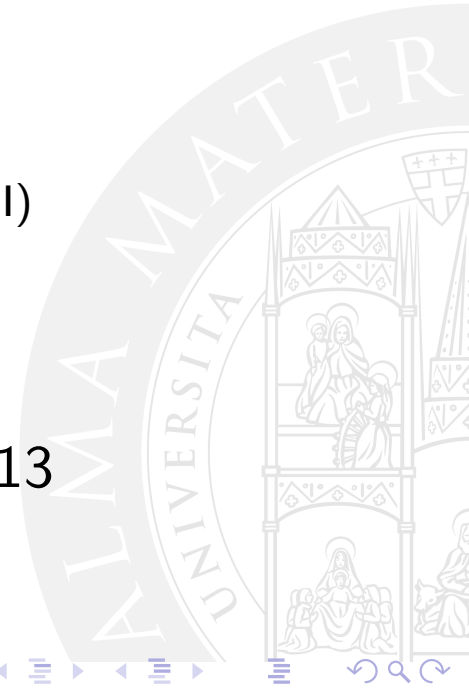
TuCSoN on Cloud: An Event-driven Architecture for Embodied / Disembodied Coordination

Stefano Mariani Andrea Omicini

{s.mariani, andrea.omicini}@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum—Università di Bologna

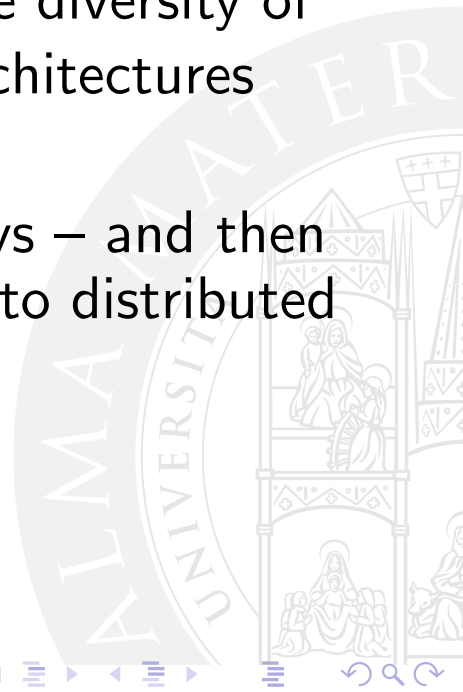
C-SmartCPS @ ICA3PP 2013
Vietri sul Mare, SA, Italy – 20 December 2013



Outline

Motivations I

- Multi-agent systems (MAS) provides abstractions and technologies so as to deal with complex application scenarios [Omicini and Mariani, 2013a]
- In turn, coordination *models*, *languages*, and *infrastructures* are deeply influenced by the evolution of MAS deployment scenarios [Omicini, 2013]
- *Heterogeneity* of application scenarios translates in the diversity of MAS (and coordination) abstractions, models, and architectures available
- In particular, *complex computational systems* nowadays – and then MAS – typically feature two quite diverse approaches to distributed systems:
  - pervasive computing [Zambonelli et al., 2011]
  - cloud computing [Hill et al., 2013]



Motivations II

Embodied / disembodied systems

- The two computational approaches are apparently opposite ones:
  - pervasive computing is the paradigm of embodied systems
  - cloud computing is the paradigm of disembodied systems
- A fundamental issue is how to reconcile the two technological paradigms

Goals

We believe that *embodied systems* – as pervasive ones – and *disembodied systems* – as cloud-based ones – should be viewed and dealt with as representing complementary approaches, to be exploited in synergy so as to better tackle real-world problems with the most suitable *degree of situatedness*.

An architecture for integration

Accordingly, we

- first propose an event-driven architecture, along with a coherent event model, promoting the *integration* of embodied and disembodied coordinated systems
- then *test it* so as to bring the TuCSoN coordination technology to the cloud

Outline

Issues

In order to obtain a coherent framework, from a technical viewpoint, a number of issues should be addressed

Identification & reference

How to *characterise* both embodied and disembodied entities within a system, and how to *refer* them

Modelling

Which abstractions should be used to deal with both embodied and disembodied computations

Architecture

Which architecture should be used to support both embodied and disembodied computations

Outline

Names, IDs & Attributes I

Identification & reference

- In any sort of distributed systems, a computational entity (either disembodied or embodied) should be identified by means of a global, univocal identifier—hence being accessible through *white-pages*-like services
- Also, *resources* (shaping computational *environmental*) could be referenced through their *properties* – typically represented in the form of *key-value* pairs – —hence being accessible through *yellow-pages*-like services

Names, IDs & Attributes II

Embodied references

In particular, in the case of embodied systems, an entity can be referenced also according to its *spatio-temporal properties*, used as *mutable attributes*:

spatial reference — any spatial characterisation identifying a set of nodes whose chosen spatial property has the same value—e.g., "all the nodes within range (X, Y, Z) from node n", but also "all nodes at latitude X, longitude Y, altitude Z"

temporal reference — any temporal characterisation identifying a set of nodes whose chosen temporal property has the same value—e.g., "all the nodes in range δ from instant t", but also "all nodes at instant t"

Outline

A Reconciling Abstraction I

- Most of the complexity of nowadays computational systems comes from interaction [Goldin and Wegner, 2008]
- Focussing on interaction, computational systems can be conceptualised as coordinated MAS [Ciancarini et al., 2000]
- The core entity in coordinated MAS is the event, working as the *connector* in the coordinated architecture
  [Mariani and Omicini, 2013a, Fortino et al., 2010]
  - everything occurring in a coordinated MAS generates an event, be it
    - a *coordination operation* request issued by an agent
    - a change in *environmental* properties
    - a change in the *space-time fabric*
  - events should be handled so as to keep track of all the related information, such as the *cause* of the event, and its *context*—who did it, toward whom it has been done, when and where, what its outcome is [Omicini, 2007, Omicini et al., 2005, Casadei and Omicini, 2009, Mariani and Omicini, 2013b]

A Reconciling Abstraction II

Embodied / disembodied coordination: the event model

$$
\begin{aligned}
\langle \mathit{Event} \rangle &::= \langle \mathit{Start} \rangle , \langle \mathit{Cause} \rangle , \langle \mathit{Evaluation} \rangle \\
\langle \mathit{Start} \rangle , \langle \mathit{Cause} \rangle &::= \langle \mathit{Activity} \rangle , \langle \mathit{Source} \rangle , \langle \mathit{Target} \rangle , \langle \mathit{Time} \rangle , \langle \mathit{Space} : \mathit{Place} \rangle \\
\langle \mathit{Source} \rangle , \langle \mathit{Target} \rangle &::= \langle \mathit{AgentId} \rangle \mid \langle \mathit{CoordMediumId} \rangle \mid \langle \mathit{EnvResId} \rangle \mid \bot \\
\langle \mathit{Activity} \rangle &::= \langle \mathit{Operation} \rangle \mid \langle \mathit{Situation} \rangle
\end{aligned}
$$

A Reconciling Abstraction III

- $\langle AgentId \rangle$, $\langle CoordMediumId \rangle$, and $\langle EnvResId \rangle$ represent global identification terms for agents, coordination media, and environment resources, respectively
- $\langle Event \rangle$ is the complete event descriptor
- $\langle Start \rangle$ is the *primary cause* of the event—either an agent or the environment
- $\langle Cause \rangle$ is the *direct cause* of the event—an agent, a medium, the environment
- $\langle Evaluation \rangle$ represents the effect of the event on the MAS
- $\langle Activity \rangle$ is the "stimulus" that actually produced the event
- $\langle Operation \rangle$ represents any coordination operation
- $\langle Situation \rangle$ represents any environmental change, including spatio-temporal ones
- $\langle Time \rangle$ is any expression of time—e.g, absolute, relative
- $\langle Space \rangle$ is the spatial characterisation—e.g., physical, virtual
- $\langle Place \rangle$ is any expression of location in space—e.g., GPS, IP

Outline

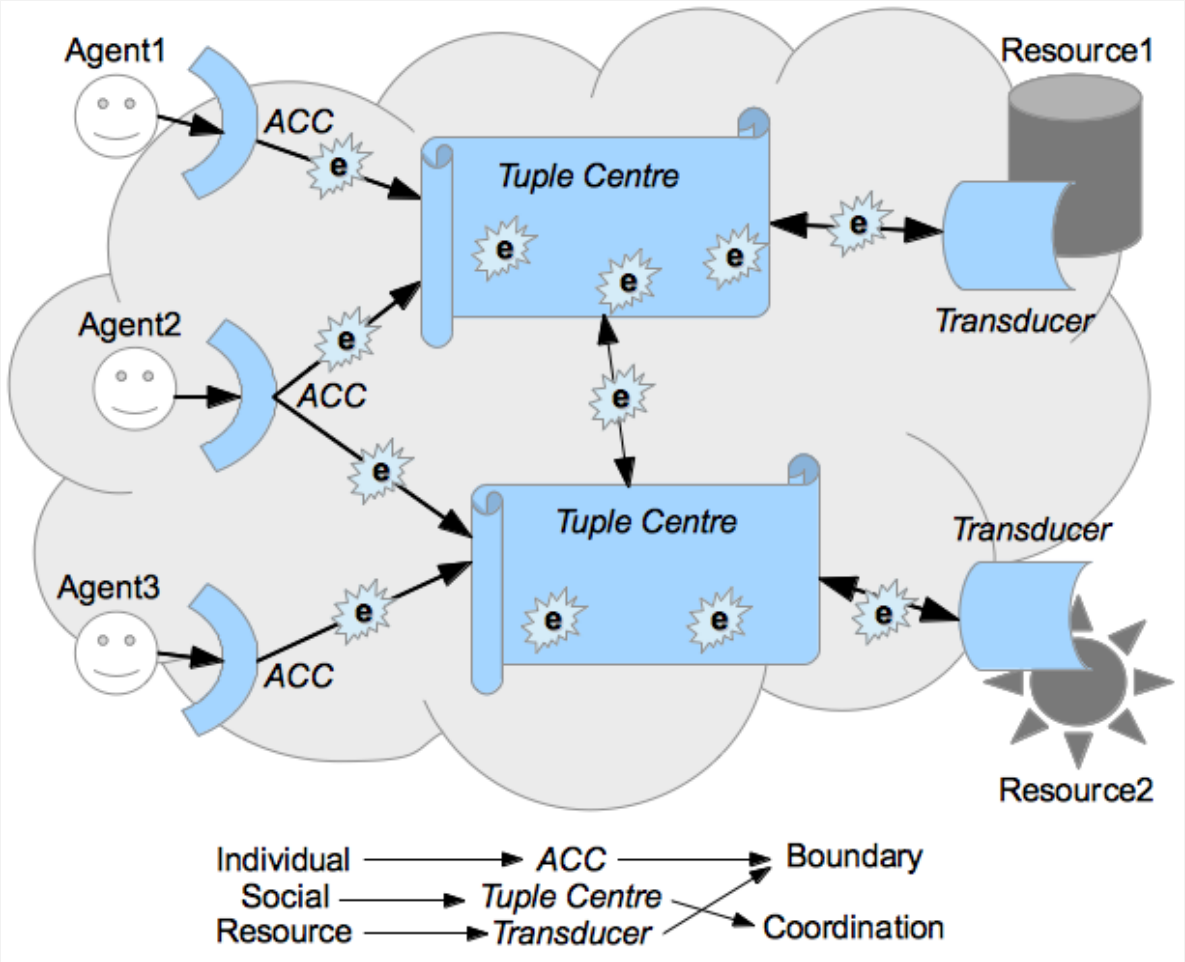
ACC, Tuple Centres & Transducers I

Nodes

- the core abstraction of our coordination architecture is the (coordination) node
- a *node* is the entity responsible for the management of the *interaction space* where all entities – such as agents and other nodes – interact with each other according to a disciplined set of coordination policies
- as a *disembodied* abstraction, a node is identified by using a universal identifier
- as an *embodied* abstraction, a node is (mostly) denoted by its spatio-temporal properties—possibly *mutable* over time, typically depending on the computational devices hosting the node itself

ACC, Tuple Centres & Transducers II

Embodied/disembodied abstractions in a coordination node



ACC, Tuple Centres & Transducers III

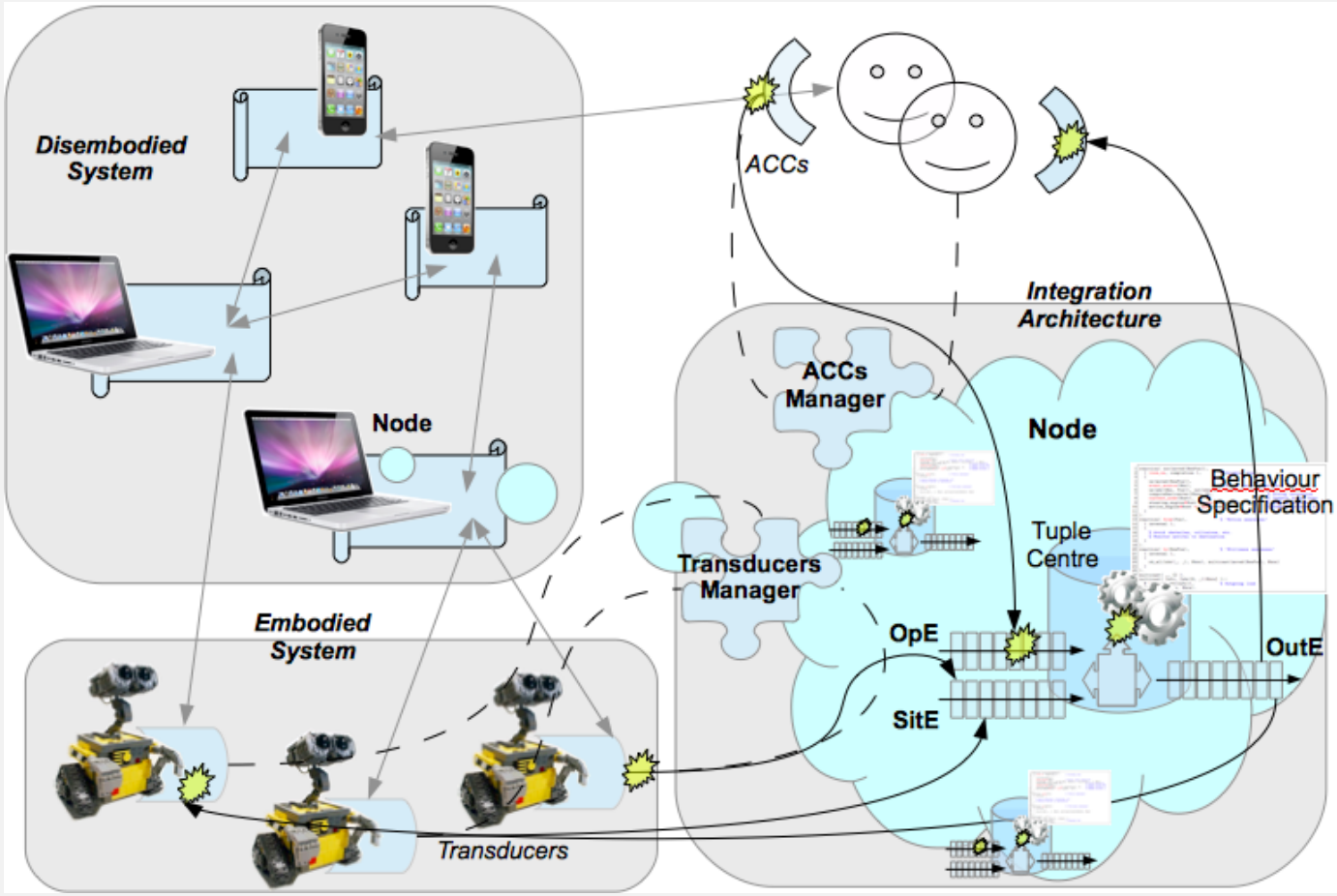
ACC — The Agent Coordination Context (indeed, ACC) is the abstraction that *enable* and *constraint* the space of interaction by providing agents with the coordination operations available according to the agent's role in the MAS [Omicini, 2002]

Tuple Centre — The tuple centre is a programmable coordination medium, ruling and *decoupling* (in *control*, *reference*, *space* and *time*) agents' interactions [Omicini and Denti, 2001]

Transducer — The transducer is the abstraction that *enable* and *constraint* environmental resources interaction capabilities, by translating resource-generated events into the event model suitable to be handled by the coordination medium [Casadei and Omicini, 2009]

ACC, Tuple Centres & Transducers IV

Embodied/disembodied coordination integration architecture



ACC, Tuple Centres & Transducers V

Node — The *node* is responsible for tuple centres lifecycle, ACC negotiation, Transducers (de)registration, events dispatching

ACC Manager — The *ACC manager* takes care of the negotiation process as well as of the mediation process, that is, allowing admissible interactions while forbidding others

Transducer Manager — Similarly, the *transducer manager* is responsible to act as an interface between the resource communication paradigm and working cycle and the tuple centre own

OpE — The Operation Event multiset is the data structure in which incoming operation events – that is, whose source is an agent or a tuple centre through linking – are stored, waiting to be processed

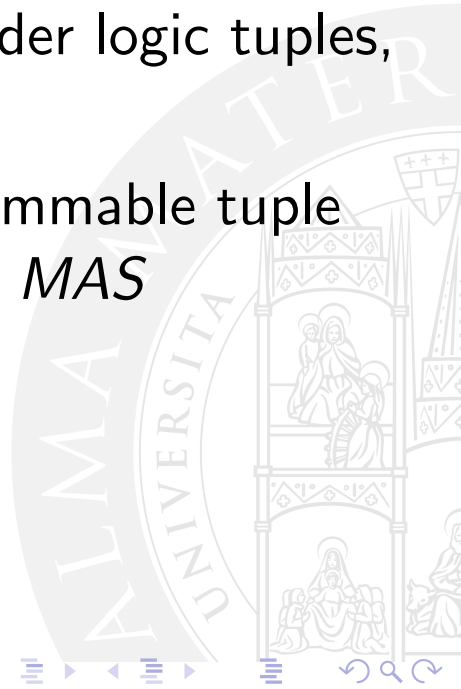
SitE — Similarly, the Situation Event multiset stores situation events – that is, transducers-generated ones or those belonging to the space-time fabric – waiting to be processed

OutE — Finally, the Output Event multiset stores outgoing events of any kind: be them linking operation events or outgoing situation events—e.g., changing a "switched on" property of a motion actuator (resource)

Outline

TuCSoN I

- TuCSoN [TuCSoN, 2013, Omicini and Zambonelli, 1999] is a Java-based middleware supporting tuple-based coordination of autonomous agents in an *open environment*
- featuring tuProlog [tuProlog, 2013, Denti et al., 2001] first-order logic tuples, TuCSoN is well suited for *intelligent agents*
- featuring ReSpecT [Omicini, 2007] tuple centres – programmable tuple spaces –, TuCSoN is well suited for *situated, adaptive MAS*



TuCSoN II

TuCSoN architecture

- TuCSoN *nodes*, hosting the coordination media (tuple centres), are univocally identified by a Java `UUID`[a]
- the ReSpecT *language* [Omicini and Denti, 2001], used to program TuCSoN tuple centres, features the event model here presented, thus supporting recording and inspection of all its properties
- the TuCSoN middleware is built according to the logical architecture discussed here, featuring ACC as well as transducers

[a] `http://docs.oracle.com/javase/7/docs/api/java/util/UUID.html`

TuCSoN on Cloud I

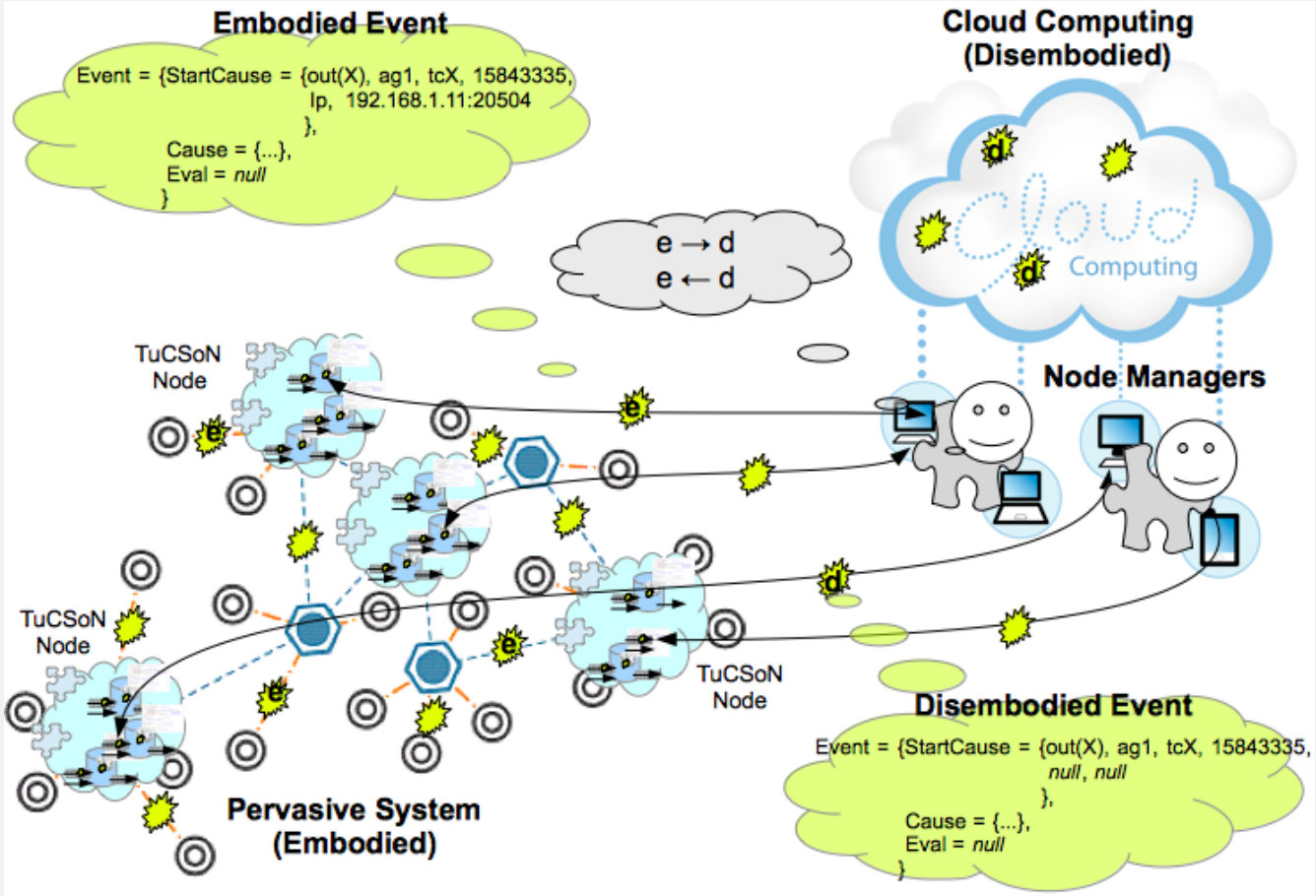
First experiments

- Recently, TuCSoN has been successfully wrapped into a *Cloudify*[a] service, so as to be run as a disembodied coordination service—in particular, a *private cloud*
- Nevertheless, the *on-Cloud* distribution can seamlessly integrate with the *out-Cloud* one, indeed, thanks to the adopted event model and architecture

[a] `http://www.cloudifysource.org`

TuCSoN on Cloud II

TuCSoN-on-Cloud typical deployment scenario



TuCSoN on Cloud III

Seamless Integration

Between the two coordinated subsystems, a new architectural component is added to the TuCSoN middleware: the Node Manager.

- it handles cloud-related issues, such as user accounting, nodes startup/shutdown, relocation, load-balancing and the like
- also, it *translates* events back and forth the embodied / disembodied system (if needed)

In fact, whereas embodied events can be effortlessly handled by *in-Cloud* TuCSoN, disembodied events *can* be filled-in with *situated data* to be effectively handled by *out-Cloud* TuCSoN—e.g., $\langle Space : Place \rangle$ attribute can be either left blank or filled with the DNS name of the whole cloud infrastructure.

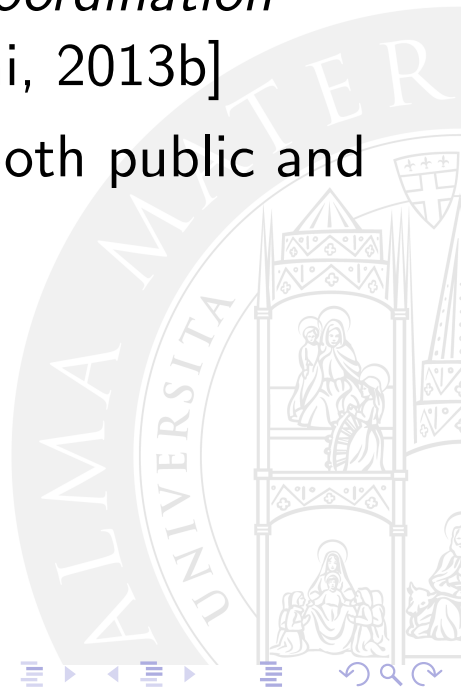
Outline

Done

- The *event model* and the *node* architectural abstraction work well as the core elements for embodied / disembodied coordination architecture for MAS
- TuCSoN middleware with *ACC* and *transducers* provides a suitable framework for testing the architecture
- First experiments of TuCSoN-on-Cloud with Cloudify work nicely

To Do

- Even if successful, the architecture is *too specific* to be generally useful
- A more general architecture based on *boundary* and *coordination artefacts* needs to be devised out [Omicini and Mariani, 2013b]
- Further experiments of TuCSoN-on-Cloud with with both public and private cloud are required



Further References

Paper

Reference [Mariani and Omicini, 2013c]

APICe `http://apice.unibo.it/xwiki/bin/view/Publications/TucsononcloudIca3pp2013`

Presentation

APICe `http://apice.unibo.it/xwiki/bin/view/Talks/TucsononcloudIca3pp2013`

Slideshare `http://www.slideshare.net/andreaomicini/tucson-on-cloud-an-eventdriven-architecture-for-embodied-disembodied-coordination`

References I

Casadei, M. and Omicini, A. (2009).
Situated tuple centres in ReSpecT.
In Shin, S. Y., Ossowski, S., Menezes, R., and Viroli, M., editors, *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, volume III, pages 1361–1368, Honolulu, Hawai'i, USA. ACM.

Ciancarini, P., Omicini, A., and Zambonelli, F. (2000).
Multiagent system engineering: The coordination viewpoint.
In Jennings, N. R. and Lespérance, Y., editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *LNAI*, pages 250–259. Springer.

Denti, E., Omicini, A., and Ricci, A. (2001).
tuProlog: A light-weight Prolog for Internet applications and infrastructures.
In Ramakrishnan, I., editor, *Practical Aspects of Declarative Languages*, volume 1990 of *LNCS*, pages 184–198. Springer.
3rd International Symposium (PADL 2001), Las Vegas, NV, USA, 11–12 March 2001. Proceedings.

Fortino, G., Garro, A., Mascillaro, S., and Russo, W. (2010).
Using event-driven lightweight DSC-based agents for MAS modelling.
International Journal of Agent-Oriented Software Engineering, 4(2):113–140.

References II

Goldin, D. and Wegner, P. (2008).
The interactive nature of computing: Refuting the strong church-turing thesis.
Minds and Machines, 18(1):17–38.

Hill, R., Hirsch, L., Lake, P., and Moshiri, S. (2013).
Guide to Cloud Computing. Principles and Practice.
Computer Communications and Networks. Springer London.

Mariani, S. and Omicini, A. (2013a).
Event-driven programming for situated MAS with ReSpecT tuple centres.
In Klusch, M., Thimm, M., and Paprzycki, M., editors, *Multiagent System Technologies*, volume 8076 of *LNAI*, pages 306–319. Springer.
11th German Conference (MATES 2013), Koblenz, Germany, 16-20 September 2013. Proceedings.

Mariani, S. and Omicini, A. (2013b).
Promoting space-aware coordination: ReSpecT as a spatial-computing virtual machine.
In *Spatial Computing Workshop (SCW 2013)*, AAMAS 2013, Saint Paul, Minnesota, USA.

References III

Mariani, S. and Omicini, A. (2013c).
TuCSoN on cloud: An event-driven architecture for embodied / disembodied coordination.
In Aversa, R., Kolodzej, J., Zhang, J., Amato, F., and Fortino, G., editors, *Algorithms and Architectures for Parallel Processing*, volume 8286 of *LNCS*, pages 285–294. Springer International Publishing Switzerland.
13th International Conference (ICA3PP-2013), Vietri sul Mare, Italy, 18-20 December 2013. Proceedings, Part II.

Omicini, A. (2002).
Towards a notion of agent coordination context.
In Marinescu, D. C. and Lee, C., editors, *Process Coordination and Ubiquitous Computing*, chapter 12, pages 187–200. CRC Press, Boca Raton, FL, USA.

Omicini, A. (2007).
Formal ReSpecT in the A&A perspective.
Electronic Notes in Theoretical Computer Science, 175(2):97–117.

References IV

Omicini, A. (2013).
Nature-inspired coordination for complex distributed systems.
In Fortino, G., Bădică, C., Malgeri, M., and Unland, R., editors, *Intelligent Distributed Computing VI*, volume 446 of *Studies in Computational Intelligence*, pages 1–6. Springer.
6th International Symposium on Intelligent Distributed Computing (IDC 2012), Calabria, Italy, 24-26 September 2012. Proceedings. Invited paper.

Omicini, A. and Denti, E. (2001).
From tuple spaces to tuple centres.
Science of Computer Programming, 41(3):277–294.

Omicini, A. and Mariani, S. (2013a).
Agents & multiagent systems: En route towards complex intelligent systems.
Intelligenza Artificiale, 7(2):153—164.
Special Issue Celebrating 25 years of the Italian Association for Artificial Intelligence.

References V


Omicini, A. and Mariani, S. (2013b).
Coordination for situated MAS: Towards an event-driven architecture.
In Moldt, D. and Rölke, H., editors, *International Workshop on Petri Nets and Software Engineering (PNSE'13)*, volume 989 of *CEUR Workshop Proceedings*, pages 17–22. Sun SITE Central Europe, RWTH Aachen University.
Joint Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'13) and the International Workshop on Modeling and Business Environments (ModBE'13), co-located with the 34th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets 2013). Milano, Italy, 24–25 June 2013. Invited paper.


Omicini, A., Ricci, A., and Viroli, M. (2005).
Time-aware coordination in ReSpecT.
In Jacquet, J.-M. and Picco, G. P., editors, *Coordination Models and Languages*, volume 3454 of *LNCS*, pages 268–282. Springer-Verlag.


Omicini, A. and Zambonelli, F. (1999).
Coordination for Internet application development.
Autonomous Agents and Multi-Agent Systems, 2(3):251–269.

References VI

TuCSoN (2013).
Home page.
http://tucson.unibo.it.

tuProlog (2013).
Home page.
http://tuprolog.unibo.it.

Zambonelli, F., Castelli, G., Ferrari, L., Mamei, M., Rosi, A., Di Marzo Serugendo, G., Risoldi, M., Tchao, A.-E., Dobson, S., Stevenson, G., Ye, Y., Nardini, E., Omicini, A., Montagna, S., Viroli, M., Ferscha, A., Maschek, S., and Wally, B. (2011).
Self-aware pervasive service ecosystems.
Procedia Computer Science, 7:197–199.

TuCSoN on Cloud: An Event-driven Architecture for Embodied / Disembodied Coordination

Stefano Mariani Andrea Omicini

{s.mariani, andrea.omicini}@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum—Università di Bologna

C-SmartCPS @ ICA3PP 2013
Vietri sul Mare, SA, Italy – 20 December 2013