

From Objects to Agents: Agent-Oriented Programming for Software Development

Alessandro Ricci¹, **Andrea Santi**¹
{a.ricci@unibo.it, a.santi@unibo.it}

¹Università degli studi di Bologna, DEIS - Cesena

FCP, 26/09/2011

Outline

- 1 Introduction
- 2 A Background Metaphor
- 3 Evaluating the Idea
 - The JaCa platform
 - Practical Examples
- 4 Conclusions

Developing Modern Software Systems: Challenges

- Realising modern software systems is an hard task
- Almost every modern application has to deal with issues such as
 - Concurrency
 - Distribution
 - Decentralisation of control
 - ...
- Proliferation of new programming languages/frameworks aiming at ease this task
 - Scala, Groovy, Clojure, etc.
 - Also rooted on not so mainstream (yet) programming models
 - The Actor-based one is a main example

Agent-Oriented Programming (AOP): the Current Situation

The notion of agent appears in several computer science contexts

- Often with different meanings
- Main acceptations are the (D)AI contexts
 - Agents as a special purpose technique to build intelligent systems

Our Perspective

AOP can be exploited for programming software systems in general

- Extending object/function-oriented programming
 - Thanks to an higher-level of abstraction
- Tackling main challenges of modern software development
 - Concurrency, decentralization of control, autonomy, adaptivity, ...

AOP for Software Development 1/2

- The idea of Agent-Oriented Programming is not new
 - The first paper about AOP is dated 1993 [Shoham, 1993]..
 - .. and since then many APLs/platforms have been proposed [Bordini et al., 2005, Bordini et al., 2009, Bordini et al., 2006]
- No significant impacts on mainstream research in programming languages and software development
 - Efforts/emphasis have been put on theoretical issues
 - No focus on principles of general-purpose computer programming

AOP for Software Development 2/2

We frame an unexplored perspective: agent-oriented abstractions for defining new general-purpose programming languages

The Objectives

- Exploit agent-oriented abstractions to develop real-world programs
- Pointing out
 - Related outcomes
 - Weaknesses and limitations..
 - .. addressing new issues for AOP: notion of types, reusability, etc.
- Set up the desiderata for a new generation of AOP languages
 - Focus on principles of general-purpose computer programming

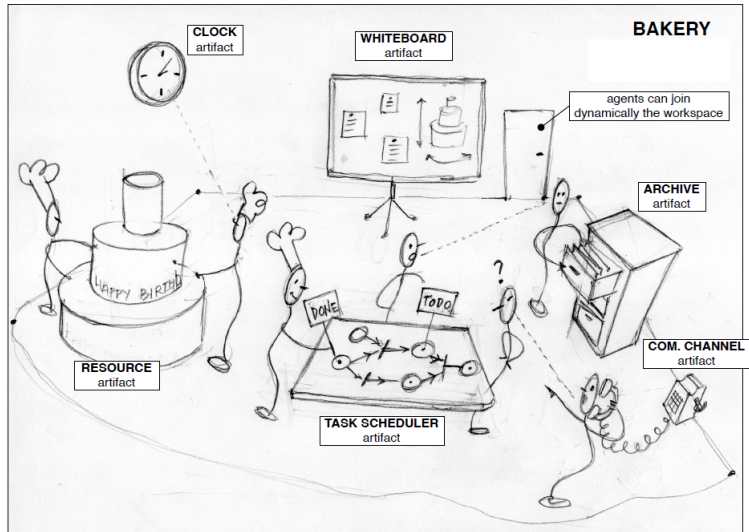
Outline

- 1 Introduction
- 2 A Background Metaphor
- 3 Evaluating the Idea
 - The JaCa platform
 - Practical Examples
- 4 Conclusions

Abstractions and Metaphors for Computer Programming

- OOP metaphor: real-world objects
 - Objects have properties and states
 - Objects can communicate and respond to communications
- Actors metaphor: anthropomorphic inspiration
 - A set of light-weight processes (actors)..
 - ..interacting only via asynchronous message-passing
- Our AOP metaphor (A&A meta-model)
 - Anthropomorphic inspiration too
 - Mimicking human cooperative work environments from Activity Theory [Nardi, 1996] and Distributed Cognition

The A&A Meta-Model: an Abstract Representation



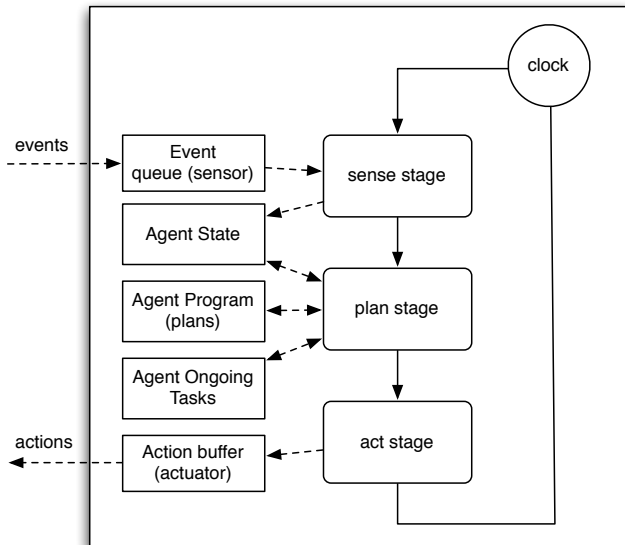
Agents, What Are They in Our Case?

- Autonomous, task-oriented pro-active entities
- Creating and co-using artifacts for supporting their activities
 - besides direct communication

Autonomous behaviour structured in terms of

- Tasks
 - Unit of work that has to be executed
 - Defining *what* have to be executed
- Plans
 - Actions to be performed for achieving a task
 - Defining *how* execute a task
- Actions
 - Basic moves the agent is able to perform
 - Environment actions (related to artifact operations)
 - Internal actions

Agents' Control Architecture



Working Environments in the A&A Meta-Model

- A first-class concept
- A software/computational layer
 - Defining the context of agents tasks
 - Providing and encapsulating (by means of artifacts)
 - Resources and services
 - State and processes
 - Ruling and mediating function
- Properly designed by the MAS engineer that defines
 - The structure and the topology of the environment..
 - .. and its desired behaviour

Artifacts in the A&A Meta-Model 1/2

- Non-autonomous, function-oriented entities
 - Controllable and observable by agents
- Modeling the tools and resources used by agents
 - Designed by MAS programmers
- Functionalities provided by artifact operations
 - Agents actions $\leftrightarrow$ Artifact operations [Ricci et al., 2010]
 - Operations define a proper action semantics [Ricci et al., 2010]
- Agents' perceptions $\leftrightarrow$ artifact observable properties and events [Ricci et al., 2010]
 - Directly mapped into agents' belief

Artifacts in the A&A Meta-Model 2/2

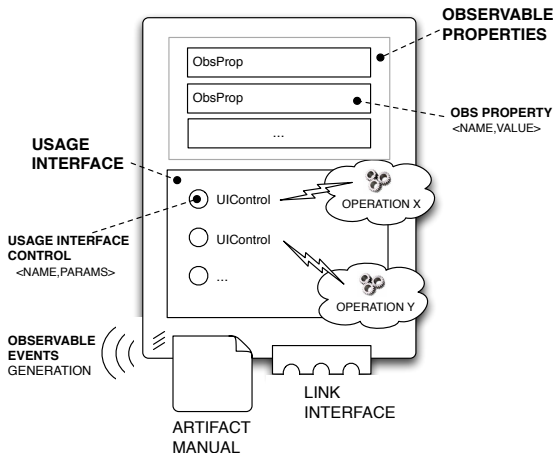


Figure: [Ricci et al., 2009]

Outline

- 1 Introduction
- 2 A Background Metaphor
- 3 Evaluating the Idea**
 - The JaCa platform
 - Practical Examples
- 4 Conclusions

Outline

- 1 Introduction
- 2 A Background Metaphor
- 3 Evaluating the Idea
 - The JaCa platform
 - Practical Examples
- 4 Conclusions

The JaCa platform

- Our *current* reference platform
- Birth from the synergistic integration of
 - *Jason* [Bordini et al., 2007] agent programming language
 - <http://jason.sourceforge.net/>
 - CArtaGO [Ricci et al., 2009] working environments
 - <http://cartago.sourceforge.net>

The JaCa Programming Model in a Glance

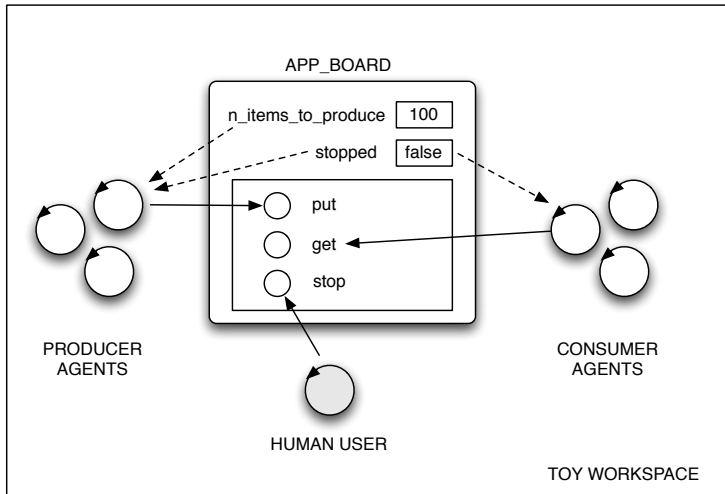
Applications realized in CArTAgO [Ricci et al., 2009] working environments where a set of *Jason* [Bordini et al., 2007] agents work together and interact creating, sharing and exploiting a dynamic set of artifacts

- **BDI Jason Agents** encapsulate the execution and the control of tasks that characterize the application scenario
- **Working environments** used as a first class abstraction
 - Encapsulating the resources and tools needed by agents
 - Allowing the design of a world aimed at the agent's use

Outline

- 1 Introduction
- 2 A Background Metaphor
- 3 Evaluating the Idea**
 - The JaCa platform
 - **Practical Examples**
- 4 Conclusions

User Controlled Producer-Consumer 1/2



User Controlled Producer-Consumer 2/2

```
00  n_items_produced(0).
01  !produce.
02
03  +!produce
04      <- !setup;
05          !produce_items.
06
07  +!produce_items : not n_items_to_produce(0)
08      <- !produce_item(Item); put(Item);
09      -+n_items_produced(N+1);
10      !produce_items.
11
12  +!produce_items : n_items_to_produce(0)
13      <- !finalize.
14
15  +!produce_item(Item) <- ...
16
17  +!finalize : n_items_produced(N)
18      <- println("completed - items produced: ",N).
19
20  +stopped(true)
21      <- .drop_all_intentions;
22      !finalize.
```

SMS Notification Manager 1/2

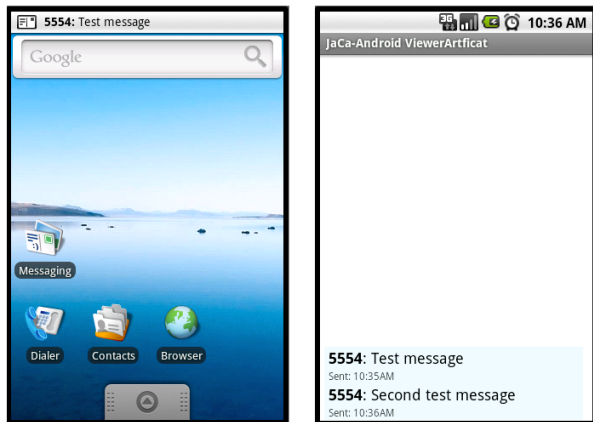


Figure: Two different kinds of SMS notifications: (a) notification performed using the standard Android status bar, and (b) notification performed using the ViewerArtifact.

SMS Notification Manager 2/2

```
00 !init.
01
02 +!init
03 <- focus("SMSArtifact");
04     focus("NotificationManager");
05     focus("ViewerArtifact").
06
07 +sms_received(Source, Message)
08   : not (state("running") & session(Source))
09   <- showNotification("jaca.android:drawable/notification",
10                     Source, Message, "jaca.android.sms.SmsViewer", Id);
11     append(Source, Message).
12
13 +sms_received(Source, Message) : state("running") & session(Source)
14   <- append(Source, Message).
```

Outline

- 1 Introduction
- 2 A Background Metaphor
- 3 Evaluating the Idea
 - The JaCa platform
 - Practical Examples
- 4 Conclusions

Summing Up

- We discussed agent-oriented programming as an evolution of Object-Oriented Programming
- Representing the essential nature of concurrent and decentralized systems
 - Where tasks are in charge of autonomous computational entities
 - Interacting and cooperating within a shared environment
- We showed in practice some of the main concepts underlying the approach by exploiting the JaCa platform

Considerations

- Decentralized mindset in programming [Resnick, 1994] thanks to
 - Encapsulation of control (into agents)
 - An interaction dimension rooted on two interaction typologies
 - Asynchronous message passing
 - Environment-mediated interactions
- An high-level rationale to structure autonomous behaviour
 - Through the notion of tasks and plans
- Integration of pro-active and reactive behaviour
- Avoidance of *"everything is an X"* perspective
 - Agents to model active parts, artifacts for passive ones

Weaknesses and Limitations

- Still no *explicit* notion of type
 - Neither for agents nor for artifacts
- Lack of a seamless integration with object/functional-oriented layer
 - We are currently using a custom library
- Weak modularisation provided by the plan construct
 - Absence of hierarchical structure for plans

Future Works

- Continue to evaluate the approach through existing technologies
 - JaCa and possible extensions
- A new generation of agent-oriented programming languages is needed
 - Tackling aspects not considered so far
 - Not related to (D)AI..
 - but to the principles of software development
 - Stressing and investigating the full value of the agent-oriented approach
- simpAL [Ricci and Santi, 2011]: Our first step towards a new generation APL

Bibliography I



Bordini, R., Braubach, L., Dastani, M., Seghrouchni, A., Gomez-Sanz, J., Leite, J., O'Hare, G., Pokahr, A., and Ricci, A. (2006).

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

In *Informatica* 30, pages 33–44.



Bordini, R., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., editors (2005).

Multi-Agent Programming Languages, Platforms and Applications - Volume 1, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Springer.

Bibliography II



Bordini, R., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., editors (2009).

Multi-Agent Programming Languages, Platforms and Applications - Volume 2, Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer.



Bordini, R., Hübner, J., and Wooldridge, M. (2007).

Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons, Ltd.



Nardi, B. (1996).

Activity theory and human-computer interaction.

Context and consciousness: Activity theory and human-computer interaction, pages 7–16.

Bibliography III



Resnick, M. (1994).

Turtles, Termites and Traffic Jams. Explorations in Massively Parallel Microworlds.

MIT Press.



Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009).

Environment programming in CArTAgO.

In Bordini, R. H., Dastani, M., Dix, J., and El Fallah-Seghrouchni, A., editors, *Multi-Agent Programming: Languages, Platforms and Applications, Vol. 2*, pages 259–288. Springer.



Ricci, A. and Santi (2011).

Designing a General-Purpose Programming Language based on Agent-Oriented Abstractions: The simpAL Project.

In *Proc. of the 1st Int. Workshop on Agents and Actors Reloaded – AGERE!*

Bibliography IV

 Ricci, A., Santi, A., and Pianti, M. (2010).

Action and Perception in Multi-Agent Programming Languages: From Exogenous to Endogenous Environments.

In The Eighth International Workshop on Programming Multi-Agent Systems ProMAS 2010, page 21.

 Shoham, Y. (1993).

Agent-oriented programming.

Artificial Intelligence, 60(1):51–92.