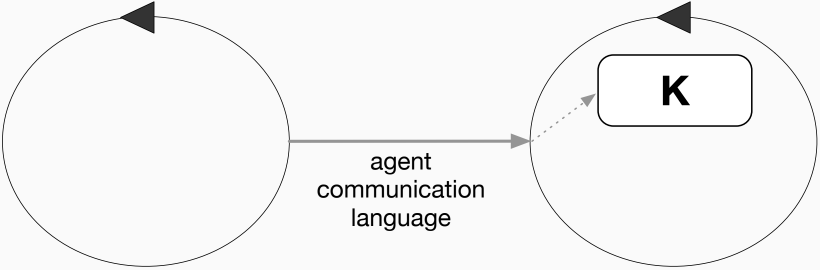
Autonomous Agents situated in an Environment



Autonomous Agents situated in an Environment

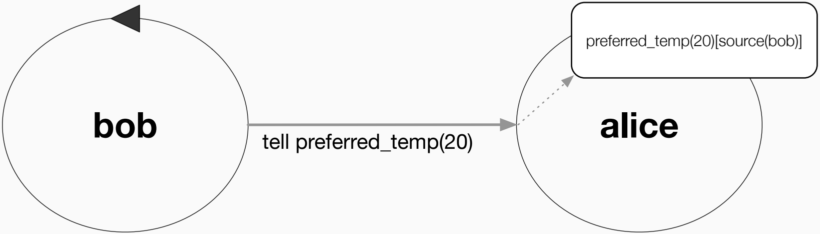


Agent – Agent Communication



- Communicative action: an action that allows an agent to directly communicate to one or more, possibly all, other agents in a multi-agent system programmed with an agent communication language (ACL)
- Interaction protocol: global description of the expected exchanges of messages issued from the parties involved in the interaction

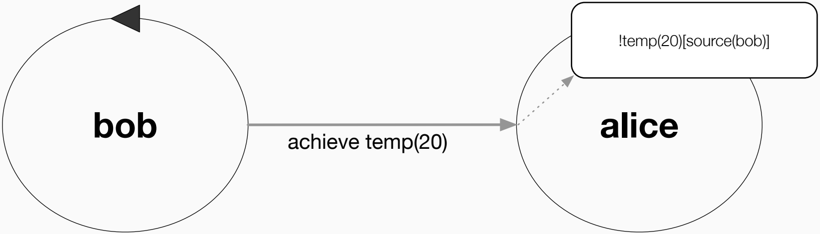
Communicative Actions



A communicative action/message has:

- A performative verb: explicit representation of the intended purpose of the message (e.g. tell, achieve, or ask) that will affect what the receiving agent does with the actual content of the message representing some knowledge, preference, or know-how.
- A content (belief, goal, plan)

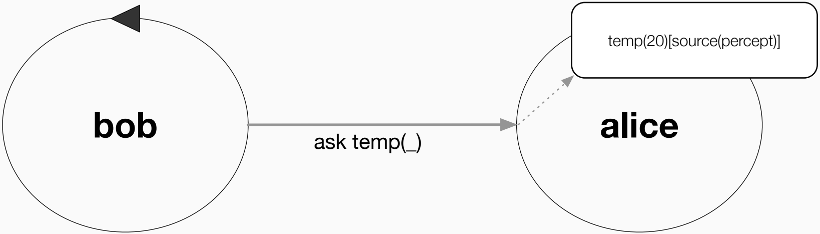
Communicative Actions



A communicative action/message has:

- A performative verb: explicit representation of the intended purpose of the message (e.g. tell, achieve, or ask) that will affect what the receiving agent does with the actual content of the message representing some knowledge, preference, or know-how.
- A content (belief, goal, plan)

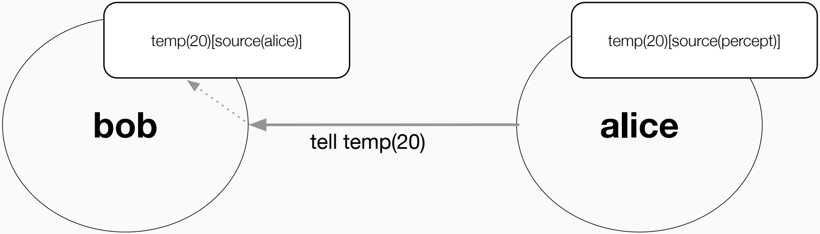
Communicative Actions



A communicative action/message has:

- A performative verb: explicit representation of the intended purpose of the message (e.g. tell, achieve, or ask) that will affect what the receiving agent does with the actual content of the message representing some knowledge, preference, or know-how.
- A content (belief, goal, plan)

Communicative Actions



A communicative action/message has:

- A performative verb: explicit representation of the intended purpose of the message (e.g. tell, achieve, or ask) that will affect what the receiving agent does with the actual content of the message representing some knowledge, preference, or know-how.
- A content (belief, goal, plan)

Situated Autonomous Agents in Interaction

Practice

Interaction among Agents in JaCaMo

- Open smart-room-interaction folder

Interaction among Agents in JaCaMo

```
mas smart_room_interaction {

  agent pa1 : personal_assistant.asl {
    beliefs: preferred("reading", high)
             preferred("watching", high)
             preferred("cooking", high)
             preferred("sport", medium)
             activity("reading")
    join: room
  }

  ...

  agent room_agent : room_agent.asl {
    focus: room.hvac
  }

  workspace room {
    artifact hvac: devices.HVAC(30)
  }
}
```

Interaction among Agents in JaCaMo

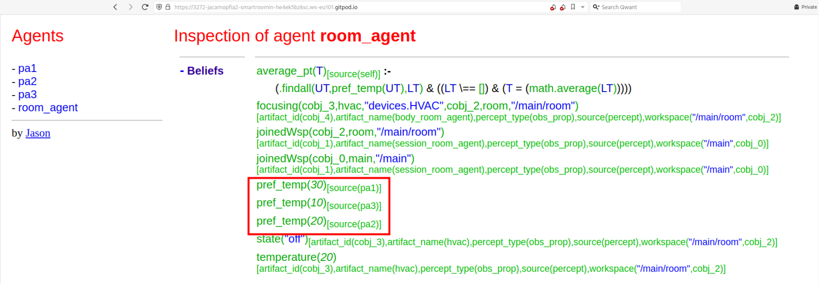
```
    // Translation of the low, medium, high temperature levels into the internal agent
level_temp(low,    10) .
level_temp(medium, 20) .
level_temp(high,   30) .

// Inference of the preferred temperature level from the activity
pref_temp(T) :- activity(A) & preferred(A, L) & level_temp(L, T) .

+activity(A) : A \== "none"
  <- ?pref_temp(T) ;
     .print("New user activity ", A, " preferred temperature is ", T) ;
     .send(room_agent, tell, pref_temp(T)) .
```

Interaction among Agents in JaCaMo

- Run

  `./gradlew`

- Inspect the `room_agent` agent



https://3272-jacamopfia2-smartroomin-he4ek5bzksc.ws-eu101.gitpod.io

Agents

- pa1
- pa2
- pa3
- room_agent

by Jason

Inspection of agent **room_agent**

- Beliefs

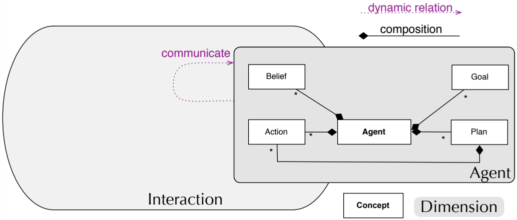
```
average_pt(T)[source(self)] :-
     (.findall(UT,pref_temp(UT),LT) & ((LT \== []) & (T = (math.average(LT))))
focusing(cobj_3,hvac,"devices.HVAC",cobj_2,room,"/main/room")
[artifact_id(cobj_4),artifact_name(body_room_agent),percept_type(obs_prop),source(percept),workspace("/main/room",cobj_2)]
joinedWsp(cobj_2,room,"/main/room")
[artifact_id(cobj_1),artifact_name(session_room_agent),percept_type(obs_prop),source(percept),workspace("/main",cobj_0)]
joinedWsp(cobj_0,main,"/main")
[artifact_id(cobj_1),artifact_name(session_room_agent),percept_type(obs_prop),source(percept),workspace("/main",cobj_0)]
pref_temp(30)[source(pa1)]
pref_temp(10)[source(pa3)]
pref_temp(20)[source(pa2)]
state("off")[artifact_id(cobj_3),artifact_name(hvac),percept_type(obs_prop),source(percept),workspace("/main/room",cobj_2)]
temperature(20)
[artifact_id(cobj_3),artifact_name(hvac),percept_type(obs_prop),source(percept),workspace("/main/room",cobj_2)]
```

Interaction among Agents in JaCaMo

Exercise

- Open smart-room-interaction
- Change the agents' activities
- Run
  ```
  ./gradlew
  ```
- Inspect the Agent and the final preferred room temperature

Interacting Autonomous Agents: Main Features

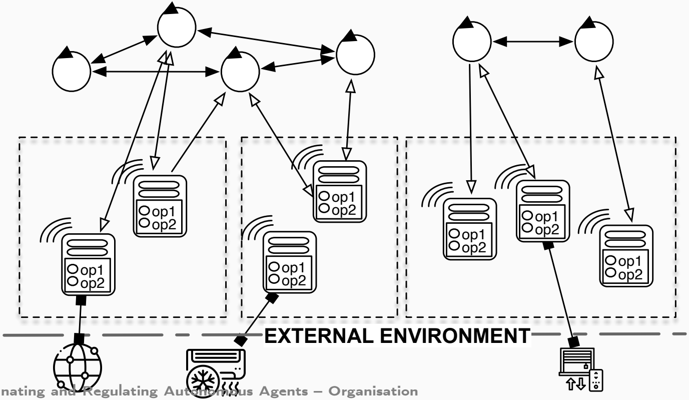


based on KQML or Jade/FIPA ACL

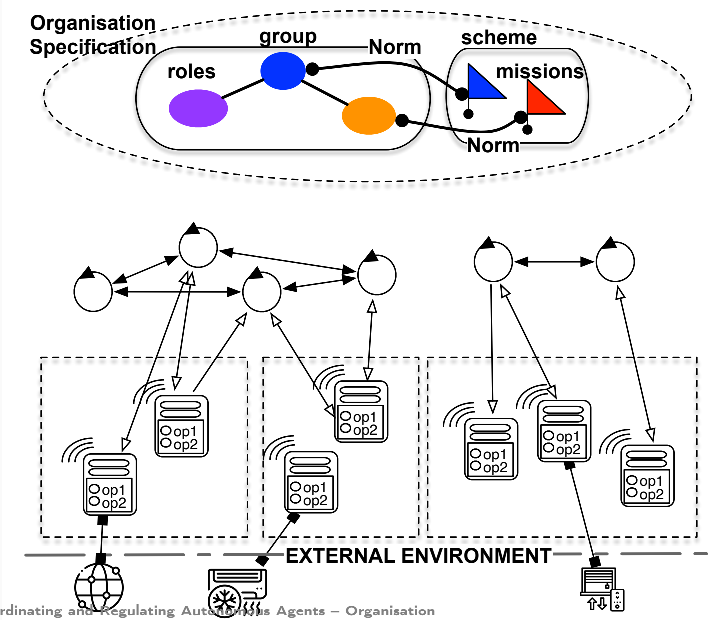
- we are not programming computers,
  we are programming agents, which are based on knowledge and autonomy
- communication is not about data exchange, but
  knowledge sharing, delegation, adoption

Coordinating and Regulating Autonomous Agents

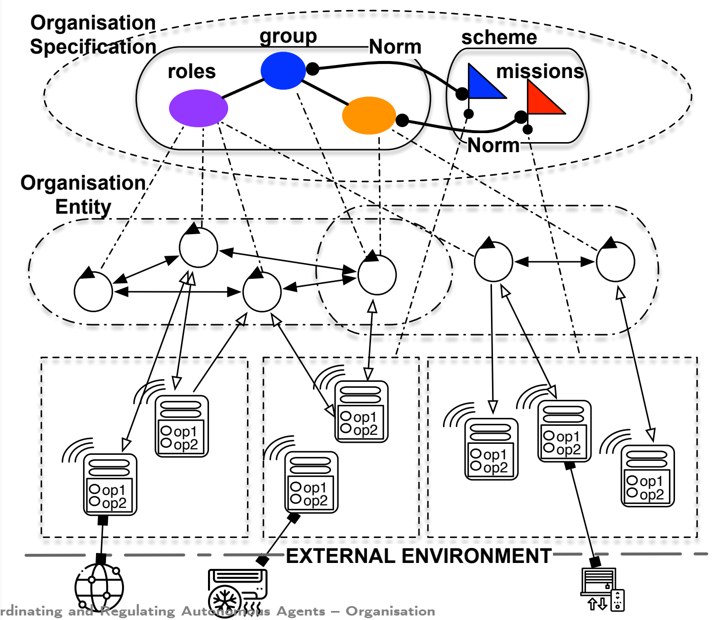
Autonomous Agents situated in an Environment



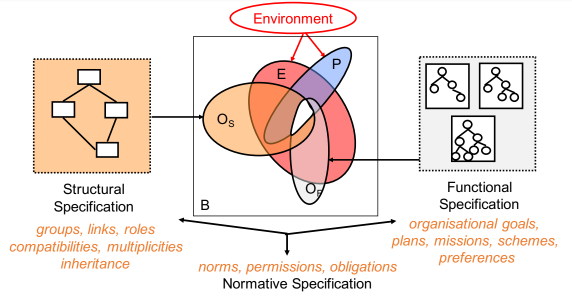
Coordinating and Regulating Autonomous Agents



Coordinating and Regulating Autonomous Agents



Coordinating and Regulating Autonomous Agents



- Integrating Structural (i.e. Roles, Groups), Functional (i.e. Organisational Goals, Missions, Schemes) independent sets of concepts within a Normative perspective (i.e. Norms with obligations, permissions, interdictions)
  ⇝ enabling agent's autonomy w.r.t. organisation (enforcement vs regimentation)
- Programming and representing the organisation ⇝ making it accessible to the designers, the agents, the coordination and regulation management infrastructure [HBKR10]

Basic Concepts

- Organisation: abstractions to declare and make accessible to agents their (current and/or expected) collective coordinated and regulated relations and activities in the shared environment ⇝ organisation entity, organisation specification

---

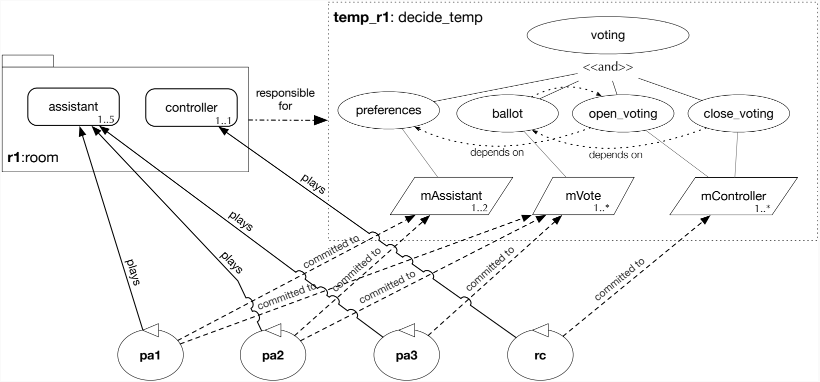
- Group: social context in which agents can play roles, undertake their expected coordinated behavior as well as their rights and duties.
- Role: statement that determines the interactions, relations, rights and duties taking place for an agent within a group.

- Organisational goal: state of affair that has to be (or has been) satisfied by one or several agents
- Mission: set of organisational goals that have to be achieved under the responsibility of an individual agent in the organisation
- Social scheme: a goal decomposition tree executed in a coordinated manner under the responsibility of agents participating to a group in accordance to their rights and duties

---

- Norm: statement of the rights and duties of agents in the context of an organisation

Declarative Organisation Programming



- Structural patterns (groups (r1:room), roles (assistant, controller), links)
- Coordination patterns (goal decomposition trees (voting, preferences, ballot, ...), missions (mAssistant, mVote,mController)
- Rights and duties (norms)

Organisation Dynamics

In the context of Organisation life-cycle:

- Creation/Deletion of an Organisation from an Organisation specification
- Entrance/Exit of an agent
- Change of Organisation specification

In the context of Organisation structure life-cycle:

- Creation/Deletion of a group
- Adoption/Leave of a role

In the context of Coordination activity life-cycle:

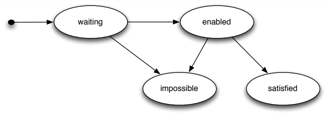
- Creation/End of a schema
- Commitment/Release of a mission
- Change of goal state

In the context of Normative Regulation activity life-cycle:

- Activation/De-activation of norms
- Fulfillment/Violation of norms
- Enforcement of norms

Organisation Dynamics

Organisational Goal dynamics



waiting initial state

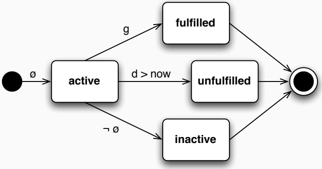
enabled goal pre-conditions are satisfied & scheme is well-formed

satisfied agents committed to the goal have achieved it

impossible the goal is impossible to be satisfied

Note: goal state from the Organization point of view may be different of the goal state from the Agent point of view

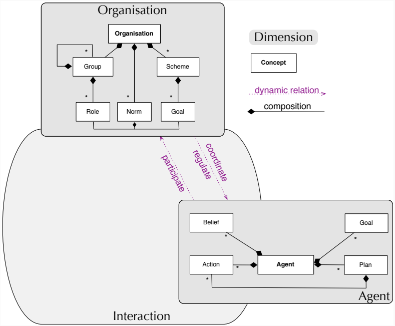
Norm dynamics



$norm\ n : \ \phi -> \ obligation(a, r, g, d)$

- ϕ: activation condition of the norm (e.g. play a role)
- g: the goal of the obligation (e.g. commit to a mission)
- d: the deadline of the obligation

Integrating Agent & Organisation Dimensions



```
+!create_org  <-   ...
 makeArtifact(shouseo, "ora4mas.nopl.OrgBoard",
   ["src/org/smart_house.xml"],
   OrgArtId)[wid(WspId)];
 focus(OrgArtId)[wid(WspId)];
 ....
 // adopts the role controller in the group
 adoptRole(controller)[artifact_id(GrArtId)];
 .... .
 ....
 +obligation(Ag,Norm,What,Deadline)
   [artifact_id(ArtId)] : .my_name(Ag) &
   (satisfied(Scheme,Goal)=What) |
   done(Scheme,Goal,Ag)=What)
 <- !Goal[scheme(Scheme)];
   goalAchieved(Goal)[artifact_id(ArtId)].
```

based on ORA4MAS [HBKR10]

- Dynamic organisational facts that represent about the participation of agents to the organisation entity:
  - coordination facts: management of dependencies between activities carried out by agents within the organisation.
  - regulation facts: normative expectation on activities carried out by agents.
- Events, Properties and Operations enabling agents to be aware of, to deliberate and act: entering/exiting the organisation, modifying the organisation, obeying/violating norms, sanctioning/rewarding other agents

Organisation Programming

- Including **organisation-reasoning** abilities into agents

```
+play(Ag,assistant,GrId) <- .send(Ag,tell,hello).
+goalState(_,close_voting,_,_,satisfied) <- ...
```

- Including **norm-reasoning** abilities into agents

```
+obligation(Ag,Norm,achieved(_,Goal,_),DeadLine)
    : .my_name(Ag) & good(mood)
<- !Goal.
```

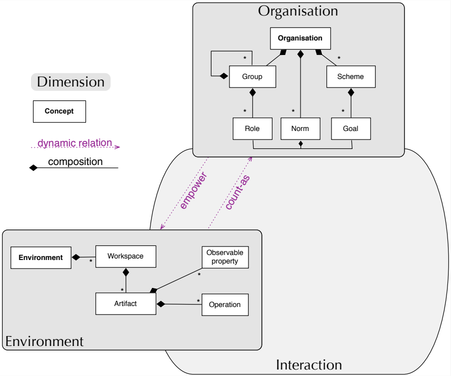
Integrating Organisation & Environment Dimensions

- In order to support the joint work of autonomous agents situated in a shared environment, an organisation needs to be situated and aware on what actions, agents are executing in the environment.

⇝ Changes in the state of the environment may count-as changes in the state of the organization.

- This dynamic relation is a practical way of situating organizations in an environment, as happens for the agents, regulating some part of the environment (e.g., a traffic light at a crossroads) in a particular way and ruling it differently in other parts.
- Organizations may empower the elements of the environment by allowing them to control and regulate actions or perception of the agents.

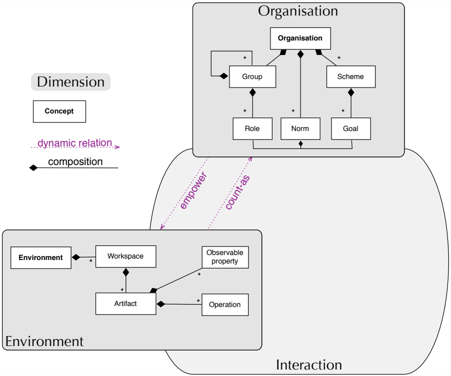
Basic concepts



```
institution_id : shInst.
status_functions:
states: play(A,R,G), responsible(G,S),
committed(A,Mission,S), achieved(S,G,A),
done(S,G,A).

constitutive_rules:
...
2:
play(A,assistant,âĂİroomâĂİ)
count-as committed(A,mAssistant,âĂİdecide_tempâĂİ)
3:
play(A,assistant,âĂİroomâĂİ)
count-as committed(A,mVote,âĂİdecide_tempâĂİ)
4:
play(A,controller,âĂİroomâĂİ)
count-as committed(A,mController,âĂİdecide_tempâĂİ)
...
```

based on Situated Artificial Institution [dBHB15]

Basic concepts



Transforming organisations into situated organisations [dBHB12], [PRBH09], [OBdRC08] so that:

- organisation may act on the environment (e.g. enact rules, regimentation)
- environment may act on the organisation (e.g. count-as rules) based on Situated Artificial Institution [dBHB15]

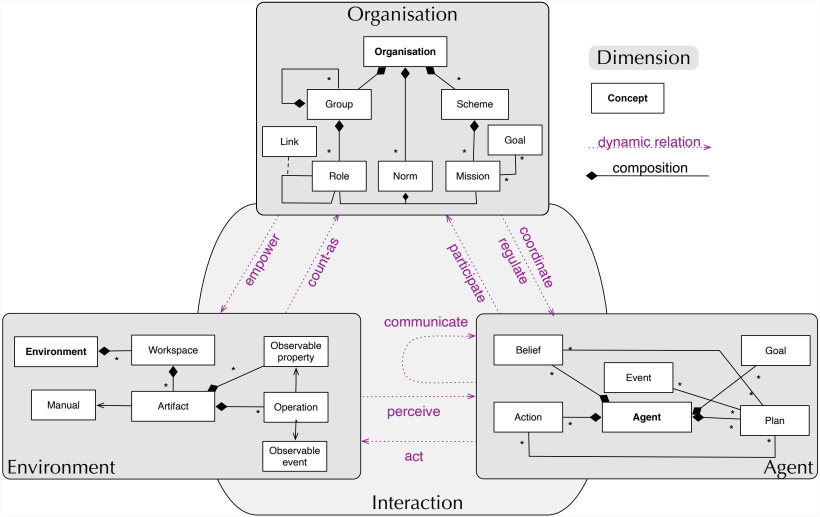
based on Situated Artificial Institution [dBHB15]

Coordinating and Regulating Autonomous Agents: Main features

- Model to specify global orchestration $\rightsquigarrow$ team strategy defined at a high level
- Ensures that the agents follow some of the constraints specified for the organisation
- Helps the agents to work together
- The organisation is *interpreted at runtime*, it is not hardwired in the agents code
- The agents ‘handle’ the organisation (i.e. their artifacts)
- It is suitable for open systems as no specific agent architecture is required
- Organization can easily be changed by the developers or by the agents themselves

Wrap-up and Next Steps

Multi-Agent Oriented Programming: Global View



Mainstream Application Domains and Technologies

- *ChatBot* [EDK+21]
- Mobile and *Wearable* Apps [CR21]
- IoT, IIoT, Web of Things and *Web* technologies: Hybrid Communities of Agents and Humans on the Web [CMG+19, CBR19] (HyperAgent project – ANR-FSF Project)
- Robotics and *Drones* (embedded agents) [MHB18]

- Agent-Based Mixed Reality Environments in *Healthcare* [CBR20]
- Multi-Agent for *Industry of the Future* [CMM18]
- Multi-Agent for Collaborative *Learning* [dSCVMB21]

Take Away

- A Multi-Agent System is not only agents, or only organization, or only environment, or only interaction! It has all these dimensions! All are *first class entities*!
- MAOP proposes a *separation of concerns* between Agent, Environment, Interaction and Organization that brings rich features to engineer open, long lived, agile, non centralized, distributed intelligent systems
- MAOP paves the way to the **inclusion of physical, digital, human and social worlds** to define socio-cognitive, physical and digital systems
- *JaCaMo* proposes a seamless integration of these different abstractions interfacing to mainstream technologies

Challenges and Some directions

- *Explainability* for all dimensions: going beyond the agents
- *Ethics* of intelligent system at the individual level but also considering the collective level w.r.t. organizational values, norms, cultures
- *Hybrid* communities of people and agents interconnecting physical and virtual environments, laws in the social world with laws in the digital world
- *Scalability* - distributed environment: agents on the Web, agents in industrial environments, in smart cities

In tight collaboration with

R.H. Bordini,
Pontificia Universidade Catolica do Rio Grande do Sul (PUCRS), Porto Alegre, Brazil

J.F. Hübner,
Federal University of Santa Catarina (UFSC), Florianópolis, Brazil

A. Ricci
University of Bologna (UNIBO), Bologna, Italy

Acknowledgements

- Jaime Sichman (USP Brazil), various colleagues and students
- JaCaMo users for helpful feedback
- CNPq, CAPES, ANP, ANR for supporting some of our current research
- Schloss Dagstuhl - Leibniz Center for Informatics

Acknowledgements

- Tutorials
  - @European Summer Schools (EASSS'10, EASSS'11, EASSS'12, EASSS'23)
  - @Brazilian Agent School (WESAAC'11, WESAAC'12, WESAAC'13, WESAAC'14)
  - @AAMAS (2015), @IJCAI (2015)
  - @Agreement Technology Summer School (2017)
  - Gratuate Programs (@ EMSE, @ UFSC, @PUCRS, @ Bologna University)
- Dagstuhl Seminars:
  - #23081 (2023), #21072 (2021), #12342 (2012), #08361 (2008), #06261 (2006)
- Bilateral Projects:
  - USP-COFECUB 98-04, CMIRA Rhône-Alpes Region 2010
- National Project:
  - FORTRUST (ANR 06-10), ETHICAA (ANR 14-18), HYPERAGENTS (ANR 20-24), NAIMAN (ANR 23-27)

References

Bibliography i

Rafael H. Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah-Seghrouchni, Jorge J. Gómez-Sanz, João Leite, Gregory M. P. O'Hare, Alexander Pokahr, and Alessandro Ricci.

A survey of programming languages and platforms for multi-agent systems.

Informatica (Slovenia), 30(1):33–44, 2006.

Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, Alessandro Ricci, and Andrea Santi.

Multi-agent oriented programming with jacamo.

Science of Computer Programming, pages –, 2011.

Bibliography ii

- Olivier Boissier, Rafael H. Bordini, Jomi Hübner, and Alessandro Ricci.

  ***Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*.**

  MIT Press, 2020.

- Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni, editors.

  ***Multi-Agent Programming: Languages, Platforms and Applications*.**

  Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer, 2005.

Bibliography iii

- Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, editors.

  ***Multi-Agent Programming: Languages, Platforms and Applications*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*.**

  Springer, 2005.

- Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, editors.

  ***Multi-Agent Programming: Languages, Tools and Applications*.**

  Springer, 2009.

Bibliography iv

Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge.
***Programming Multi-Agent Systems in AgentSpeak Using Jason*.**
John Wiley & Sons, 2007.

Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldrige.
***Programming Multi-Agent Systems in AgentSpeak using Jason*.**
Wiley Series in Agent Technology. John Wiley & Sons, 2007.

Michael E. Bratman.
***Intention, Plans, and Practical Reason*.**
Harvard University Press, Cambridge, 1987.

Bibliography v

- Andrei Ciortea, Olivier Boissier, and Alessandro Ricci.

  Engineering world-wide multi-agent systems with hypermedia.

  In Danny Weyns, Viviana Mascardi, and Alessandro Ricci, editors, *Engineering Multi-Agent Systems - 6th International Workshop, EMAS 2018, Stockholm, Sweden, July 14–15, 2018, revised selected papers*, volume 11375 of *LNCS*, pages 285–301. Springer, 2019.

- Angelo Croatti, Manuel Bottazzi, and Alessandro Ricci.

  Agent-based mixed reality environments in healthcare: The smart shock room project.

  In Yves Demazeau, Tom Holvoet, Juan M. Corchado, and Stefania Costantini, editors, *Advances in practical applications of agents, multi-agent systems, and trustworthiness. The PAAMS collection*, pages 398–402, Cham, 2020. Springer International Publishing.

Bibliography vi

Philip R. Cohen and Hector J. Levesque.
Intention = choice + commitment.
In *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 410–415. Morgan Kaufmann, 1987.

Andrei Ciortea, Simon Mayer, Fabien L. Gandon, Olivier Boissier, Alessandro Ricci, and Antoine Zimmermann.
A decade in hindsight: The missing bridge between multi-agent systems and the world wide web.
In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor, editors, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS'19, Montreal, Canada, May 13–17, 2019*, pages 1659–1663, 2019.

Bibliography vii

Andrei Ciortea, Simon Mayer, and Florian Michahelles.
Repurposing manufacturing lines on the fly with multi-agent systems for the web of things.
In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10–15, 2018*, pages 813–822. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Angelo Croatti and Alessandro Ricci.
Programming agent-based mobile apps: The jaca-android framework.
In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 1724âĂŞ1726,

Bibliography viii

Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

Mehdi Dastani.
2apl: a practical agent programming language.
Autonomous Agents and Multi-Agent Systems, 16(3):214–248, 2008.

Maiquel de Brito, Jomi Fred Hübner, and Rafael H. Bordini.
Programming institutional facts in multi-agent systems.
In *COIN-12, Proceedings*, 2012.

Maiquel de Brito, Jomi F. Hübner, and Olivier Boissier.
Bringing constitutive dynamics to situated artificial institutions.
In *Proc. of 17th Portuguese Conference on Artificial Intelligence (EPIA 2015)*, volume 9273 of *LNCS*, pages 624–637. Springer, 2015.

Yves Demazeau.

From interactions to collective behaviour in agent-based systems.

In *Proceedings of the 1st. European Conference on Cognitive Science*, pages 117–132, 1995.

Mateus da Silveira Colissi, Renata Vieira, Viviana Mascardi, and Rafael H. Bordini.

A chatbot that uses a multi-agent organization to support collaborative learning.

In Constantine Stephanidis, Margherita Antona, and Stavroula Ntoa, editors, *HCI international 2021 - posters*, pages 31–38, Cham, 2021. Springer International Publishing.

Bibliography x

Débora Engelmann, Juliana Damasio, Tabajara Krausburg, Olimar Borges, Mateus Colissi, Alison R Panisson, and Rafael H Bordini.

Dial4jaca–a communication interface between multi-agent systems and chatbots.

In *International conference on practical applications of agents and multi-agent systems*, pages 77–88. Springer, 2021.

Michael Fisher, Rafael H. Bordini, Benjamin Hirsch, and Paolo Torroni.

Computational logics and agents: A road map of current technologies and future trends.

Computational Intelligence, 23(1):61–91, 2007.

Bibliography xi

Michael Fisher.

Metatem: The story so far.

In Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, editors, *PROMAS*, volume 3862 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2005.

Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque.

Congolog, a concurrent programming language based on the situation calculus.

Artif. Intell., 121(1-2):109–169, 2000.

Bibliography xii

Jomi Fred Hübner, Olivier Boissier, Rosine Kitio, and Alessandro Ricci.

Instrumenting multi-agent organisations with organisational artifacts and agents: "Giving the organisational power back to the agents".

Journal of Autonomous Agents and Multi-Agent Systems, 20(3):369–400, 2010.

Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer.

Formal semantics for an abstract agent programming language.

In Munindar P. Singh, Anand S. Rao, and Michael J. Wooldridge, editors, *Intelligent Agents IV, Agent Theories, Architectures, and Languages, 4th International Workshop, ATAL '97, Providence,* *Rhode Island, USA, July 24–26, 1997, Proceedings*, volume 1365 of *LNCS*, pages 215–229. Springer, 1997.

Koen V. Hindriks.
Programming rational agents in GOAL.
In Bordini et al. [BDDEFS05], pages 119–157.

Michael N. Huhns.
Interaction-oriented programming.
In *First international workshop, AOSE 2000 on Agent-oriented software engineering*, pages 29–44, Secaucus, NJ, USA, 2001. Springer-Verlag New York, Inc.

Marcelo Souza Menegol, Jomi Fred Hübner, and Leandro Buss Becker.

Coordinated UAV search and rescue application with JaCaMo.

In Yves Demazeau, Bo AnJavier Bajo, and Antonio Fernández-Caballero, editors, *Proc. of 16th International Conference International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2018)*, volume 10978 of *LNCS*, pages 335–338, 2018.

Bonnie A Nardi.

Context and consciousness: Activity theory and human-computer interaction.

mit Press, 1996.

Bibliography xv

Fabio Y. Okuyama, Rafael H. Bordini, and Antônio Carlos da Rocha Costa.

A distributed normative infrastructure for situated multi-agent organisations.

In Matteo Baldoni, Tran Cao Son, M. Birna van Riemsdijk, and Michael Winikoff, editors, *DALT*, volume 5397 of *Lecture Notes in Computer Science*, pages 29–46. Springer, 2008.

Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf.

Jadex: A BDI reasoning engine.

In Bordini et al. [BDDEFS05], pages 149–174.

Bibliography xvi

Michele Piunti, Alessandro Ricci, Olivier Boissier, and Jomi F. Hübner.

Embodied organisations in mas environments.

In Lars Braubach, Wiebe van der Hoek, Paolo Petta, and Alexander Pokahr, editors, *Proceedings of 7th German conference on Multi-Agent System Technologies (MATES 09), Hamburg, Germany, September 9-11*, volume 5774 of *LNCS*, pages 115–127. Springer, 2009.

David V. Pynadath, Milind Tambe, Nicolas Chauvat, and Lawrence Cavedon.

Toward team-oriented programming.

In Nicholas R. Jennings and Yves Lespérance, editors, *ATAL*, volume 1757 of *LNCS*, pages 233–247. Springer, 1999.

Anand S. Rao.
Agentspeak(l): Bdi agents speak out in a logical computable language.
In Walter Van de Velde and John W. Perram, editors, *MAAMAW*, volume 1038 of *LNCS*, pages 42–55. Springer, 1996.

A. S. Rao and M. P. Georgeff.
BDI agents: from theory to practice.
In Victor Lesser, editor, *Proceedings of the First International Conference on MultiAgent Systems (ICMAS'95)*, pages 312–319. AAAI Pess, 1995.

Bibliography xviii

Alessandro Ricci, Michele Piunti, and Mirko Viroli.
Environment programming in multi-agent systems – an artifact-based perspective.
Autonomous Agents and Multi-Agent Systems, June 2010.
Published Online with ISSN 1573-7454 (will appear with ISSN 1387-2532).

Alessandro Ricci, Michele Piunti, Mirko Viroli, and Andrea Omicini.
Environment programming in cartago.
In *Multi-Agent Programming: Languages,Platforms and Applications,Vol.2*. Springer, 2009.

Alessandro Ricci, Michele Piunti, Mirko Viroli, and Andrea Omicini.
***Multi-Agent Programming: Languages, Tools and Applications*, chapter Environment Programming in CArtAgO, pages 259–288.**
Springer US, 2009.

Bibliography xix

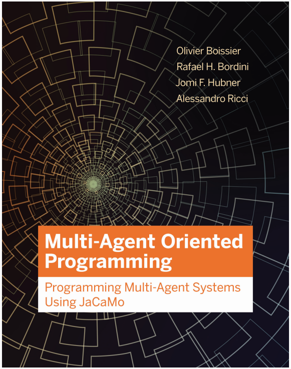
- Yoav Shoham.
  Agent-oriented programming.
  Artif. Intell., 60(1):51–92, 1993.

- Michael Winikoff.
  JACK intelligent agents: An industrial strength platform.
  In Bordini et al. [BDDEFS05], pages 175–193.

Further Resources

- `https://github.com/jacamo-lang/jacamo/`

- O. Boissier, R.H. Bordini, J.F. Hübner, and A. Ricci
  Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo
  MIT Press, 2020.


Olivier Boissier
Rafael H. Bordini
Jomi F. Hubner
Alessandro Ricci
Multi-Agent Oriented Programming
Programming Multi-Agent Systems Using JaCaMo

Book Web Site

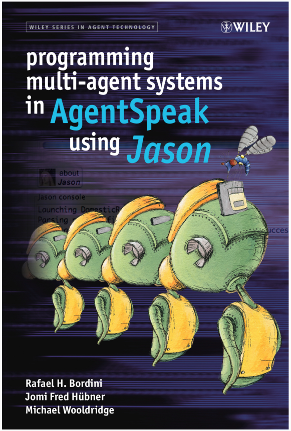
The book is accompanied by a website `http://jacamo.sourceforge.net/book` from which can be downloaded and run:

- all the examples
- and complete systems

Further Resources

- https://github.com/jason-lang/jason/
- http://jason.sourceforge.net

- R.H. Bordini, J.F. Hübner, and M. Wooldrige *Programming Multi-Agent Systems in AgentSpeak using Jason* John Wiley & Sons, 2007.


WILEY SERIES IN AGENT TECHNOLOGY
WILEY
programming multi-agent systems in AgentSpeak using Jason
Rafael H. Bordini
Jomi Fred Hübner
Michael Wooldridge