Exploiting Agent-Oriented Programming for Developing Android Applications

Andrea Santi[1] Marco Guidi[1] Alessandro Ricci[1]
a.santi@unibo.it, marco.guidi7@studio.unibo.it,
a.ricci@unibo.it

[1]Università degli studi di Bologna, DEIS - Cesena

WOA, 6/09/2010

Outline

AOP: the Current Situation

- The notion of agent and AOP appears in several computer science contexts
  - Often with different meanings
  - Main acceptations are the AI/DAI contexts
    - agents exploited as a technique to develop special-purpose/*intelligent* systems [Bordini et al., 2005, Bordini et al., 2009, Bordini et al., 2006]

- No significant impacts on mainstream research in programming languages and software development
  - Most efforts/emphasis have been put on theoretical issues
  - No focus on principles of general-purpose computer programming

Rebooting AOP for Software Development

- AOP/APLs could be exploited for programming software systems in general
  - Supporting a decentralized mindset in problem solving, designing systems, programming
  - Extending object/function-oriented programming
  - Tackling main challenges of modern software development
    - concurrency, decentralization of control, autonomy, adaptivity
- We refer to this as Agent Oriented Computing (AOC)

Why Agents and MAS as a Paradigm?

- Looking for a high-level abstraction level for computing, designing and programming systems
- Software and Concurrency revolution [Sutter and Larus, 2005]
  - *"the free lunch is over"*

Ongoing Research Lines

- Exploiting agent-oriented abstractions to develop real-world programs
  - Stressing existing technologies: JaCa platform
    - Applying it in modern application domains
  - Pointing out
    - Related outcomes
    - Weaknesses and limitations
- Devising a new language – simpAL
  - Focusing on principles/practices of general-purpose computer programming
  - Aiming at the diffusion of AOC as a mainstream programming paradigm

Outline

Modern Mobile Applications: Features & Complexities 1/2

- A new generation of mobile devices
  - Android-based devices
  - iPhone
  - MeeGo-based devices
- Radically changing the concept of smartphone thanks to
  - Increase in hardware specifications
  - The presence of every kind of known connectivity
    - Situating the device in a computational network..
    - ..analogous to the one promoted by Internet things/ ubiquitous computing vision
  - Extremely rapid O.S evolution

Modern Mobile Applications: Features & Complexities 2/2

- New perspectives and opportunities
  - New application scenarios
  - Applications become *nomadic* and ...
  - ... situated in both a physical and computational environment

New challenges in mobile application development

- New issues to be addressed
  - Concurrency
  - Asynchronous interactions
    - Web sites/Services, social-networks, messaging/mail clients, etc.
- The application becomes user-centric
- Context-sensitive behavior
  - Geographical position, presence/absence of connectivity..

Mainstream Technologies

- Apple iOS
  - Objective-C development framework
- MeeGo
  - Development framework Qt-based
- Android
  - New abstractions for the engineering of mobile applications
    - Activity, Service, Intent, ContentProvider...
  - But finally: still *yet another* Object-oriented (Java) framework
    - No a good solution for: reactivity, context-dependent behaviour..

Good Case Study for AOC

- The presented complexities can be tackled with AOC
  - Conceptually: thanks to a proper high-level abstraction layer
  - Practically: thanks to proper agent-based technologies
- Scaling to future mobile applications

Outline

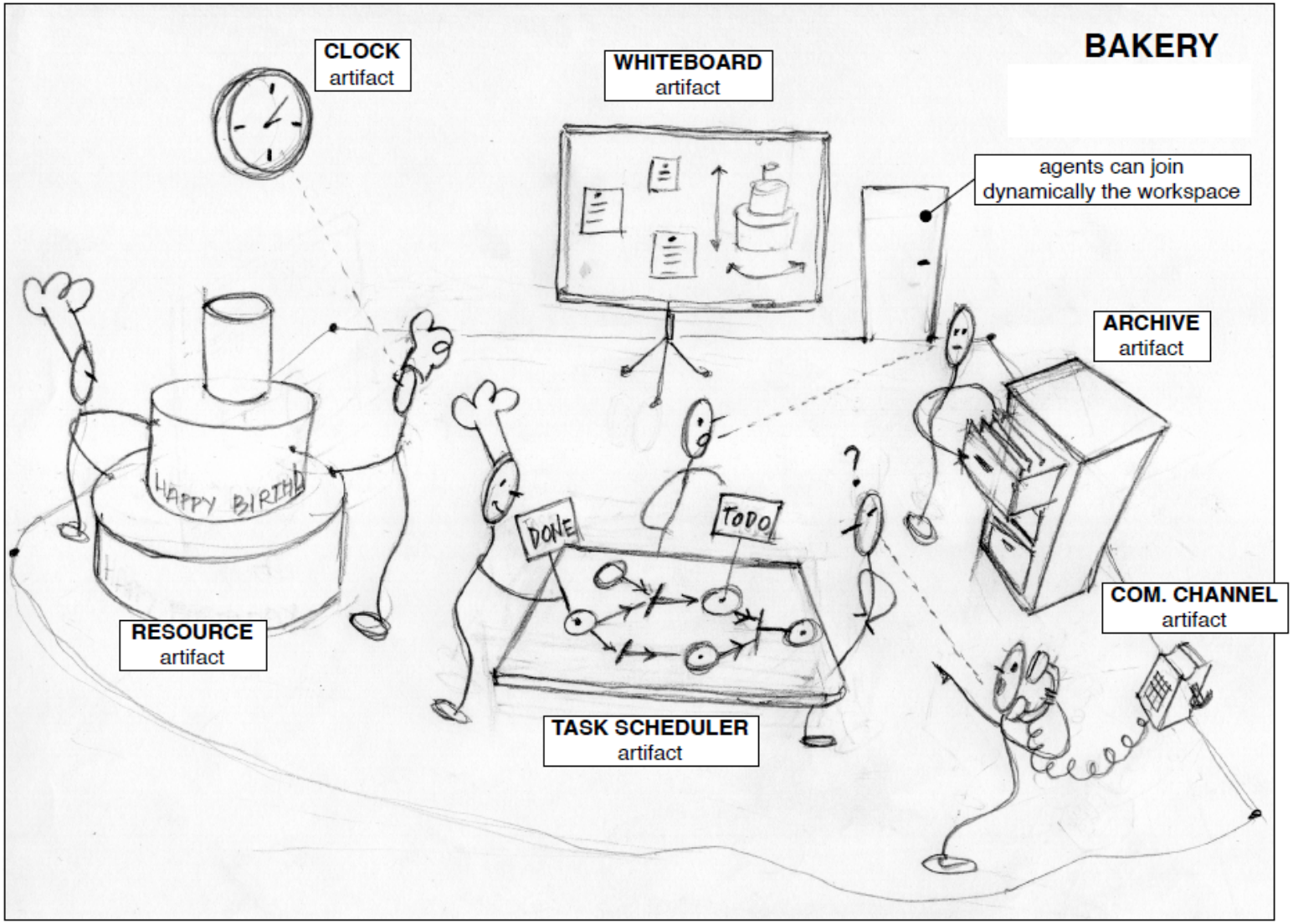
Outline

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 the business activities/tasks that characterize the application scenario
- **Working environments** used as a first class abstraction
  - Encapsulating the business resources and tools needed by agents to operate in the application domain
  - Allowing the design of a world aimed at the agent's use

JaCa Background Metaphor : an Abstract Representation



The JaCa Programming Model: Basic Abstractions

- Agents
  - tasks (goals), plans, beliefs, actions/perception
  - Direct communication through ACL
  - Indirect interaction through the environment
- Working Environment
  - artifacts in workspaces
  - resources and tools encapsulating the functionalities that can be shared and used by agents
  - operations, observable properties and signals
- Agent/Environment integration [Ricci et al., 2010]
  - Agents' actions $\leftrightarrow$ Artifacts' operations
  - Agents' beliefs $\leftrightarrow$ Artifacts' Percepts/observable properties/signal

JaCa in Modern and Relevant Application Domains

- Proper porting of the standard JaCa platform
- Introducing a set of specifics artifacts for the application domain

Existing Projects

- JaCa-Web: JaCa for developing rich Web Client applications
  - http://jaca-web.sourceforge.net/
- JaCa-WS: JaCa in the application context of SOA/WS applications
  - http://cartagows.sourceforge.net/
- JaCa-VM: JaCa for the development of virtualization applications
  - Still in early development stages

Outline

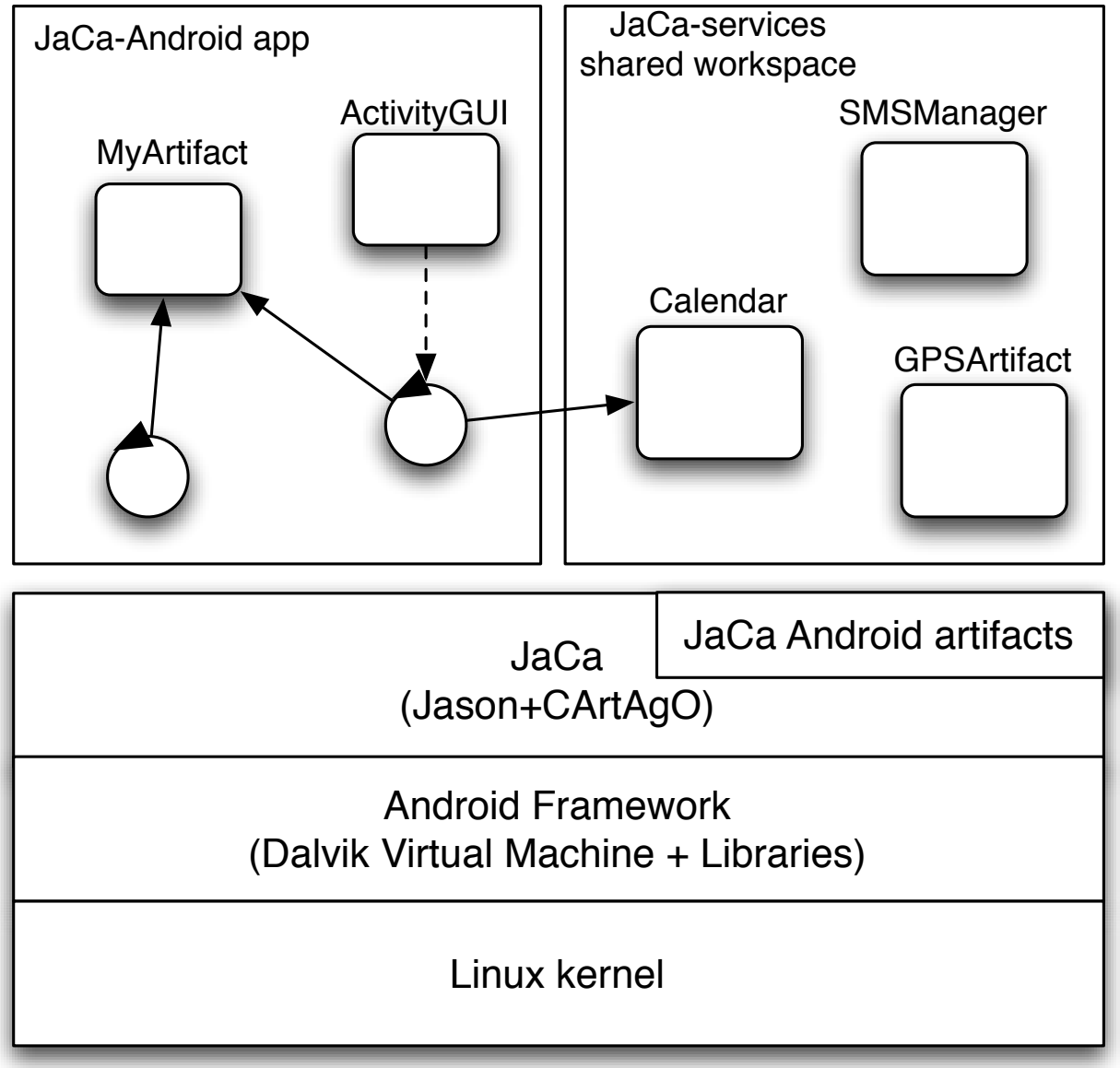
The JaCa-Android Platform

- Proper porting of the standard JaCa platform in the Android context
- Open-source project
  - http://jaca-android.sourceforge.net/
- Introducing a set of specifics artifacts
  - Standard Android components becomes fully-fledged artifacts
  - Allowing the development of Android applications at the agent level
- Seamless integration with existing Android application

Tackling Mobile Complexities with JaCa-Android

- Task/activity oriented behaviours directly mapped onto agents
  - Either using multiple agents
    - concurrently executing tasks
  - Or using a single agent
    - managing the interleaved execution of multiple tasks
- Context-sensitive behaviour
  - Thanks to agents' capability of adapting the behaviour on the basis of the current context information
- Managing of asynchronous interactions
  - properly specifying the agents reactive behaviour

The JaCa-Android Platform: an Abstract Representation



Outline

SMS Notification Manager 1/2

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

SMS Notification Manager 2/2

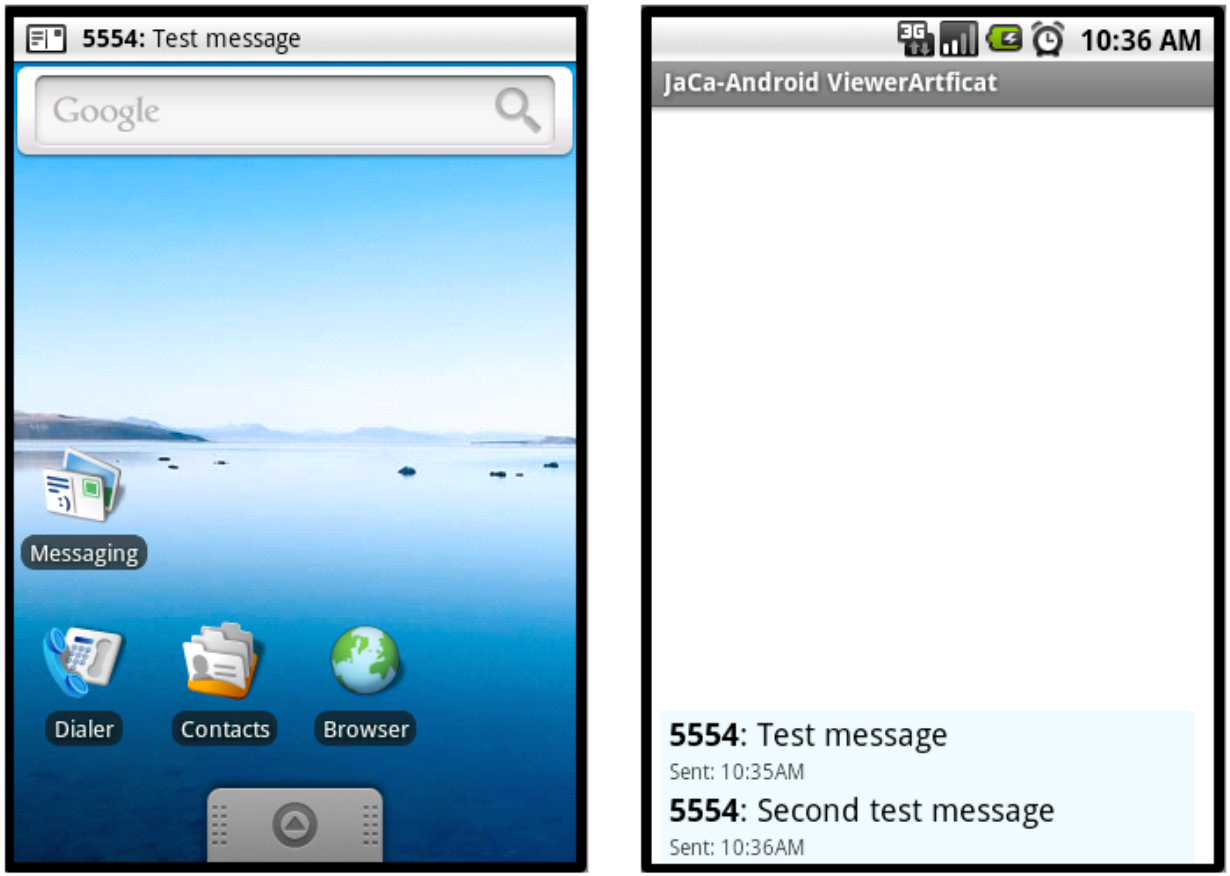


Figure: The two different kinds of SMS notifications: (*a*) notification performed using the standard Android status bar, and (*b*) notification performed using the `ViewerArtifact`.

A Smart Navigator 1/3

```
00  preferences([...]).
01  relevance_range(10).
02
03  !assist_user_trips.
04
05  +!assist_user_trips
06    <- focus("GPSManager"); focus("GoogleMapsArtifact");
07       focus("SmartNavigatorGUI"); focus("TrafficConditionsNotifier").
08
09  +route(StartLocation, EndLocation)
10    <- !handle_navigation(StartLocation, EndLocation).
11
12  +!handle_navigation(StartLocation, EndLocation)
13    <- ?relevance_range(Range);
14       ?current_position(Pos);
15       -+leaving(StartLocation);
16       -+arriving(EndLocation);
17       calculate_route(StartLocation, EndLocation, OutputRoute);
18       -+route(OutputRoute);
19       subscribe_for_traffic_condition(OutputRoute, Range);
20       set_current_position(Pos);
21       update_map.
```

A Smart Navigator 2/3

```
22  +new_traffic_info(TrafficInfo)
23    <- ?preferences(Preferences);
24       ?leaving(StartLocation);
25       ?arriving(EndLocation);
26       !check_info_relevance(TraffincInfo,Preferences);
27       !update_route(StartLocation, EndLocation, TrafficInfo, NewRoute);
28       !update_subscription(NewRoute);
29       update_map.
30
31  +current_position(Pos)
32    <- ?route(Route);
33       !check_position_consistency(Pos, Route);
34       set_current_position(Pos);
35       update_map.
36
37  -!check_position_consistency(Pos, Route) : arriving(EndLocation)
38    <- !handle_navigation(Pos, EndLocation).
```

A Smart Navigator 3/3

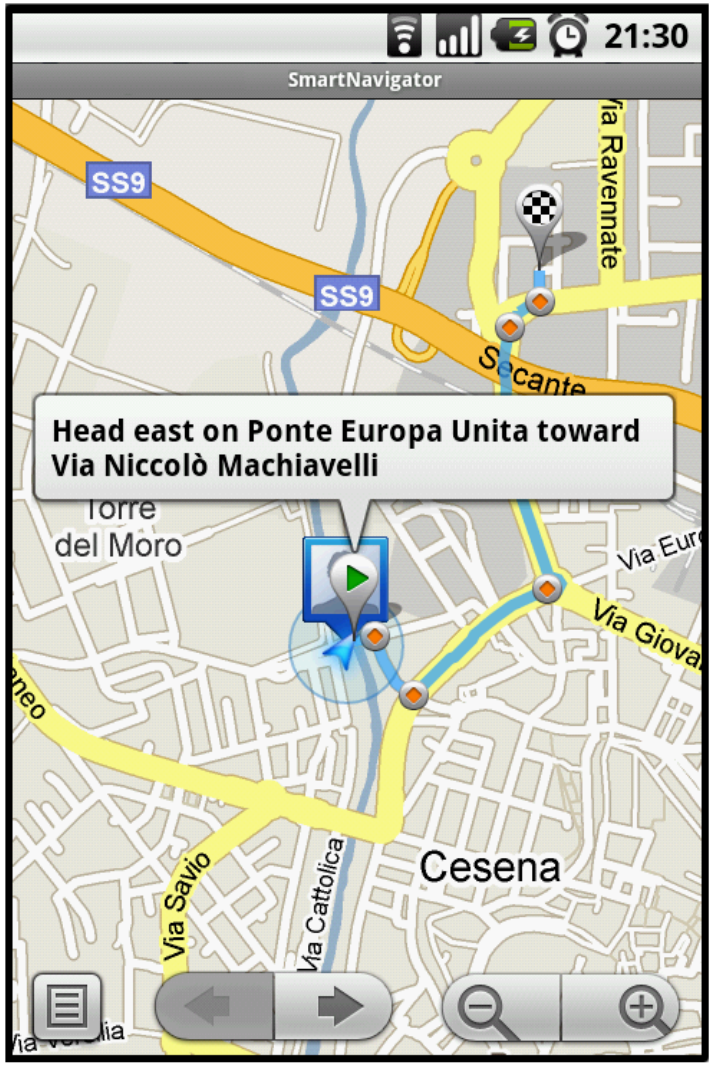


Figure: The GUI of the JaCa-Android SmartNavigator application.

Outline

Conclusions and Future Works 1/2

- 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
- Application in the mobile/nomadic application context
- Showing in practice main concepts underlying the approach
  - Exploiting the JaCa-Android platform

Conclusions and Future Works 2/2

- Issues that will be addressed in future JaCa-Android releases
  - Efficient management of the CPU workload
  - Smart use of the battery
- However, a new generation of agent-oriented programming languages is needed
  - To stress and investigate the full value of the agent-oriented approach
  - Tackling aspects not considered so far in existing agent technologies
  - Being not related to AI but to the principles of software development

Bibliography I


Berger, M., Rusitschka, S., Toropov, D., Watzke, M., and Schlichte, M. (2002).
Porting distributed agent-middleware to small mobile devices.
In *AAMAS Workshop on Ubiquitous Agents on Embedded, Wearable and Mobile Devices*.


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.

Bibliography II


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.


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.

Bibliography III


Koch, F., Meyer, J.-J. C., Dignum, F., and Rahwan, I. (2005).
Programming deliberative agents for mobile services: The 3apl-m platform.
In *PROMAS*, pages 222–235.


Muldoon, C., O'Hare, G. M. P., Collier, R. W., and O'Grady, M. J. (2006).
Agent factory micro edition: A framework for ambient applications.
In *Int. Conference on Computational Science (3)*, pages 727–734.


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.

Bibliography IV


Ricci, A., Santi, A., and Piunti, 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.


Sutter, H. and Larus, J. (2005).
Software and the concurrency revolution.
ACM Queue: Tomorrow's Computing Today, 3(7):54–62.