Introduction to TuCSoN

Andrea Omicini
after Alessandro Ricci

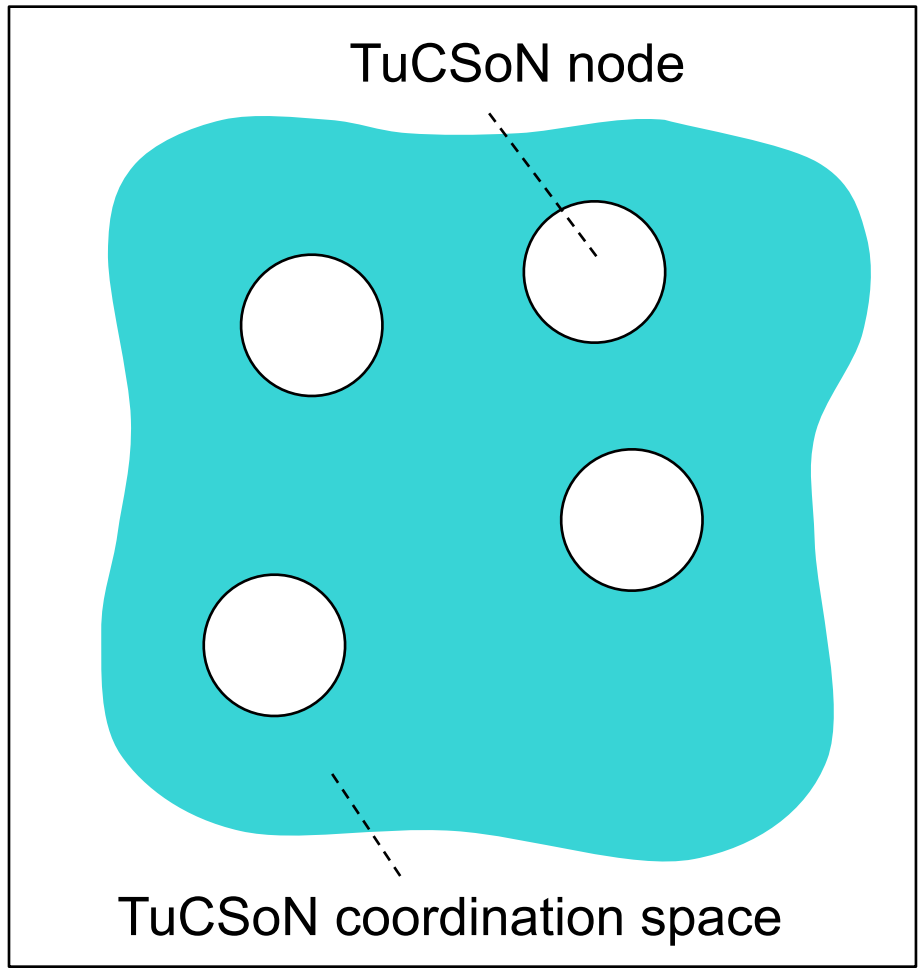
TuCSoN
COORDINATION INFRASTRUCTURE

- Tuple Centre Spread over the Network (Omicini & Zambonelli, 1999)

- Coordination infrastructure providing Tuple Centres as coordination services
  - tuple centres are distributed over the network, collected in *nodes*
  - agents interact and coordinate through tuple centres located in nodes

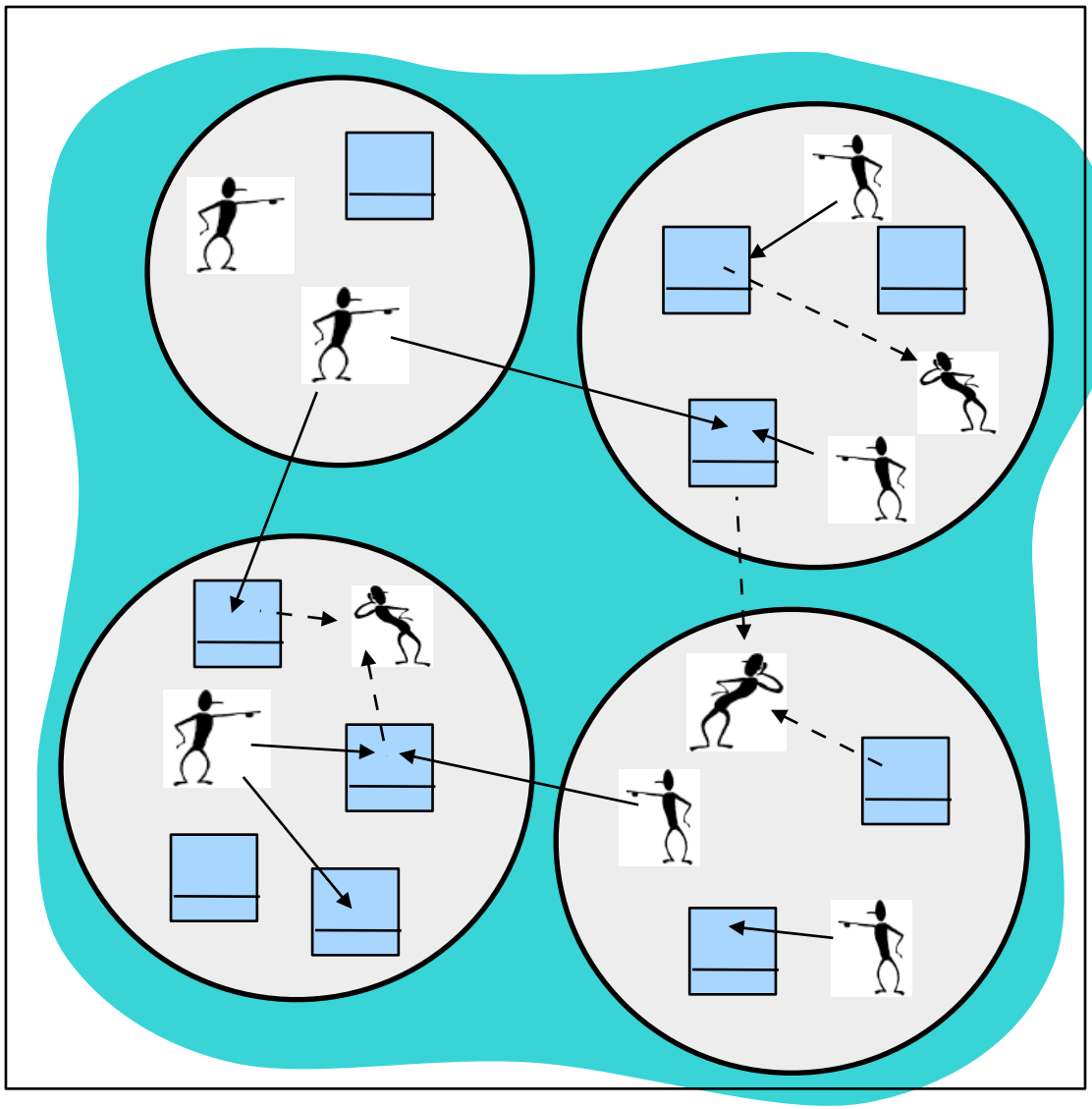
TuCSoN COORDINATION SPACE

- *Coordination space* = set of distributed *nodes*
  - Each TuCSoN node is an Internet node
    - Identified by the IP (logic) address
- Each coordination space belongs to some *organisation*



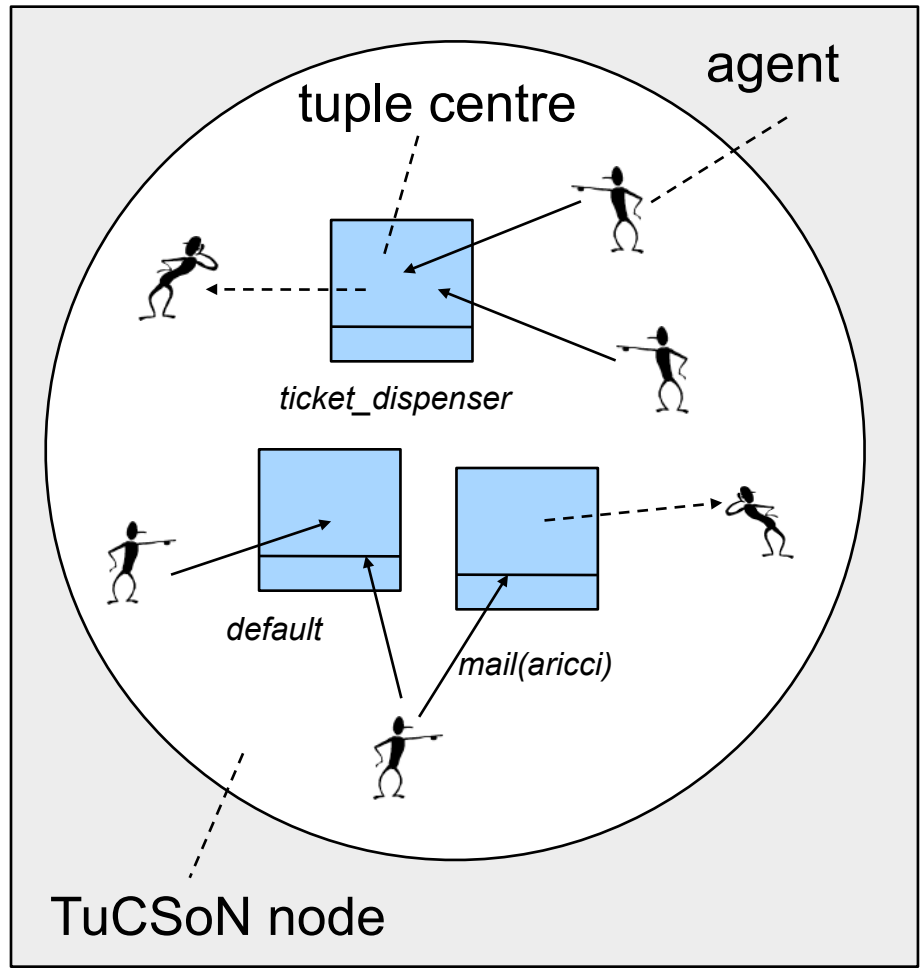
TuCSoN COORDINATION SPACE: DETAIL

- Each node can host an unlimited number of *labelled* tuple centre
  - identified by means of a unique name
    - *Ex: ticket_dispenser*
- Agents can access / use either *local* tuple centres or tuple centres hosted in other nodes
  - full tuple centre identifier: <local_name>@<node>



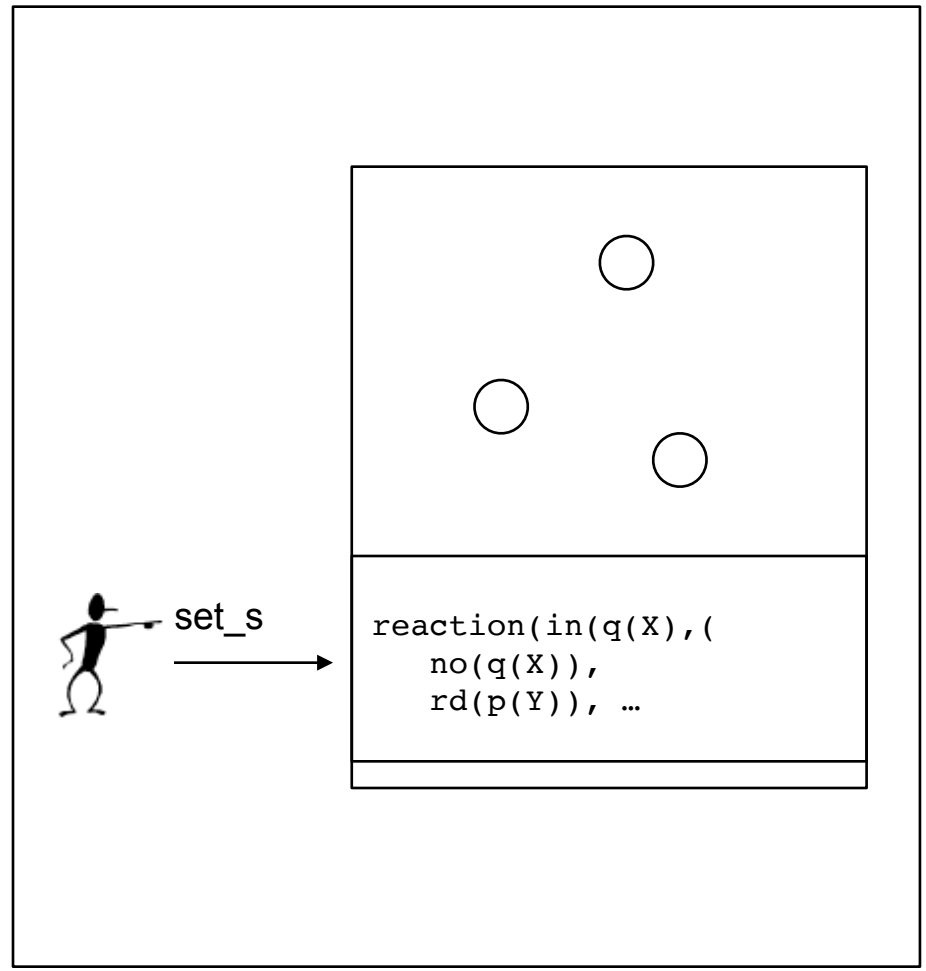
TuCSoN NODE

- Each node has a *default* tuple centre
  - referred implicitely when agents do not specify tuple centre name
- No explicit tuple centre creation needed
  - a tuple centre is automatically created by the local infrastructure when referred to within an operation
- For agents accessing local tuple centres, no need to specify node address
  - local name is enough



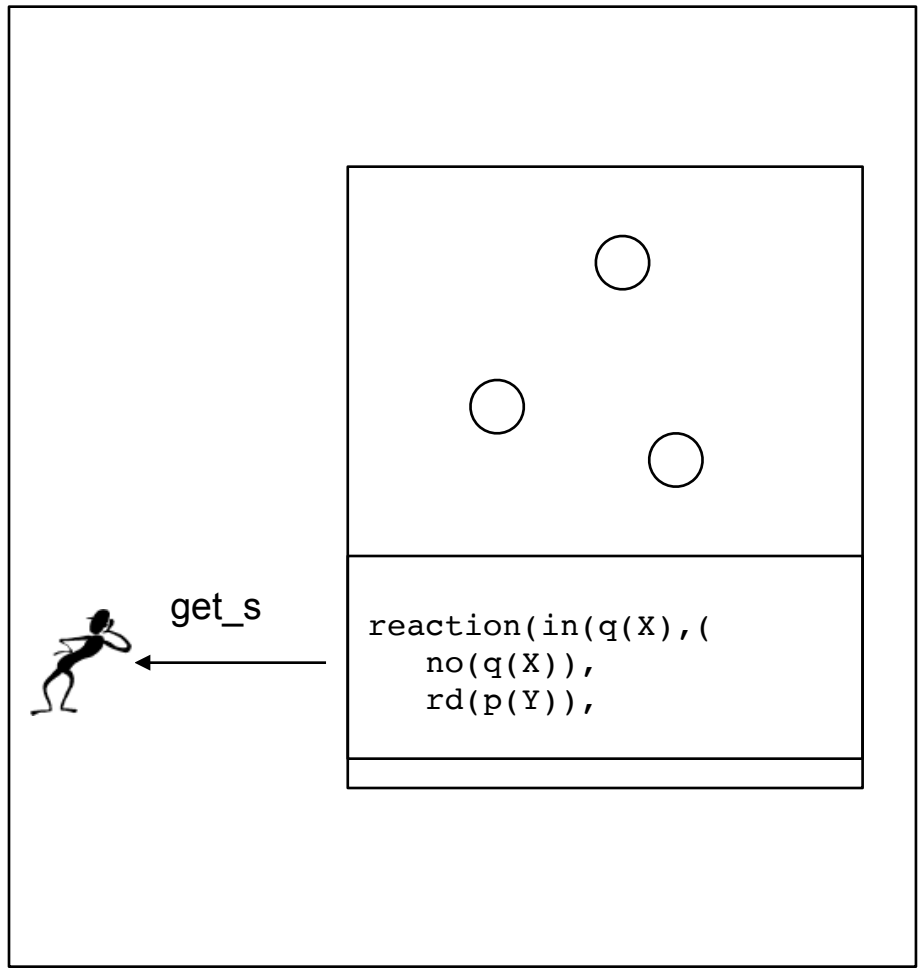
DYNAMIC PROGRAMMABILITY

- Tuple centre behaviour can be set/changed at runtime
  - set_s operation
- Good for open systems
  - changing / adapting / tuning coordination policies at runtime



DYNAMIC INSPECTABILITY

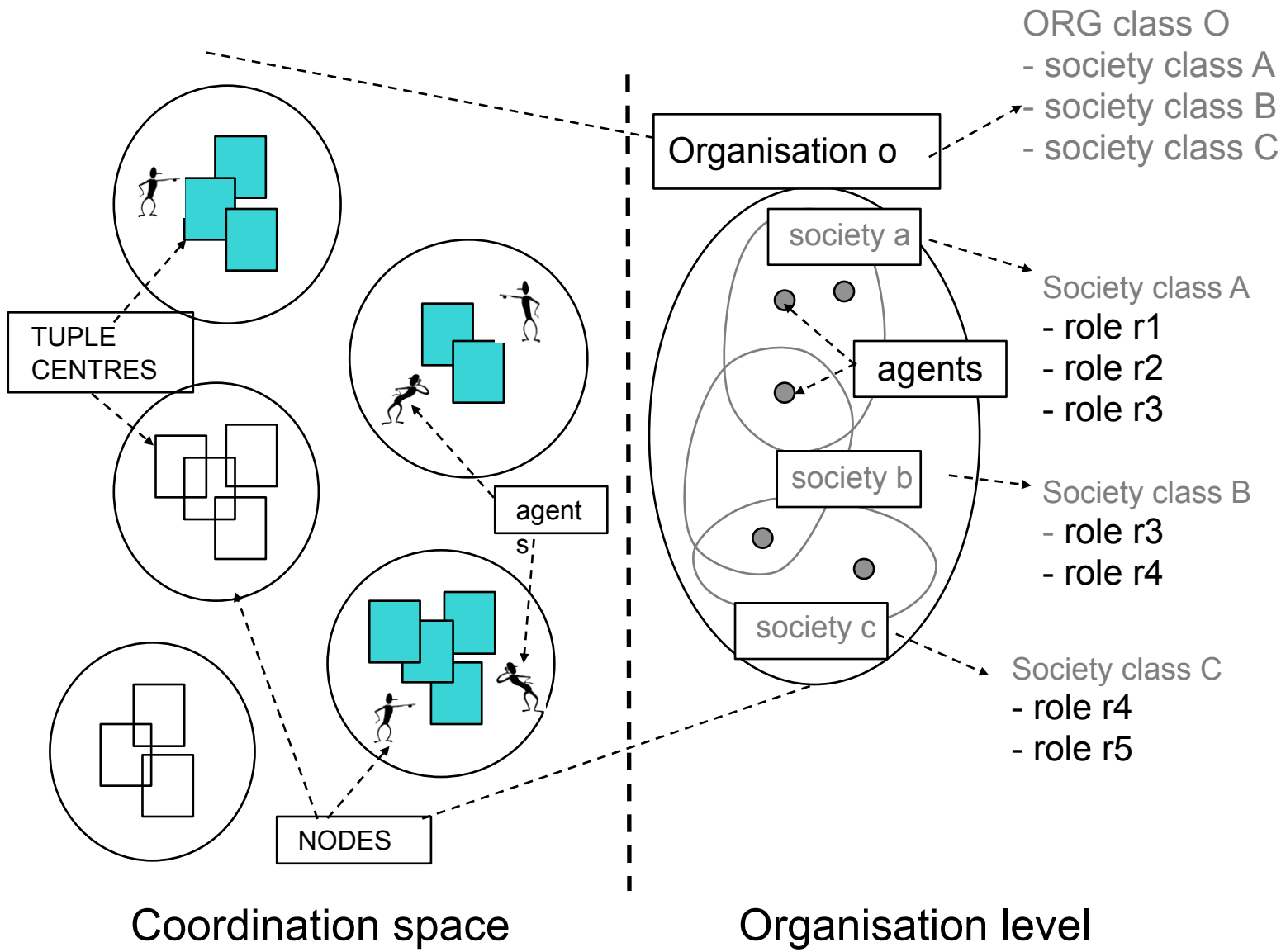
- Tuple centre behaviour can be inspected dynamically, at runtime
  - get_s operation



TuCSoN MOST RECENT EXTENSIONS

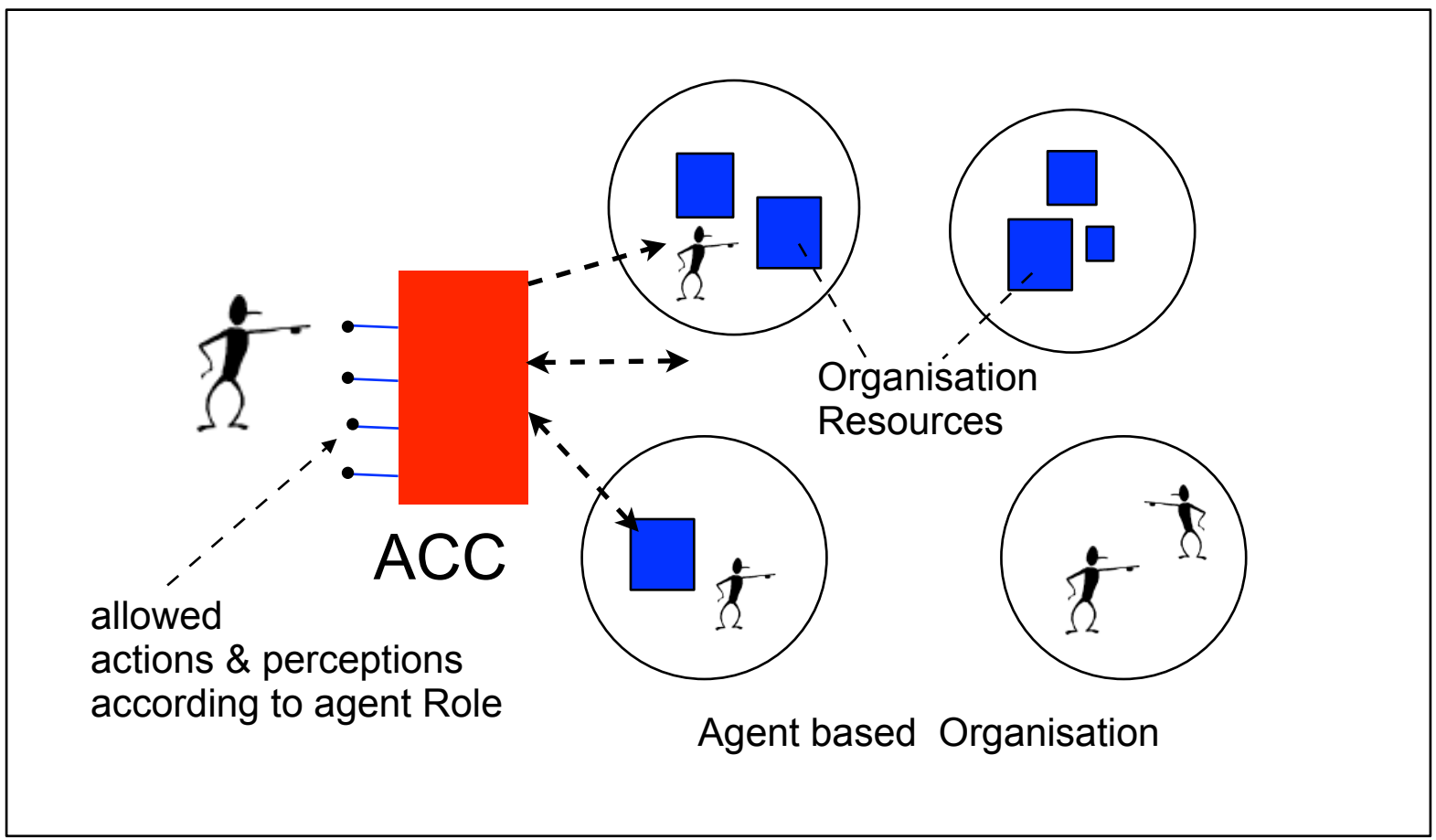
- Main extensions toward supporting *organisation* and *security* aspects
  - an integrated approach
- Main elements
  - Role-Based Access Control (RBAC) architecture
  - Agent Coordination Context (ACC) abstraction
- Scenario
  - in order to act within an organisation (coordination space), an agent *must* first obtain an ACC, as interface to act upon organisation resources (such as tuple centres)
  - an ACC is dynamically created and released by the infrastructure
    - configured to enable and constrain agent action space according to the *role(s)* played by the agent

TuCSoN ORGANISATION MODEL



ACC ABSTRACTION

- Encapsulating *role policies*, defining & constraining agent action space
- Modelling a *working session* inside an organisation



TuCSoN TECHNOLOGY

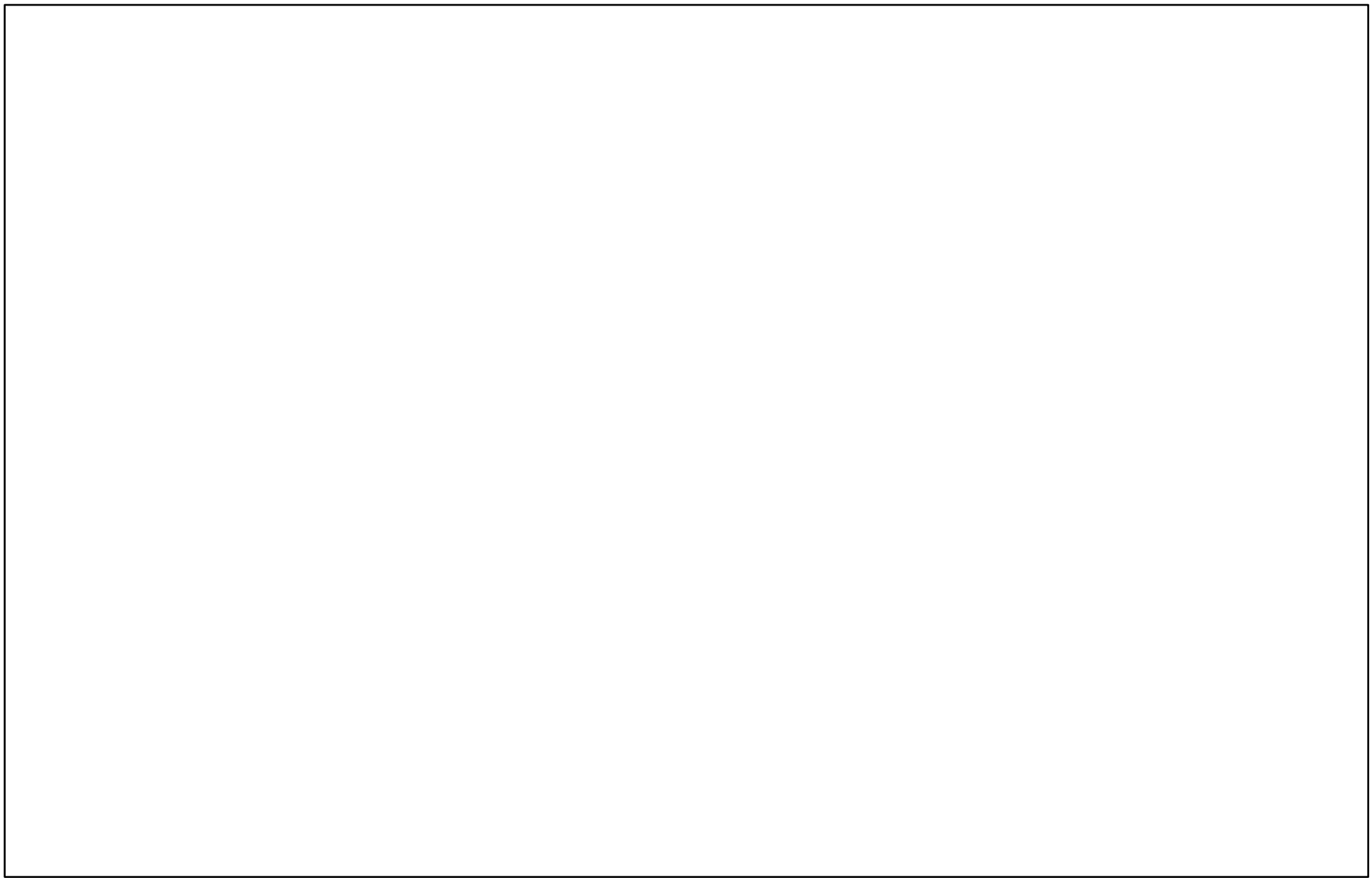
- Open-source project
  - fully Java-based technology, available at http://www.alice.unibo.it/projects/tucson
    - current version: 1.4.5 (minimal support to RBAC)
- Main parts
  - TuCSoN API
    - API for developing agents accessing & using TuCSoN services
      - Java & Prolog support
  - TuCSoN Service
    - runtime to be installed on hosts to make them TuCSoN nodes
  - TuCSoN Tools
    - Inspector
      - controlling and monitoring tuple centre behaviour
    - Shell
      - to let humans interact with tuple centres

TuCSoN API: OVERVIEW

- Package alice.tucson.api
  - classes for exploiting TuCSoN services
  - interface TuCSoNContext defining basic coordination primitives
    - out, in, rd, inp, rdp, set_spec, get_spec
  - entry class Tucson to get an ACC, implementing the TuCSoNContext interface
  - tuProlog Library to define Prolog agents using TuCSoN
  - auxiliary classes, for creating simple Agents and Automata
- Package alice.logictuple
  - interfaces and classes implementing logic tuples
  - class LogicTuple
    - implementing a logic tuple (tuple + template)
  - class TupleArgument, Value, Var to implement tuple parts
- See manual + Javadoc for details

HELLO WORLD EXAMPLE

ANOTHER EXAMPLE



A SIMPLE AGENT

TuCSoN IN PROLOG

```
'alice.tucson.api.Tucson2PLibrary'
```

TuCSoN SERVICE & TOOLS

- Package alice.tucson.service
  - containing the TuCSoN service side (daemon)
  - class alice.tucson.service.Node as main class to install TuCSoN Service
- Package alice.tucson.tools
  - containing TuCSoN tools
  - class alice.tucson.tools.Shell
    - as a shell with a command line interface to directly interact with tuple centres
  - class alice.tucson.tools.Inspector
    - as inspector to monitor, control and change tuple centre content and behaviour

GETTING STARTED: TuCSoN DOWNLOAD

- TuCSoN distribution can be downloaded from http://www.alice.unibo.it/projects/tucson
  - download section, distribution file: tucson-1.4.5.zip
- Once downloaded, choose a directory and unzip the file
  - we refer to such a directory as <TUCSON_HOME>
- The distribution contains the following subfolders:
  - lib
    - containing the tucson library (tucson.jar) and the library containing the examples (examples.jar)
  - src
    - containing sources of the examples
  - doc
    - containing the manual and the API Javadoc

INSTALLING THE SERVICE

- Once downloaded and unzipped, we are ready to install the TuCSoN Service on the current host
  - making current host a TuCSoN node
- Open a shell, change the dirs to <TUCSON_HOME>, then digit:

```
java -cp lib/tucson.jar alice.tucson.service.Node
```

- If everything goes well, the TuCSoN daemon is installed on the node
  - ...and the node becomes a TuCSoN node...

TuCSoN SERVICE CHECK

- In order to check if a TuCSoN is installed on a certain node, you can type:

```
telnet <Node IP name or Address> 20504
```

- TuCSoN by default uses port 20504: the telnet command triggers a TuCSoN daemon to send an ack.

LAUNCHING THE FIRST EXAMPLE

- Open a new shell, move to <TUCSON_HOME> and then type

```
java -cp lib/tucson.jar:lib/examples.jar
            alice.tucson.examples.basic.HelloWorld

   NOTE: use ; instead of : as path separator if you
   are on Windows OS
```

- This will launch the basic HelloWorld application which inserts a tuple on the default tuple centre of localhost

A LOOK TO THE HelloWorld CODE

```
package alice.tucson.examples.basic;

import alice.tucson.api.*;
import alice.logictuple.*;

public class HelloWorld {

  public static void main(String[] args) throws Exception {

    TupleCentreId tid = new TupleCentreId("default");
    TucsonContext cnt = Tucson.enterDefaultContext();

    long now = System.currentTimeMillis();
    LogicTuple tuple = new LogicTuple("msg",
                                      new Value("Hello world!"),
                          new Value("time",new Value(now)));

     cnt.out(tid,tuple);
     System.out.println("Tuple inserted: "+tuple);

     LogicTuple template = new LogicTuple("msg",new Var("Msg"),new Var("Time"));
     LogicTuple msg = cnt.in(tid,template);

     System.out.println("Tuple retrieved name: "+msg.getName());
     System.out.println("Msg argument: "+msg.getArg(0));
     System.out.println("Time argument: "+msg.getArg(1).getArg(0));
  }
}
```

"BEING LIKE AN AGENT": USING THE SHELL TOOL

- Open a new shell, move on the <TUCSON_HOME> then type

```
java -cp lib/tucson.jar alice.tucson.tools.Shell
```

- A shell is launched, enabling the possibility for humans to interact with tuple centres by means of commands
  - Examples:
    - out(p(1))
      - inserts p(1) in the default tc on localhost
    - board ? out(temp_array(1,13.5))
      - inserts a tuple in the board tuple centre on localhost
    - board @ '137.204.107.188' ? out(temp_array(1,13.5))
      - inserts a tuple on the board tuple centre on the specified TuCSoN node

USING THE INSPECTOR

- Open a new shell, move on the <TUCSON_HOME> then type

```
java -cp lib/tucson.jar alice.tucson.tools.Inspector
```

- The inspector tool is launched, ready for inspecting any tuple centre of any node
  - Let's inspect the default tuple centre on localhost
- After selecting the tuple centre, the inspector provides the various views to monitor and control the tuple centre
  - tuple set view, pending query view, reaction view
  - behaviour specification view (VERY IMPORTANT...)

MONITORING DYNAMICS

- Let's use the inspector to monitor a simple application dynamics: *The Dining Philosopher Example*
- Open a shell, change the dir to <TUCSON_HOME> and type:

```
java -cp lib/tucson.jar:lib/examples.jar
                    alice.tucson.examples.philo.PiloRun

    NOTE: use ; instead of : as path separator if you
    are on Windows OS
```

- This will launch the dining philosophers, which use the table tuple centre to coordinate
- We can use the inspector to monitor table tuple centre dynamics