

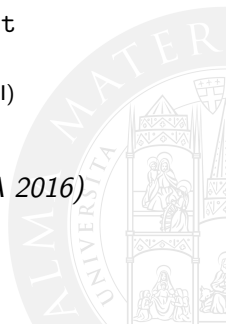
Multi-paradigm Coordination for MAS

Integrating Heterogeneous Coordination Approaches in MAS Technologies

Stefano Mariani **Andrea Omicini**
{s.mariani, andrea.omicini}@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna

XVII Workshop “From Objects to Agents” (WOA 2016)
Catania, Italy, 30 July 2016



- 1 Motivation & Goal
- 2 State-of-Art Agent-oriented Frameworks
- 3 Enabling Multi-paradigm Coordination
- 4 Conclusive Remarks



Outline

- 1 Motivation & Goal
- 2 State-of-Art Agent-oriented Frameworks
- 3 Enabling Multi-paradigm Coordination
- 4 Conclusive Remarks



Motivation & Goal I

- multi-agent systems (MAS) impose **heterogeneous** requirements on the **interaction means and paradigms**
 - e.g., message-passing vs. shared space, synchronous vs. asynchronous, proactive vs. reactive, etc.
 - e.g., software agents vs. physical devices
- **coordination** approaches can be classified according to diverse criteria
 - *means* of interaction—e.g. messages vs. tuples
 - *synchronism* of interaction primitives—synchronous vs. asynchronous
 - *programming style*—e.g. reactive (actors), vs. pro-active (agents)
 - *objective* or *subjective* [OO03], depending on *who* is responsible for the “burden of coordination”—agents themselves or a third-party

Motivation & Goal II

- respecting agents' **autonomy** is a foremost issue in design of coordination models for MAS
 - ! *reactive* programming styles often lead to **inversion of control**
 - ! *synchronous* semantics *without* adequate *uncoupling* leads agents to **passively** withstand the outcomes of coordination

Goal

provide MAS designers with seamless support to **multi-paradigm coordination** while **preserving autonomy** of agents

- ✓ agents *deliberatively* adapt coordination paradigm and interaction means based on static preference or *run-time* contingencies
- ✓ such a choice *does not affect negatively* other agents' activities
- ⇒ focus on JADE [BPR99], *Jason* [BHW07], and TuCSoN [OZ99]

Outline

- 1 Motivation & Goal
- 2 State-of-Art Agent-oriented Frameworks**
- 3 Enabling Multi-paradigm Coordination
- 4 Conclusive Remarks



JADE in a Nutshell I

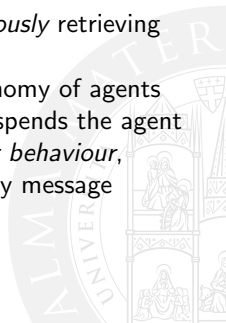
JADE

JADE [BPR99] is a Java-based framework and infrastructure to develop and deploy *agent-based distributed* applications in compliance with FIPA standard specifications for *interoperable* MAS

- *autonomy* of agents is supported by the **behaviour** abstraction
 - ! Java objects executed *pseudo-concurrently* within a **single thread** by a hidden *non-preemptive round-robin scheduler*
 - ! behaviours *switch only at completion*—in the meanwhile *no other behaviour* can execute
 - ! behaviours execute *from the beginning* every time—no way to “stop-then-resume” a behaviour at arbitrary statements

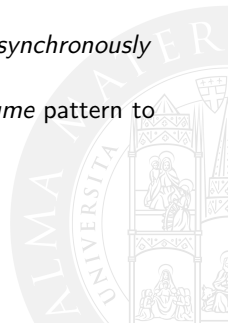
JADE in a Nutshell II

- *coordination* adopts the subjective stance through the **agent communication channel** (ACC), providing *asynchronous message-passing*
 - agents have a mailbox where incoming communications wait to be *pro-actively* considered
 - `receive()` vs. `blockingReceive()` for *(a)synchronously* retrieving messages
 - ! “**block-then-resume**” **pattern** to avoid hindering autonomy of agents with `blockingReceive()`—`blockingReceive()` suspends the agent *as a whole*, method `block()` suspends *only the caller behaviour*, automatically resumed by JADE upon reception of any message



JADE in a Nutshell III

- ... on top of which **FIPA protocols** (e.g. Contract Net [Smi80] and Achieve Rational Effect [Fou02]) are designed and provided as *callback frameworks*
 - callback methods correspond to each interaction step (message to be received or sent)
 - promote a **reactive programming** style, where agents *synchronously wait* for messages
 - ✓ internally implemented abiding to the *block-then-resume* pattern to keep responsiveness to other interactions

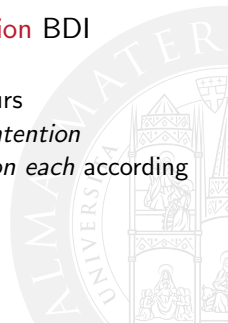


Jason in a Nutshell I

Jason

Jason [BHW07] is a development framework and runtime system for *BDI* agents implemented in a dialect of *AgentSpeak* [Rao96]

- *autonomy* of agents is supported by the **plan** / **intention** BDI abstractions
 - a plan is scheduled as soon as a *triggering event* occurs
 - not directly executed “as is” but instantiated as an *intention*
 - ! intentions are pseudo-concurrently executed *one action each* according to a **single-threaded** round-robin scheduler



Jason in a Nutshell II

- *coordination* adopts the subjective stance through an **asynchronous message passing** layer
 - *no explicit receive* primitive involved, nor a mailbox to monitor: depending on *illocutionary force* of messages, reception causes different *events triggering* execution of plans
 - *tell* primitive to add a belief to the belief base of receiver agent, causing a *belief addition event* handled by plans starting with *+b* clause head—thus messages can be **either re-actively or pro-actively** considered
 - ✓ *asynchronous* vs. *synchronous* *askOne* primitive to query another agent's belief base—**caller intention only** (not the whole agent) is suspended, then automatically resumed when possible

TuCSoN in a Nutshell I

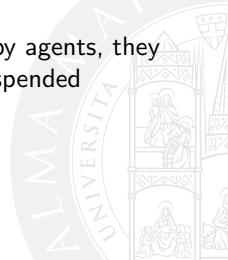
TuCSoN

TuCSoN [OZ99] is a Java-based, (logic) tuple-based coordination model and infrastructure for open, distributed MAS, providing *objective coordination as service* through ReSpecT *tuple centres* [OD01]

- TuCSoN *preserves autonomy* of agents through **agent coordination contexts** (ACC) [Omi02] ...
 - ✓ assigned to agents as they enter a TuCSoN-coordinated MAS, for **uncoupling synchronism** of coordination operations invocation from *suspensive semantics* of coordination primitives
 - mapping operations to events **asynchronously** dispatched to ReSpecT tuple centres

TuCSoN in a Nutshell II

- ...and “two-phases” **execution** of coordination operations
 - *invocation vs. completion* phase: in the former, operation *request* is dispatched to the target tuple centre; in the latter, its *response* is sent back to the invoker
 - ! **synchronism** of invocation (*un*)*couples* the fate of the coordination operation w.r.t. that of the invoker agent—e.g. a suspensive LINDA in [Gel85]
 - ! if (a)synchronous invocation is deliberately chosen by agents, they (*do not*) *get suspended, too*, if the operation gets suspended



Outline

- 1 Motivation & Goal
- 2 State-of-Art Agent-oriented Frameworks
- 3 Enabling Multi-paradigm Coordination**
- 4 Conclusive Remarks



TuCSon4JADE in a Nutshell

- ✓ *autonomy-preserving* integration presented in [MOS14]
- ! main issue is making TuCSon *synchronous invocation* mode compatible with JADE concurrency model based on *behaviours*
- ✗ **direct API access** hinders agents autonomy: control flow is *coupled* to that of coordination operations, which is harmful in case of *synchronous* invocation [ORV⁺04]
- ⇒ **TuCSon4JADE**: bridge component suitably *suspending* / *resuming* *caller behaviours* depending on outcome of coordination operations

<http://apice.unibo.it/xwiki/bin/view/TuCSon/4JADE>

TuCSoN4Jason in a Nutshell

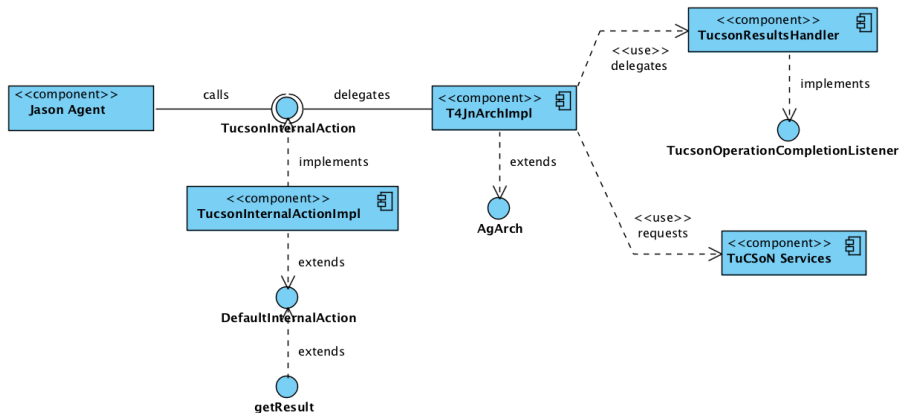
- ! main integration issue is making TuCSoN *synchronous invocation* semantics compatible with Jason concurrency model based on *intentions*
- ✗ **direct API access** – through Jason internal actions – hinders agents autonomy as in the case of JADE
- ⇒ **TuCSoN4Jason**: fine-grained integration of TuCSoN two-phases execution with Jason intention suspension mechanism, properly *suspending / resuming caller intentions* depending on outcome of coordination operations

<http://apice.unibo.it/xwiki/bin/view/TuCSoN/4Jason>

TuCSoN4Jason: Architecture I

- ❶ **custom agent architecture** enabling *customisation* of the BDI engine hidden behind each *Jason* agent
 - dispatches operation requests and responses
 - tracks pending and completed operations
- ❷ **operation completion listener** enabling TuCSoN to *couple* its invocation semantics with *Jason* intention suspension mechanism
 - dispatches notifications of operation completion
 - ✓ (automatically) **resumes suspended** intentions when their result becomes available
- ❸ **custom internal actions** enabling *Jason* agents to request TuCSoN coordination services and inspect operations outcome
 - provides *Jason* agents coordination primitives
 - ✓ primitive `getResult` handles **(automatic) intention suspension** when result of a getter primitive (such as `in`) is not yet available

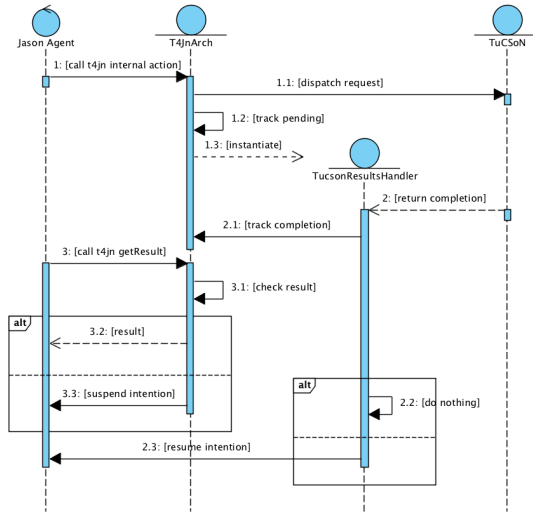
TuCSon4Jason: Architecture II



TuCSoN4Jason: Flow of Interactions I

- 1 upon internal action call the custom agent architecture (**T4JnArch**) *asynchronously* dispatches operation request to TuCSoN
- 2 then it *tracks pending state* of operation and spawns a *dedicated completion handler* (**TucsonResultsHandler**)
- 3 upon operation completion the handler is notified by TuCSoN with the result
- 4 state of the operation is changed from “pending” to “completed”
- 5 if needed, the *suspended intention* within which **getResult()** was called is *resumed*

TuCSoN4Jason: Flow of Interactions II



TuCSoN4Jason: Implementation I

Behind the scenes

- ! any TuCSoN4Jason internal action has an **asynchronous invocation semantics**, so that agents are free to choose when to risk suspension (of the **caller intention only**) by calling `getResult()`
- ⇒ when doing so, TuCSoN4Jason checks if operation result is available
 - if it is, the current intention can proceed
 - otherwise the intention gets suspended



TuCSoN4Jason: Implementation II

```
...  
// result NOT available  
if (!(results.containsKey(actionId))) {  
    Circumstance c = ts.getC();  
    // suspend calling intention  
    Intention i = c.getSelectedIntention();  
    this.suspendIntention = true;  
    i.setSuspended(true);  
    c.addPendingIntention(  
        jason.stdlib.suspend.SELF_SUSPENDED_INT  
        + i.getId(), i);  
    // track suspended intention  
    arch.getSuspendedIntentions().put(actionId, i);  
    mutex.unlock(); // thread-safety  
    return true;  
}  
...
```

Figure: Intention suspension when getResult() is called with no result available

TuCSoN4Jason: Implementation III

```
// corresponding action caused suspension
if (this.suspendedIntentions.containsKey(this.actionId)) {
    Intention suspendedIntention = this.suspendedIntentions
        .remove(this.actionId);
    this.mutex.unlock(); // thread-safety
    ... // parse result
    Circumstance c = this.ts.getC();
    Iterator<String> ik = c.getPendingIntentions().keySet().iterator();
    while (ik.hasNext()) { // scan pending intentions
        String k = ik.next();
        if (k.startsWith(suspend.SUSPENDED_INT)) {
            Intention i = c.getPendingIntentions().get(k);
            if (i.equals(suspendedIntention)) { // find intention
                i.setSuspended(false);
                ik.remove();
                ... // other reasoner-related stuff
                c.resumeIntention(i);
            }
        }
    }
}
```

Figure: Resuming a suspended intention when its result becomes available

TuCSoN4Jason: Wrap Up

TuCSoN4Jason lets *Jason* agents **simultaneously** adopt **different coordination paradigms and means**, depending on run-time needs, *without them hindering* each other nor any other ongoing interaction activities

- ✓ **subjective** coordination $\Rightarrow$ *Jason asynchronous message* passing—handled, e.g., reactively through belief addition events (+b)
- ✓ **objective** coordination $\Rightarrow$ TuCSoN4Jason facilities
 - ✓ **synchronous** TuCSoN operations $\Rightarrow$ follow an operation invocation with immediate call to `getResult()`
 - ✓ **asynchronous** TuCSoN operations
 - ✓ **pro-active** programming style $\Rightarrow$ call `getResult()` in the *Jason* plan when the information is needed
 - ✓ **reactive** programming style $\Rightarrow$ call `getResult()` in a *parallel plan* dedicated to handling operation result

Multi-paradigm ContractNet: Scenario

- **contract net protocol** (CNP) [Smi80] re-interpreted in a *multi-paradigm coordination* setting
 - ! a *single* call-for-proposals (CFP) tuple in a shared contract-net space is more efficient than messaging each seller $\Rightarrow$ **tuple-based CFP**
 - ! purchase is typically a 1-to-1 interaction $\Rightarrow$ **message-based purchase**
 - Book trading scenario
 - n seller agents advertise their catalogue of books
 - m buyer agents browse such catalogues looking for books
 - buyers start a CFP
 - sellers reply with actual proposals
 - buyers choose which one to accept
- ! **concurrency property** $\Rightarrow$ sellers should *stay reactive* to CFPs in the middle of a purchase transaction—besides multiple CFPs or purchases

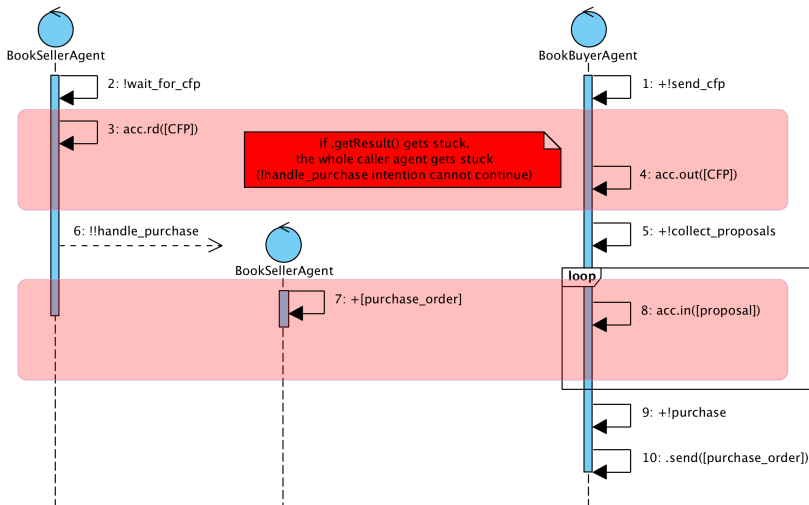
Paradigmatic example of practical relevance of **preserving autonomy** while enabling multi-paradigm coordination

Multi-paradigm ContractNet: Naive Approach I

- no *TuCSon4Jason* $\implies$ just call TuCSon API
 - ! while the seller replies to CFPs, or carries on purchase orders, waiting for further CFPs causes *suspension of the intention*—due getter primitives *suspensive semantics*
- this is fine: it is exactly for this suspensive semantics that the LINDA model works—thus, TuCSon
- ✗ what is not-so-fine is that being the getter operation stuck, the caller agent *as a whole* is stuck too, and **cannot** schedule other intentions in the meanwhile

Concurrency property is lost $\implies$ **autonomy of agents hindered**

Multi-paradigm ContractNet: Naive Approach II

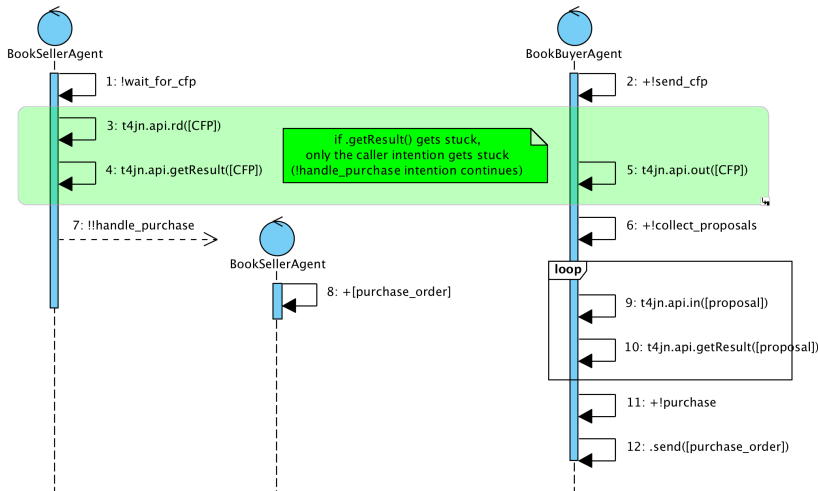


Multi-paradigm ContractNet: TuCSoN4Jason Approach I

- **TuCSoN4Jason** $\Rightarrow$ call TuCSoN4Jason API
 - ! while the seller replies to CFPs, or carries on purchase orders, ... same as before :)
 - again, this is fine: as before :)
 - ✓ what is now fixed is that *suspensive semantics* is **confined** to the caller intention, by the bridge
- $\Rightarrow$ *only the caller intention* is suspended, whereas other activities can carry on in the meanwhile

Concurrency property holds $\Rightarrow$ **autonomy of agents preserved**, too

Multi-paradigm ContractNet: TuCSoN4Jason Approach II



Outline

- 1 Motivation & Goal
- 2 State-of-Art Agent-oriented Frameworks
- 3 Enabling Multi-paradigm Coordination
- 4 Conclusive Remarks**



Which Impact?

Enabling *multi-paradigm coordination* along a number of orthogonal dimensions of coordination

- objective vs. subjective stance
- synchronous vs. asynchronous invocation
- pro-active vs. reactive programming style

brings about many requirements, especially whenever coordinables are allowed to *change* coordination paradigm and means *at run-time*

Technologies like TuCSoN4Jason and TuCSoN4JADE are *key-enablers* for applications featuring *distributed* and *heterogeneous* components which

- depend on communication and coordination to carry out their duties
- require diverse coordination means depending on both their own capabilities and run-time contingencies

References I



Rafael H. Bordini, Jomi F. Hübner, and Michael J. Wooldridge.
Programming Multi-Agent Systems in AgentSpeak using Jason.
John Wiley & Sons, Ltd, October 2007.



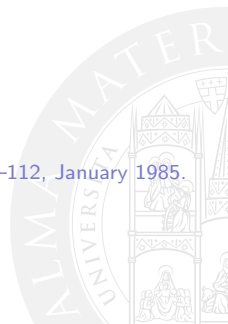
Fabio Luigi Bellifemine, Agostino Poggi, and Giovanni Rimassa.
JADE—a FIPA-compliant agent framework.
In *4th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-99)*, pages 97–108, London, UK, 19–21 April 1999. The Practical Application Company Ltd.



Foundation for Intelligent Physical Agents.
FIPA Communicative Act Library Specification.
<http://www.fipa.org/specs/fipa00037/>, 6 December 2002.



David Gelernter.
Generative communication in Linda.
ACM Transactions on Programming Languages and Systems, 7(1):80–112, January 1985.



References II



Stefano Mariani, Andrea Omicini, and Luca Sangiorgi.

Models of autonomy and coordination: Integrating subjective & objective approaches in agent development frameworks.

In Lars Braubach, David Camacho, and Salvatore Ventcinque, editors, *Intelligent Distributed Computing VIII*, volume 570 of *Studies in Computational Intelligence*, pages 69–79. Springer, 2014.

8th International Symposium on Intelligent Distributed Computing (IDC 2014), Madrid, Spain, 3-5 September 2014. Proceedings.



Andrea Omicini and Enrico Denti.

From tuple spaces to tuple centres.

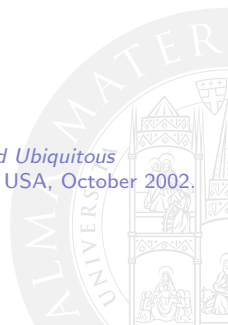
Science of Computer Programming, 41(3):277–294, November 2001.



Andrea Omicini.

Towards a notion of agent coordination context.

In Dan C. Marinescu and Craig Lee, editors, *Process Coordination and Ubiquitous Computing*, chapter 12, pages 187–200. CRC Press, Boca Raton, FL, USA, October 2002.



References III



Andrea Omicini and Sascha Ossowski.

Objective versus subjective coordination in the engineering of agent systems.

In Matthias Klusch, Sonia Bergamaschi, Peter Edwards, and Paolo Petta, editors, *Intelligent Information Agents: An AgentLink Perspective*, volume 2586 of *Lecture Notes in Computer Science*, pages 179–202. Springer Berlin Heidelberg, 2003.



Andrea Omicini, Alessandro Ricci, Mirko Viroli, Marco Cioffi, and Giovanni Rimassa.

Multi-agent infrastructures for objective and subjective coordination.

Applied Artificial Intelligence, 18(9-10):815–831, 2004.



Andrea Omicini and Franco Zambonelli.

Coordination for Internet application development.

Autonomous Agents and Multi-Agent Systems, 2(3):251–269, September 1999.

Special Issue: Coordination Mechanisms for Web Agents.



Anand S. Rao.

AgentSpeak(L): BDI agents speak out in a logical computable language.

In Walter Van de Velde and John W. Perram, editors, *Agents Breaking Away*, volume 1038 of *Lecture Notes in Computer Science*, pages 42–55. Springer, 1996.

7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), Eindhoven, The Netherlands, 22-25 January 1996, Proceedings.

References IV



Reid G. Smith.

The Contract Net Protocol: High-level communication and control in a distributed problem solver.

IEEE Transactions on Computers, C-29(12):1104–1113, 1980.



URLs

Slides

- on APICe

→ <http://apice.unibo.it/xwiki/bin/view/Talks/MulticoordWoa2016>

- on SlideShare

→ <http://www.slideshare.net/andreaomicini/multiparadigm-coordination-for-mas>

Paper

- on APICe

→ <http://apice.unibo.it/xwiki/bin/view/Publications/MulticoordWoa2016>

- on CEUR

→ http://ceur-ws.org/Vol-????/paper_?.pdf

Multi-paradigm Coordination for MAS

Integrating Heterogeneous Coordination Approaches in MAS Technologies

Stefano Mariani **Andrea Omicini**
{s.mariani, andrea.omicini}@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna

XVII Workshop “From Objects to Agents” (WOA 2016)
Catania, Italy, 30 July 2016

