LEVELS OF ABSTRACTIONS IN DESIGNING & PROGRAMMING SYSTEMS OF COGNITIVE AGENTS

A. Ricci
DISI, University of Bologna
OBJECTIVE
OBJECTIVE

• Some glances about Agent-Oriented Programming and Multi-Agent Oriented Programming
  - examples using JaCaMo framework/technology
OBJECTIVE

- Some glances about Agent-Oriented Programming and Multi-Agent Oriented Programming
  - examples using JaCaMo framework/technology
- Main viewpoint
  - Level of abstraction, from design to runtime through programming
OBJECTIVE

- Some glances about Agent-Oriented Programming and Multi-Agent Oriented Programming
  - examples using JaCaMo framework/technology
- Main viewpoint
  - Level of abstraction, from design to runtime through programming
- Some points for the discussion
  - AOP and (M)AOP for the web/hypermedia?
AGENT-ORIENTED PROGRAMMING

- AI view
  - modeling/designing/programming autonomous systems, referred as agents
- SE view
  - using agents as first-class modeling/designing/programming abstraction
AGENT ABSTRACTION
AGENT ABSTRACTION
AGENT ABSTRACTION

- task/goal-oriented
- pro-active + reactive
- decision making
PARADIGMS & METAPHORS

imperative => machines

functional => math

OOP => world of objects

agents =>
PARADIGMS & METAPHORS

imperative => machines
functional => math
OOP => world of objects
agents => world of humans
ACTIONS & PERCEPTS
ACTIONS & PERCEPTS

- control uncoupling
  - action execution model is asynchronous
    - success/failure events
  - percepts as obs state events
ACTIONS & PERCEPTS

- control uncoupling
  - action execution model is asynchronous
    - success/failure events
  - percepts as obs state events
- vs. other models
  - vs. method/proc calls
  - vs. async msg (actor) passing
AGENT COMMUNICATION
• Agent Communication

Languages
- “speech acts”
- ~asynchronous message passing + action semantics
Agent Communication

- **Languages**
  - “speech acts”
  - ~asynchronous message passing + action semantics

- **vs. other model**
  - vs. async (actor) msg passing
“COGNITIVE” MODEL
“COGNITIVE” MODEL

• AOP as a computing paradigm
  - *mentalistic* and *societal* view of computation [Soham, 1993]
  - level of abstraction to design and program
“COGNITIVE” MODEL

• AOP as a computing paradigm
  - *mentalistc* and *societal* view of computation [Soham, 1993]
  - level of abstraction to design and program

• BDI (Belief-Desire-Intention) model/architecture (80ies)
  - inspired by the theory of human practical reasoning [Bratman, 1987]
  - Procedural Reasoning System (PRS) [Georgeff et al, 1988]
“COGNITIVE” MODEL

- Beliefs
  - information state
- Goals
  - tasks to do
  - achieve | maintenance
- Plans
  - how to achieve the goals
  - modules of agent behaviour
“COGNITIVE” MODEL

• **Beliefs**
  - information state

• **Goals**
  - tasks to do
  - achieve | maintenance

• **Plans**
  - how to achieve the goals
  - modules of agent behaviour

![Diagram](image)
"COGNITIVE" MODEL

• Beliefs
  - information state

• **Goals**
  - tasks to do
  - achieve | maintenance

• Plans
  - how to achieve the goals
  - modules of agent behaviour

---

// examples in Jason

!achieve_temp(20).

/* more declarative style */
!temp(20).

/* long-term task */
!achieve_and_keep_temp(20).
“COGNITIVE” MODEL

• Beliefs
  - information state
• Goals
  - what tasks to do
  - achieve | maintenance
• Plans
  - how to achieve the goals
  - modules of agent behaviour
• Plan model
  - pro-active plans
  - reactive plans
• Hierarchical model
  - sub-goals

<event> : <context>
  <- <body>.

PLAN MODEL | JASON EXAMPLE
PLAN MODEL | JASON EXAMPLE

- Plan model
  - **pro-active plans**
  - reactive plans
- Hierarchical model
  - sub-goals

```plaintext
%!achieve_temp(Target):
  temp(Current) & Target > Current
  <- startWarming;
  !warm_until(Target).

%!achieve_temp(Target):
  temp(Current) & Target < Current
  <- startCooling;
  !cool_until(Target).

%!achieve_temp(Target):
  temp(Current) & Target == Current
  <- stopHVAC.
```
• Plan model
  - pro-active plans
  - reactive plans
• Hierarchical model
  - sub-goals

// long-term / maintenance task
// target(T): belief used to track
// the target temperature

+temp(Current):
  target(Target) & Target != Current
  <- !achieve_temp(Target).
Plan model
  - pro-active plans
  - reactive plans
Hierarchical model
  - sub-goals

+!achieve_temp(Target):
  temp(Current) & Target < Current
  <- startWarming;
    !warm_until(Target).

+!achieve_temp(Target):
  temp(Current) & Target > Current
  <- startCooling;
    !cool_until(Target).

+!achieve_temp(Target):
  temp(Current) & Target == Current
  <- stopHVAC.
“COGNITIVE” MODEL

- **Intention**
  - a plan in execution
  - can fail => plan failure handling
  - can be inspected, suspended, resumed, aborted
  - multiple intentions can be in execution concurrently

```
+!achieve_temp(Target) :
  temp(Current) & Target < Current
  :- startWarming;
  !warm_until(Target).

-!achieve_temp(Target) :
  :- print(“broken”);
  send_email.

+!warm_until(Target) :
  temp(Current) & Current > Target
  :- .drop_intention(warm_until);
  !achieve_temp(Target).
```
“COGNITIVE” MODEL

• **Reflection/meta-level features**
  ‣ adding/changing plans at runtime
  ‣ inspecting/changing motivation state
  ‣ ...

// adding a plan action
+!g1 <-
... .add_plan("+b : true <- .print(b).”);
...

// checking for an intention
+!g1 : .intend(g2)
<- ... .suspend_intention(g2);
...
CONTROL LOOP | REASONING CYCLE

- Abstract/general
CONTROL LOOP | REASONING CYCLE

- Abstract/general

- (BDI) Reasoning cycle

```
begin
  while true do
    p ← perception()
    B ← brf(B, p) ; // belief revision
    D ← options(B, I) ; // desire revision
    I ← filter(B, D, I) ; // deliberation
    execute(I) ; // means-end
  end
end
```
CONTROL LOOP | REASONING CYCLE

- Abstract/general

- (BDI) Reasoning cycle

```plaintext
while true do
  B ← brf(B, perception())
  D ← options(B, I)
  I ← filter(B, D, I)
  π ← plan(B, I, A)
  while π ≠ ∅ and ¬succeeded(I, B) and ¬impossible(I, B) do
    execute( head(π) )
    π ← tail(π)
    B ← brf(B, perception())
    if reconsider(I, B) then
      D ← options(B, I) ;
      I ← filter(B, D, I) ;
      if ¬sound(π, I, B) then
        π ← plan(B, I, A) ;

PLANS and PLAN LIBRARY
means-end in BDI =>
- get and exec a plan
from a plan library

revise commitment to plan – re-
planning for context adaptation

reconsider the intentions
```
AGENTS vs. OBJECTS/ACTORS
AGENTS vs. OBJECTS/ACTORS

- vs. objects in OOP
  - active vs. passive
  - stronger encapsulation
    ‣ state + behaviour + *control* of the behaviour
    ‣ “decision making”
AGENTS vs. OBJECTS/ACTORS

• vs. objects in OOP
  - active vs. passive
  - stronger encapsulation
    ‣ state + behaviour + control of the behaviour
    ‣ “decision making”

• vs. actors
  - not reactive but pro-activity
    ‣ reasoning cycle vs. event-loop
    ‣ task/goal-oriented vs. message-driven
FROM AOP TO MAOP
(“MULTI-AGENT ORIENTED PROGRAMMING”)
FROM AOP TO MAOP
(“MULTI-AGENT ORIENTED PROGRAMMING”)

• Integrating further design & programming dimensions and abstractions aside to agent [Boissier et al, JaCaMo papers]
  - environment dimension
  - organisation dimension
FROM AOP TO MAOP
(“MULTI-AGENT ORIENTED PROGRAMMING”)

• Integrating further design & programming dimensions and abstractions aside to agent [Boissier et al, JaCaMo papers]
  - **environment** dimension
  - **organisation** dimension

• Key points
  - separation of concerns
  - again: level of the abstraction
    ‣ i.e. away from the *everything-is-an-agent* perspective
ENVIRONMENT
AS FIRST-CLASS DIMENSION
• Environment as first-class design/programming abstraction
  - modularising functionalities and services available to agents
ENVIRONMENT
AS FIRST-CLASS DIMENSION

• Environment as first-class design/programming abstraction
  - modularising functionalities and services available to agents
• JaCaMo: A&A model (Agents & Artifacts)
  - inspired by Activity Theory & Distributed Cognition
  - environment as a dynamic set of artifacts
    ‣ created/used/shared by agents
    ‣ tools mediating agent activities
    ‣ ~objects at the agent LoA
ARTIFACT ABSTRACTION
ARTIFACT ABSTRACTION

- Artifact first-class abstraction
• Artifact first-class abstraction
  - usage interface
    ‣ operations (~actions)
    ‣ observable properties
- Artifact first-class abstraction
  - *usage interface*
    - *operations* (~actions)
    - *observable properties*
  - *link interface*
    - to connect artifacts

**ENVIRONMENT**

- **c0: Counter**
  - inc
  - count/1
  - tick

- **h: HVAC**
  - temp/1
  - startC
  - startW
  - off
  - temp/1
  - startC
  - startW
  - off

- **b: WhiteBoard**
  - write
  - read

- **g: GUI**
  - setText
  - edit/1
  - pressed

**AGENT**
• Artifact first-class abstraction
  - **usage interface**
    ‣ **operations** (~actions)
    ‣ **observable properties**
  - **link interface**
    ‣ to connect artifacts
  - **manual**
    ‣ what functionalities & how to use
• JACaMo environment
ARTIFACT ABSTRACTION

- JACaMo environment
  - Java-based API & runtime

```java
public class Counter extends Artifact {
    private int nTicks;
    void init() {
        defineObsProperty("count", 0);
        nTicks = 0;
    }
    @OPERATION void inc() {
        ObsProperty prop = getObsProperty("count");
        prop.updateValue(prop.intValue() + 1);
        nTicks++;
        signal("tick " + nTicks);
    }
}
```
WORKSPACES

- Structuring complex/distributed environments in *workspaces*
  - logical containers of artifacts
  - agents can dynamically join and work in multiple workspaces
  - workspaces can be distributed over the network
ORGANISATION AS FIRST-CLASS DIMENSION
ORGANISATION
AS FIRST-CLASS DIMENSION

- Organisation as first-class design/programming abstraction
  - specifying the structure and coordinated behaviour of a **MAS as a whole**
ORGANISATION
AS FIRST-CLASS DIMENSION

- Organisation as first-class design/programming abstraction
  - specifying the structure and coordinated behaviour of a **MAS as a whole**
- JaCaMo organisation
  - roles, links, groups
  - social goals, missions schemes
  - norms
ORGANISATION AS FIRST-CLASS DIMENSION

- Organisation as first-class design/programming abstraction
  - specifying the structure and coordinated behaviour of a **MAS as a whole**
- JaCaMo organisation
  - roles, links, groups
  - social goals, missions schemes
  - norms
- Tackling MAS-level complexity
  - coordination, openness, regulated autonomy
Simplified Conceptual View (MOISE meta-model [Hübner et al., 2009])

Excerpts from organisation program:

```
<structural-specification>
<role-definitions>
<role id="auctioneer" />
<role id="participant" />
</role-definitions>
</structural-specification>

<group-specification id="auctionGroup">
<roles>
<role id="auctioneer" min="1" max="1"/>
<role id="participant" min="0" max="300"/>
</roles>
</group-specification>
</structural-specification>

<functional-specification>
  <scheme id="doAuction">
    <goal id="auction">
      <argument id="Id" />
      <argument id="Service" />
      <plan operator="sequence">
        <goal id="start" />
        <goal id="bid" ttf="10 seconds" />
        <goal id="decide" ttf="1 hour" />
      </plan>
    </goal>
  </scheme>
</functional-specification>

<normative-specification>
  <norm id="n1" type="permission"
          role="auctioneer"
          mission="mAuctioneer" />
  <norm id="n2" type="obligation"
          role="participant"
          mission="mParticipant" />
</normative-specification>
```

Normative spec.

norm n1 : plays(A, auctioneer, G) -> forbidden(A, n1, plays(A, participant, G), _'forever').
Simplified Conceptual View (MOISE meta-model [Hübner et al., 2009])

Excerpts from organisation program:

- Explicit representation of the org
  - that agents can inspect, reason about, and change
- Reified at runtime through artifacts
  - that agents can monitor, manage, adapt

Structural spec.  Functional spec.  program in NPL
WRAP-UP

Organisation

- Group
- Scheme
  - Role
  - Norm
  - Goal

Environment

- Workspace
- Signal
  - Artifact
  - Operation
  - Property

Agent

- Goal
  - Action
  - Belief

Actions:
- empower
- count-as
- participate
- regulate
- coordinate
- act
- perceive
POINTS FOR DISCUSSION

• AOP/MAOP for WoX/Hypermedia (?)
  - benefits
  - open issues
• Roadmap?
  - call for action?