

Exploiting Intelligent Agent-Based Technologies for Programming Smart Mobile Applications{DEMO}

Andrea Santi¹, Alessandro Ricci,¹
{a.santi@unibo.it, a.ricci@unibo.it}

ALMA MATER STUDIORUM—Università di Bologna, DEIS³

AGERE!@SPLASH
24/10/2011

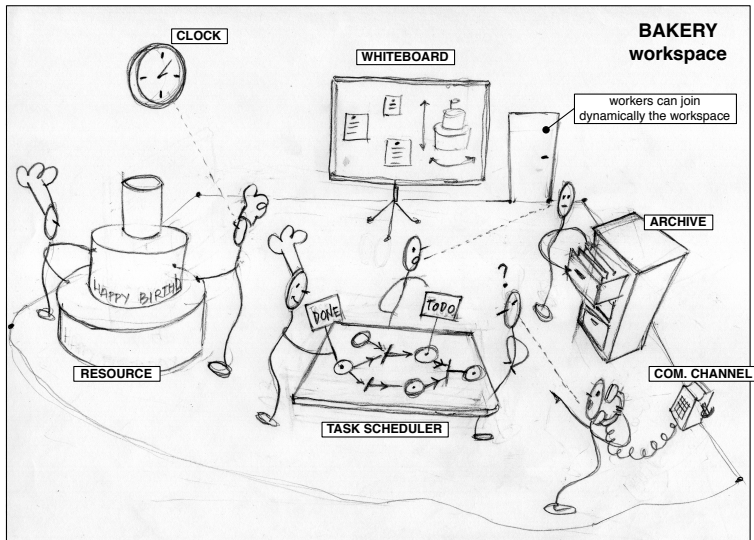
Developing Smart Mobile Applications: Challenges

- New generation mobile devices opens the way to a new generation of smart mobile applications [Landay et al., 2009, Wright, 2009]
- Featuring
 - Context-aware and geo-sensitive behaviour
 - Reactivity
 - Asynchronous interaction management
 - ...
- [Jennings, 2001, Bradshaw, 1997] Agents and multi-agent systems have been introduced as a paradigm for building complex software systems
 - Exhibiting some of the main features required to modern and fore-smart mobile applications

The JaCa-Android Platform

- A platform for building and running smart mobile applications
- Rooted on
 - An agent-oriented level of abstraction and ...
 - ... concrete agent-based technologies
- Built upon the synergistic integration of
 - Jason* [Bordini et al., 2007] a BDI-based APL
 - CARtAgO* [Ricci et al., 2009] a framework for programming agents' working environments
- Aiming at ease the design and programming of current and fore-coming smart mobile applications
- Open source research project
 - <http://jaca-android.sourceforge.net>

Background Metaphor: an Abstract Representation

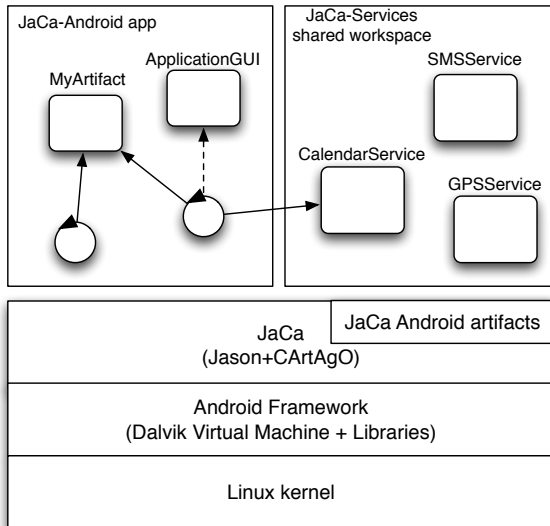


The JaCa Programming Model in a Glance

Mobile application realized as a CArTAgO [Ricci et al., 2009] working environment where a set of *Jason* [Bordini et al., 2007] agents work together and interact exploiting and sharing a dynamic set of artifacts

- **BDI [Rao and Georgeff, 1995] Jason Agents** encapsulate the execution and the control of the tasks in charge of the mobile application
- **Working environments** encapsulate the resources and tools agents need by means of artifacts
 - Providing functionalities available in the smartphone (e.g. GPS, Sensors, etc.)
 - Allowing to build/interact with compliant Android components (e.g. Activity, Service, etc.)

The JaCa-Android Platform: an Abstract Representation



Key Features

- Easy management of asynchronous interactions
 - By specifying agents' reactive behaviour in relation to the reception of particular events
- Integration of reactive and pro-active behavior
 - Thanks to agents's control architecture
- Quite seamless support to context-sensitive behavior management
 - Thanks to agents' capability of adapting their behaviour on the basis of current context information

What You Will See in The Demo

- A concrete taste of the JaCa-Android programming model
- A set of application samples
 - To prove in practice the effectiveness of the platform
- A set of basic performance and memory occupation analysis highlighting
 - The usability of the platform
 - Application performance *comparable* to classical Android applications

Bibliography I



Bordini, R., Hübner, J., and Wooldridge, M. (2007).
Programming Multi-Agent Systems in AgentSpeak Using Jason.
John Wiley & Sons, Ltd.



Bradshaw, J. M., editor (1997).
Software agents.
MIT Press, Cambridge, MA, USA.



Jennings, N. R. (2001).
An agent-based approach for building complex software systems.
Commun. ACM, 44(4):35–41.



Landay, J. A., Joseph, A. D., and Reynolds, F. (2009).
Guest editors' introduction: Smarter phones.
IEEE Pervasive Computing, 8:12–13.

Bibliography II



Rao, A. S. and Georgeff, M. P. (1995).
BDI Agents: From Theory to Practice.
In First International Conference on Multi Agent Systems (ICMAS95).



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.



Wright, A. (2009).
Get smart.
Commun. ACM, 52:15–16.

Programming the Agent in *Jason*

Jason is an APL based on the BDI model [Rao and Georgeff, 1995]

Agents are designed and programmed on the base of

Beliefs information the agent has about the world and other agents

Goals set of objectives the agent wants to achieve

Plans procedural knowledge (course of actions) an agent can exploit in order to fulfill some task

Agent Plan Structure

A *Jason* plan has the following general structure:

event : **context** $\leftarrow$ **body**.

Where:

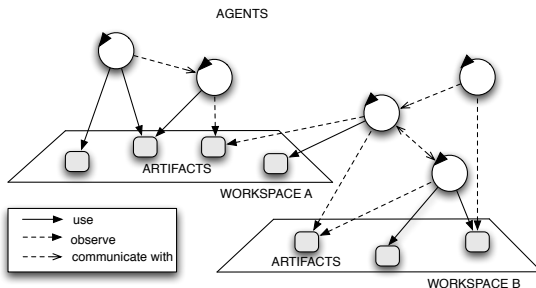
event The event the plan is meant to handle. It could be either a goal addition/failure event or an event generated from the environment (a signal or an obs property update)

context The condition that specifies when the plan can be used

body The course of actions to be executed to handle the **event** (if the **context** is believed true)

Programming the Environment in CArtAgO

- Structured in terms of multiple (distributed) workspace
- Provide agents the tools and resource they need by means of artifacts
- Artifacts are non-autonomous function-oriented entities
 - Controllable and observable by agents
 - Functionalities provided by artifact operations
 - State exposed by means of observable properties
 - Signal generation for notifying notable events



Artifact in a Nutshell

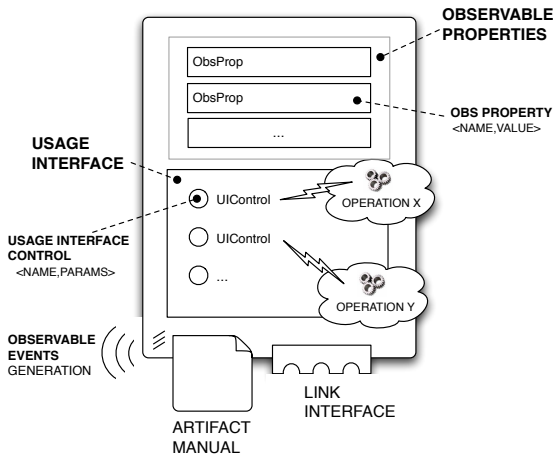


Figure: [Ricci et al., 2009]

Context-Aware SMS Notification Manager 1/2

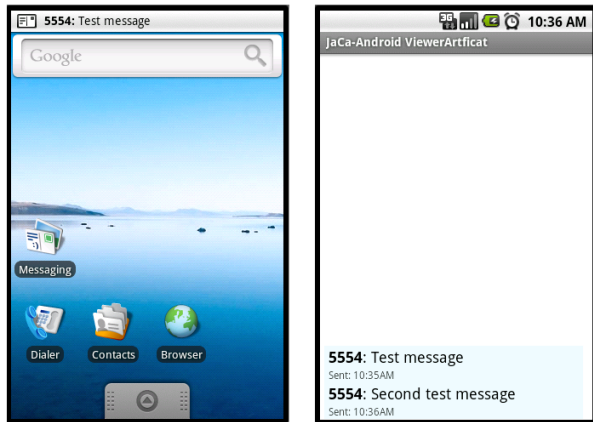


Figure: Two different kinds of SMS notifications: (a) notification performed using the standard Android status bar, and (b) notification performed using the ViewerArtifact.

Context-Aware SMS Notification Manager 2/2

```
sms_received(0).
my_source("4455").
source_relevant(Source) :- my_source(S) & S == Source.

!manage_sms.
+!manage_sms
  <- makeArtifact("smsreceiver","jaca...SMSManager", [true], IdRec);
    focus(IdRec);
    makeArtifact("notificator", "jaca...NotificationManager", [], IdNotif);
    lookupArtifact("viewer",IdViewer);
    focus(IdViewer).

+sms_received(Source, Message) : sms_received(NumSms) & source_relevant(Source)
  <- .println("[AG:] new sms received");
    -+sms_received(NumSms+1);
    !notify_sms(Source, Message).

+!notify_sms(Source, Message) : viewer_state("not_displayed")
  <- .println("[AG:] notification in status bar");
    showNotification(Source, Message, "jaca....SmsViewer", false, Id);
    addSMSToList(Source, Message).

+!notify_sms(Source, Message) : viewer_state("displayed")
  <- addSMSToList(Source, Message).
```