tATAmI: A Platform for the Development and Deployment of Agent-Based AmI Applications

Andrei Olaru[b], **Marius-Tudor Benea**[a,b,*], Amal El Fallah Seghrouchni[a], Adina Magda Florea[b]

*E-mail: tudorbenea@gmail.com

[a]LIP6 – University Pierre and Marie Curie (UPMC), France
[b]CS Department – University Politehnica of Bucharest (UPB), Romania

ANT 2015

June 4, 2015

Outline

Introduction

- **Ambient Intelligence** (AmI) – *empowering people by means of* distributed *digital environments sensitive, adaptive, and responsive to human needs, habits, gestures, and emotions*. Acampora, G., et al., 2013
- AmI applications' properties:
  - intrinsic distribution of the architecture;
  - dynamic underlying topologies;
  - frequent changes in execution context.

Introduction

- **Ambient Intelligence** (AmI) – *empowering people by means of* distributed *digital environments sensitive, adaptive, and responsive to human needs, habits, gestures, and emotions*. Acampora, G., et al., 2013
- AmI applications' properties:
  - intrinsic distribution of the architecture;
  - dynamic underlying topologies;
  - frequent changes in execution context.

Agent-oriented approach!

Introduction

- **Ambient Intelligence** (AmI) – *empowering people by means of* distributed *digital environments sensitive, adaptive, and responsive to human needs, habits, gestures, and emotions*. Acampora, G., et al., 2013
- AmI applications' properties:
  - intrinsic distribution of the architecture;
  - dynamic underlying topologies;
  - frequent changes in execution context.

  Agent-oriented approach!
- **The problem:** the lack of a good agent-based development framework for AmI that allows the development of applications that:
  - are deployed on a **hybrid network** of devices (*mobile platforms*!);
  - offers a good **context representation** and detection mechanism;
  - is **interoperable** with other platforms (sensors, cloud services, etc.);
  - is **modular**;
  - allows **quick** and **repeatable** experiments;
  - is **easy to use**, reducing the time to go from design to implementation.
  - facilitates the creation of **powerful user interfaces**.

Introduction

- **Ambient Intelligence** (AmI) – *empowering people by means of* distributed *digital environments sensitive, adaptive, and responsive to human needs, habits, gestures, and emotions*. Acampora, G., et al., 2013
- AmI applications' properties:
  - intrinsic distribution of the architecture;
  - dynamic underlying topologies;
  - frequent changes in execution context.

  Agent-oriented approach!
- **The problem:** the lack of a good agent-based development framework for AmI that allows the development of applications that:
  - are deployed on a **hybrid network** of devices (*mobile platforms*!);
  - offers a good **context representation** and detection mechanism;
  - is **interoperable** with other platforms (sensors, cloud services, etc.);
  - is **modular**;
  - allows **quick** and **repeatable** experiments;
  - is **easy to use**, reducing the time to go from design to implementation.
  - facilitates the creation of **powerful user interfaces**.
- **Our solution:** **tATAmI** (`https://github.com/tATAmI-Project`) platform.

tATAmI – towards Agent Technologies for AmI

- development of applications deployable on **networks of Java-enabled** computers and **Android** mobile devices;
- based on **Jade** and **modular**;
- good **context representation** mechanism:
  - *hierarchical organization* of agents & agent management and *mobility* inspired from ambient calculus Cardelli, L., Gordon, D., 2000;
  - *flexible representation of* agent *knowledge* & *powerful knowledge matching* mechanism.
- interoperable with **web services** & exposes agents as web services;
- **quick** and **repeatable execution** of experiments (scenarios described in XML, timeline for events + visualization and simulation);
- complete **separation of** the **GUI** $\Rightarrow$ fertile ground for powerful GUIs;
- **S-CLAIM** language Baljak, V., et al., 2012:
  - *simple* and concerned only with the agent related aspects;
  - inspired from and improving *CLAIM* El Fallah Seghrouchni, A., et al., 2003.

S-CLAIM Code Example

Example (Agent Class Definition)

```
(agent Course ?courseName ?parent
    (behavior
        ...

    )
)
```

S-CLAIM Code Example

Example (Agent Class Definition)

```
(agent Course ?courseName ?parent
    (behavior
        ...

        (initial register
            (send ?parent (struct message managesCourse this ?courseName))
        )

        ...
    )
)
```

S-CLAIM Code Example

Example (Agent Class Definition)

```
(agent Course ?courseName ?parent
    (behavior
        ...
        (reactive changeRoom /*reacts to a message that informs about the new room*/
            (receive scheduling ?courseName ?roomName)
            (addK (struct knowledge scheduling ?courseName ?roomName))
            (if (readK (struct knowledge roomAgent ?roomName ?roomAgentName))
                then
                    (forAllK (struct knowledge userAgent ?userName ?userAgentName)
                        (send ?userAgentName (struct message scheduling ?courseName
                            ?roomAgentName))
                    )
                    (in ?roomAgentName)
                else
                    (send ?parent (struct message whoManagesRoom this ?roomName))
            )
        )
        ...
    )
)
```

S-CLAIM Code Example

Example (Agent Class Definition)

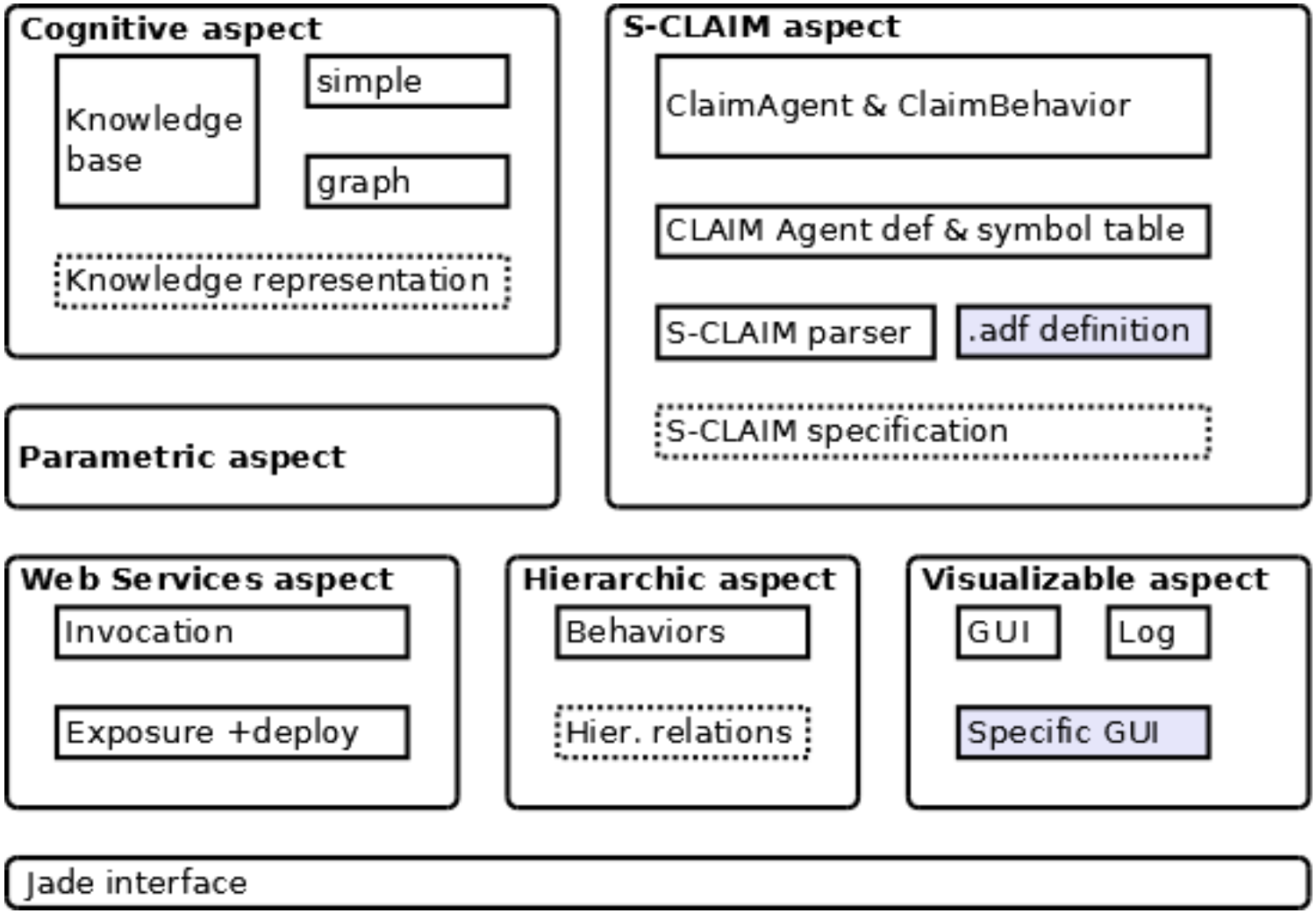
```
(agent Course ?courseName ?parent
    (behavior
        ...

        (proactive verifyStartingCondition
            (condition (not (readK (struct knowledge courseStarted))))
            ... // assign values to ?studentsInRoom, ?minNoOfStudents
                        // and ?professorAgent based on the KB
            (if (greaterOrEqual ?studentsInRoom ?minNoOfStudents)
                then
                    (calculatePercent ?result ?studentsInRoom ?minNoOfStudents)
                    /*the professor is informed that the course can start*/
                    (send ?professorAgent (struct message presentStudents ?result))
            )
            (wait 60000)
        )

        ...
    )
)
```

Outline

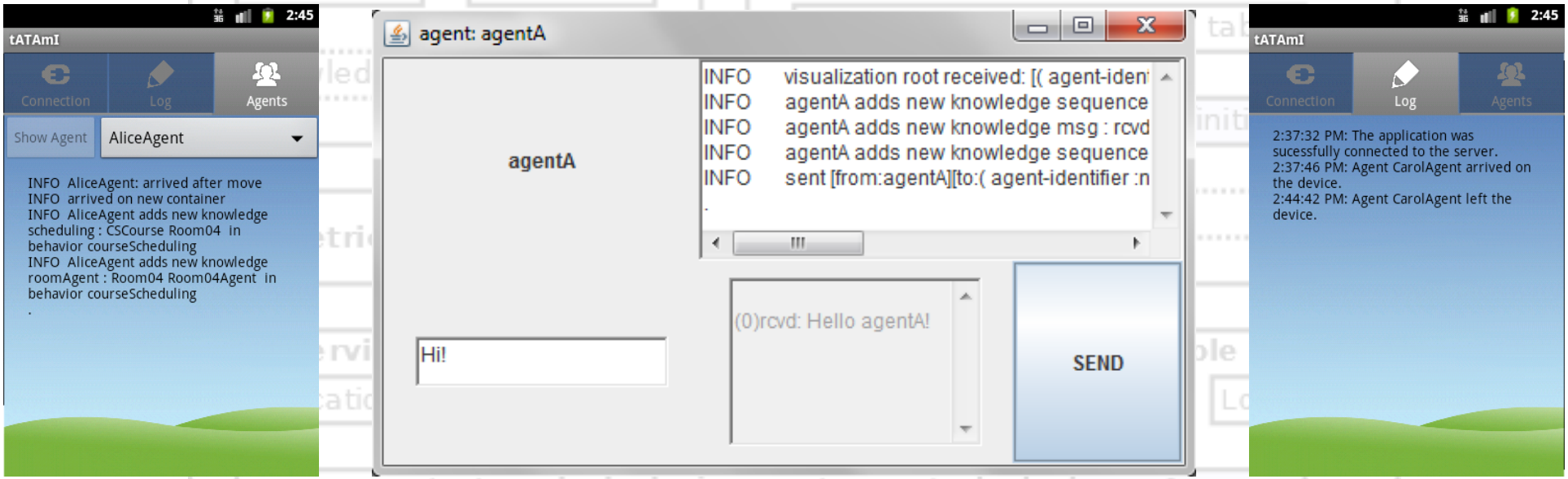
Structure of the Platform

- **Agent**
  - core of tATAmI.
- **Simulation**
  - deployment of agents and simulated events;
  - simulation = complete execution of the agent system, from start to complete halt;
  - entire deployment configured through an XML scenario file.
- **Visualization**
  - provides the user with information on the agent system at a glance.

Agent Segment

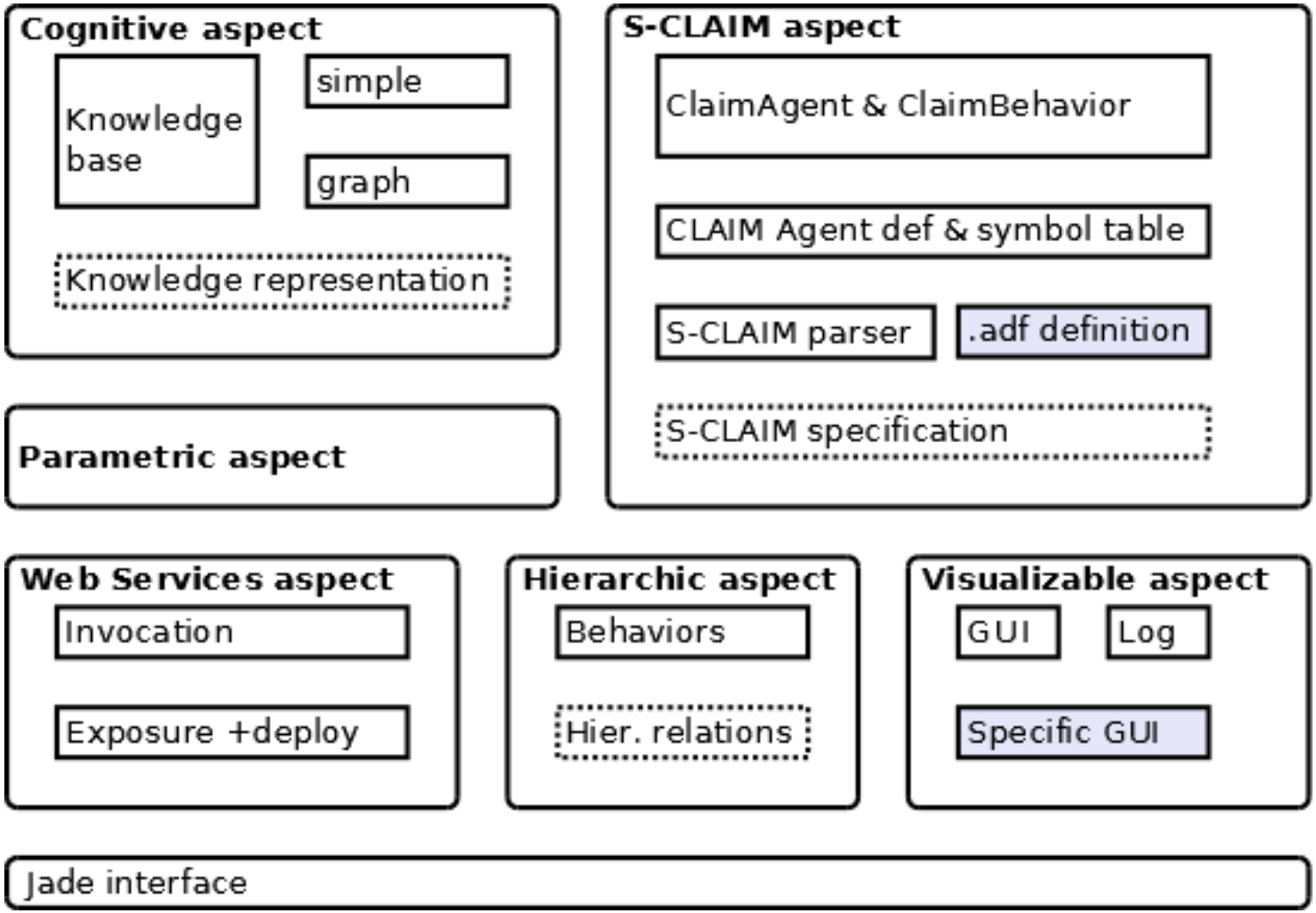


Agent Segment

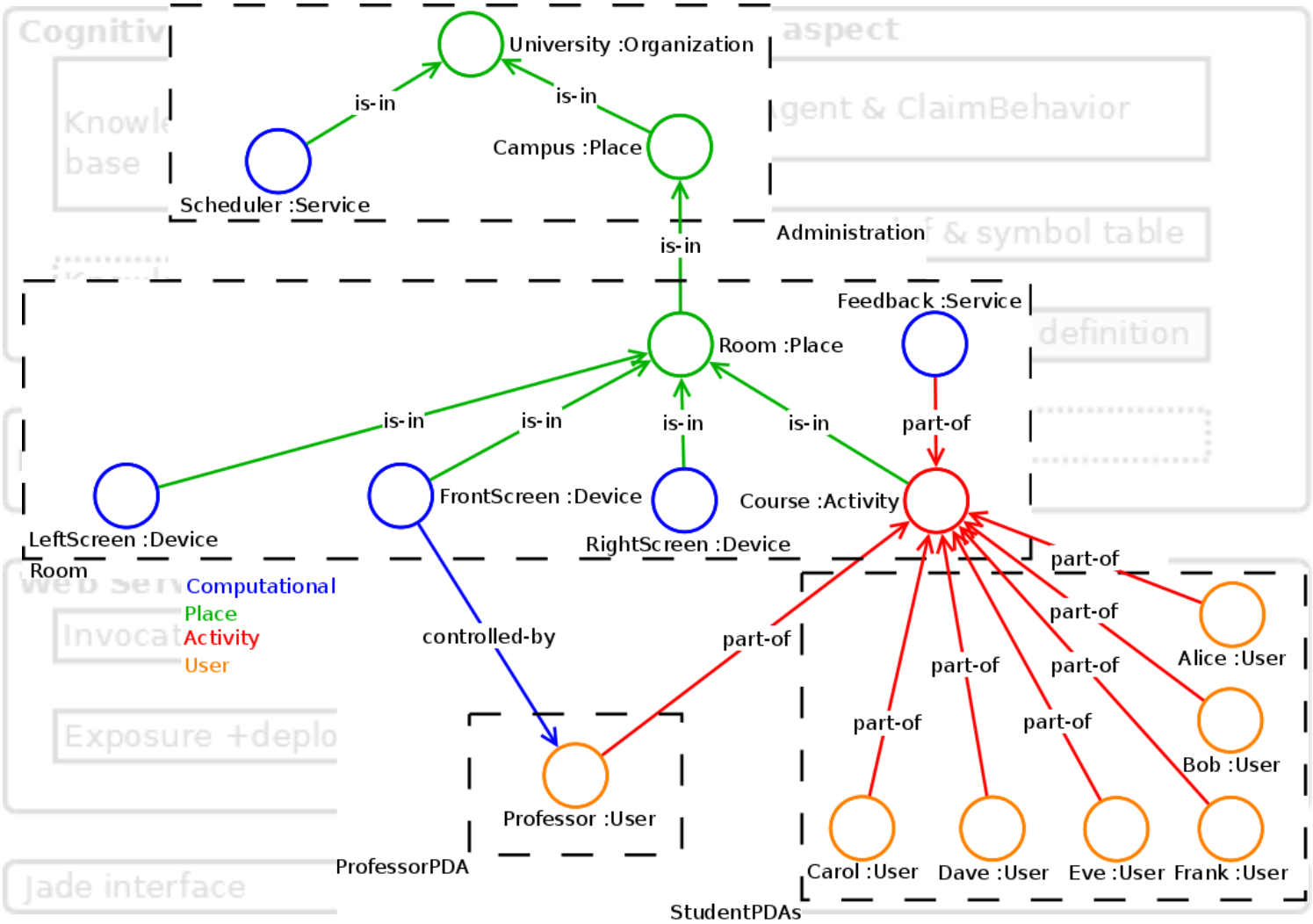


Agent **GUI** and **Log** examples (Visualizable aspect)

Agent Segment

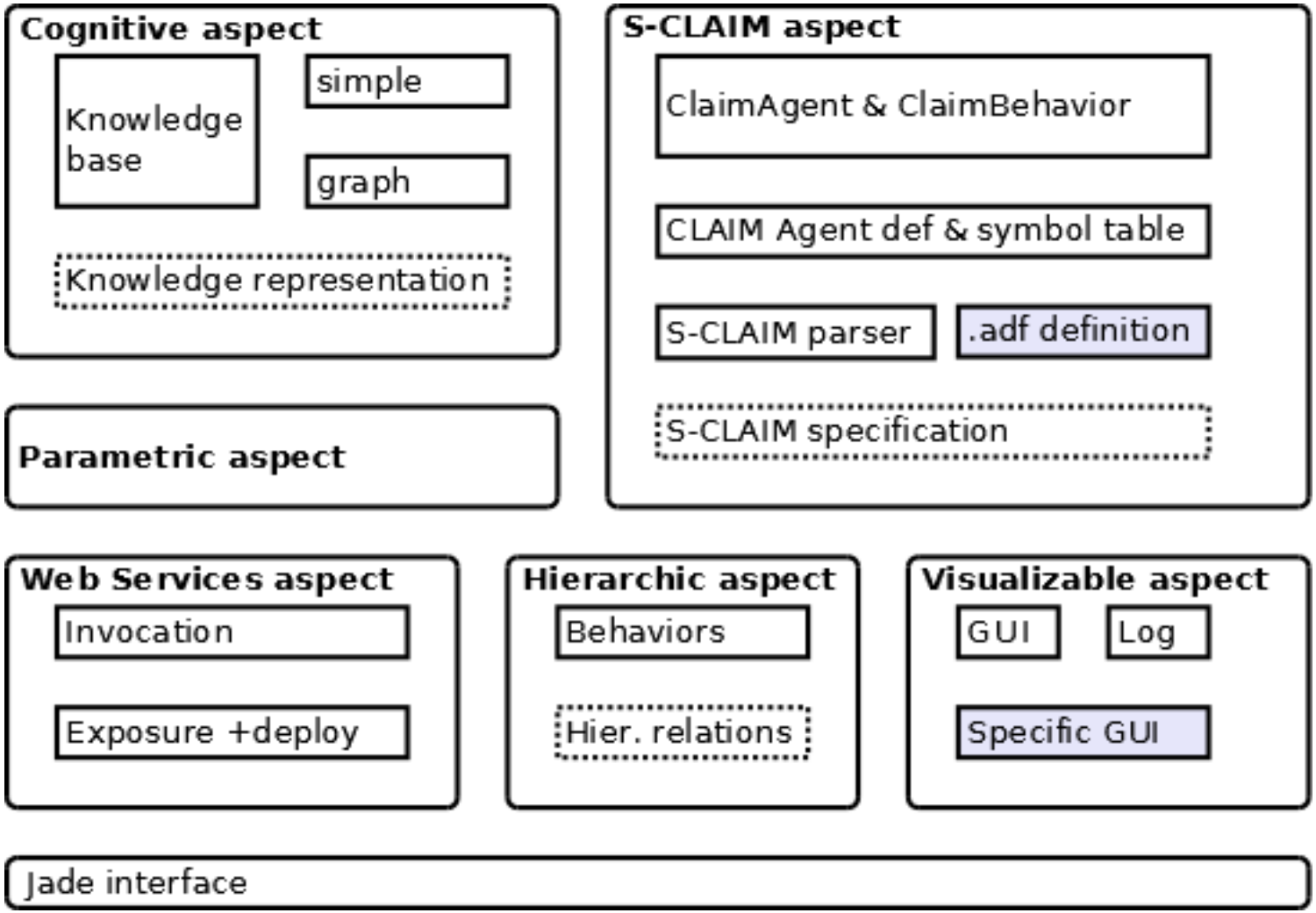


Agent Segment

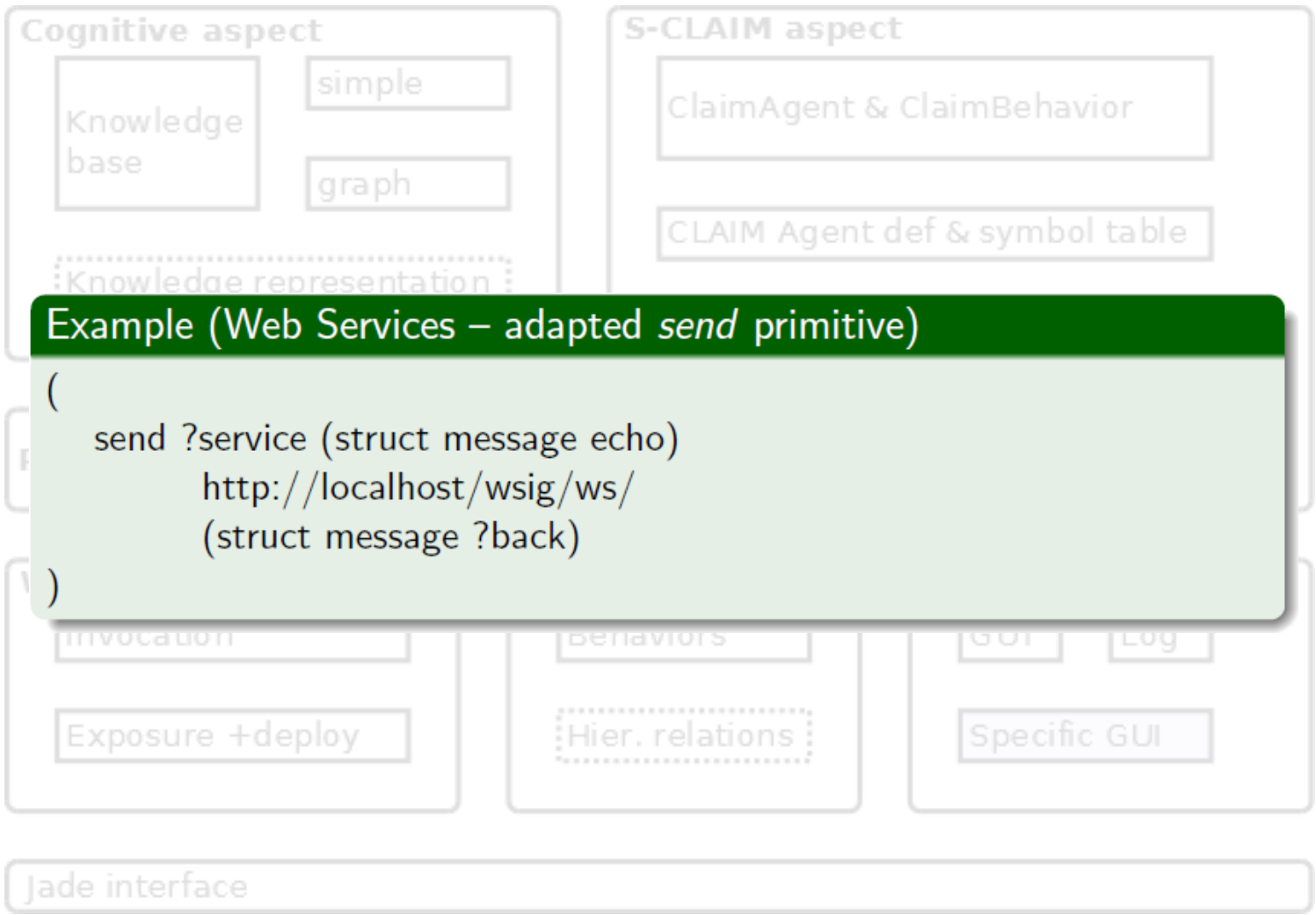


Agentification of **SmartRoom** scenario (Hierarchic organization)

Agent Segment



Agent Segment

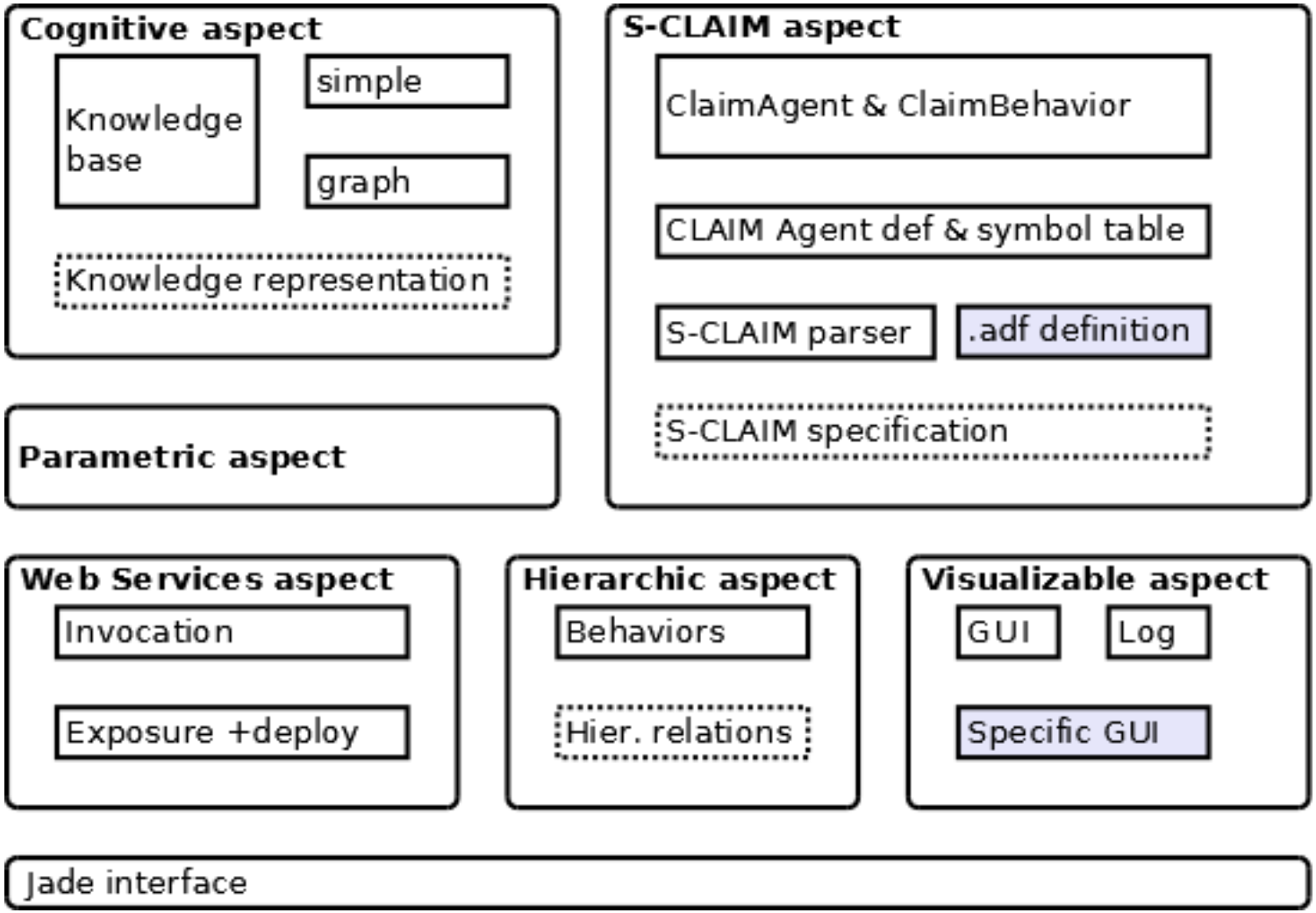


Example (Web Services – adapted *send* primitive)

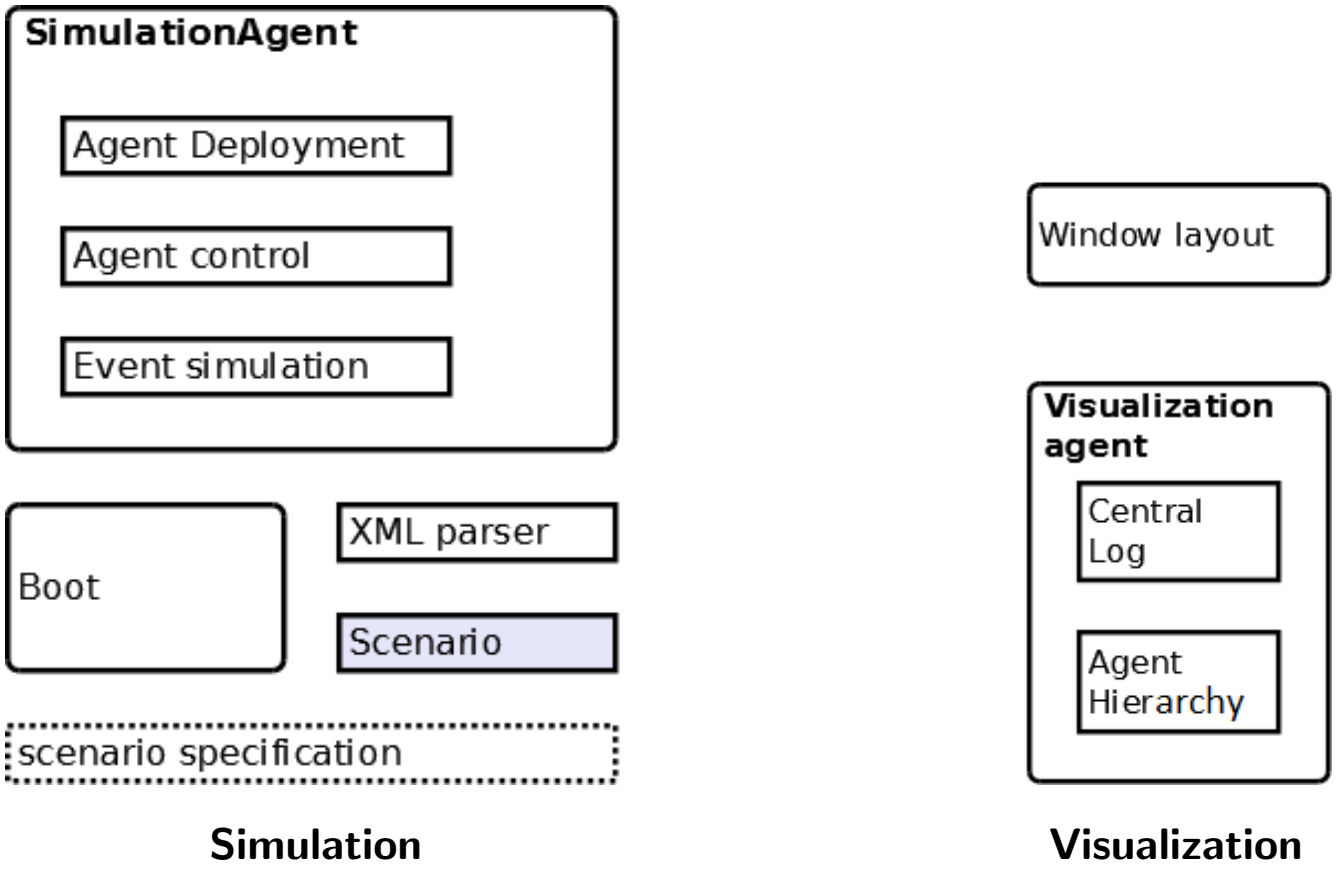
```
(
    send ?service (struct message echo)
            http://localhost/wsig/ws/
            (struct message ?back)
)
```

Modified S-CLAIM *send* primitive for interoperability with **web services**

Agent Segment



Simulation & Visualization Segments

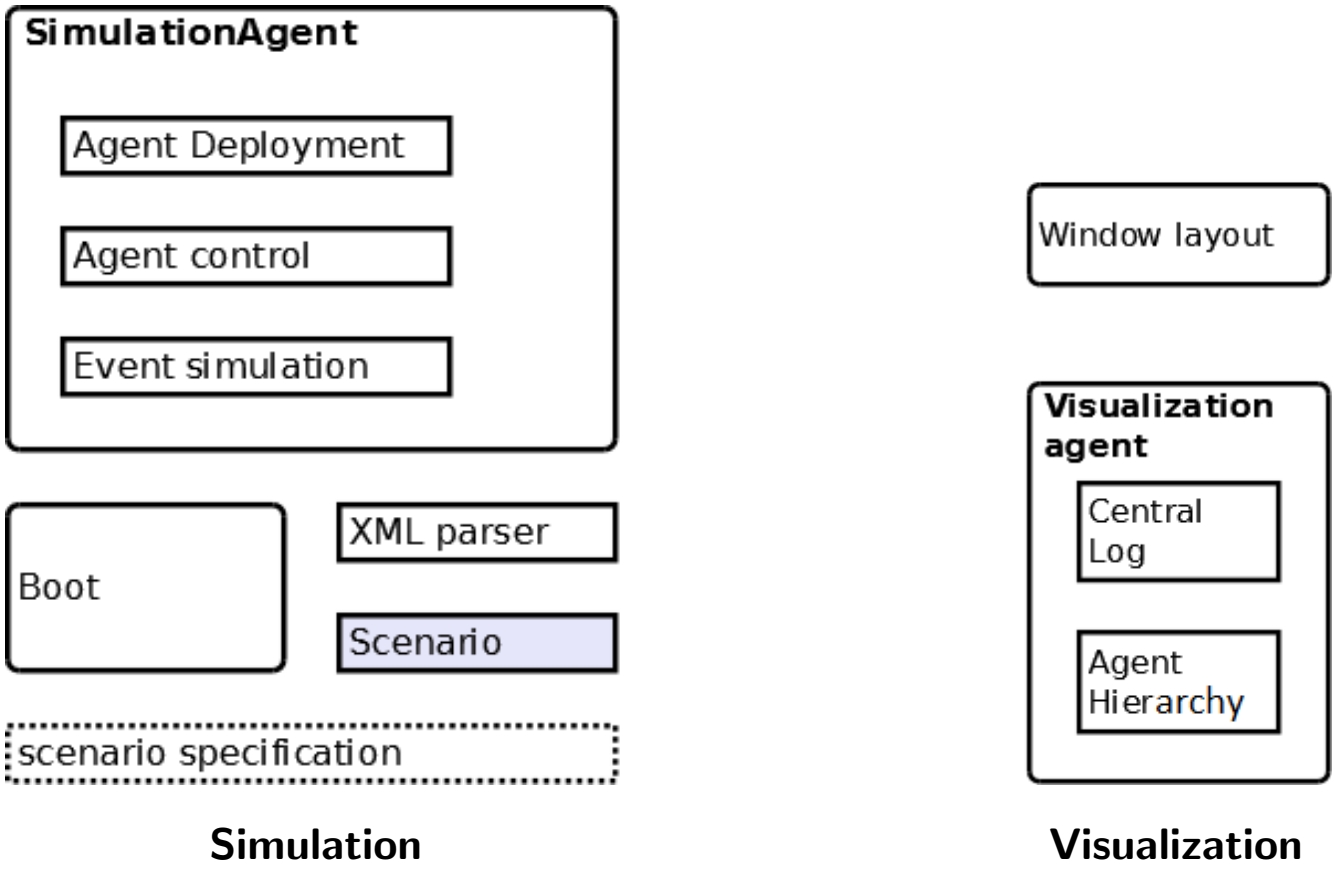


Simulation & Visualization Segments

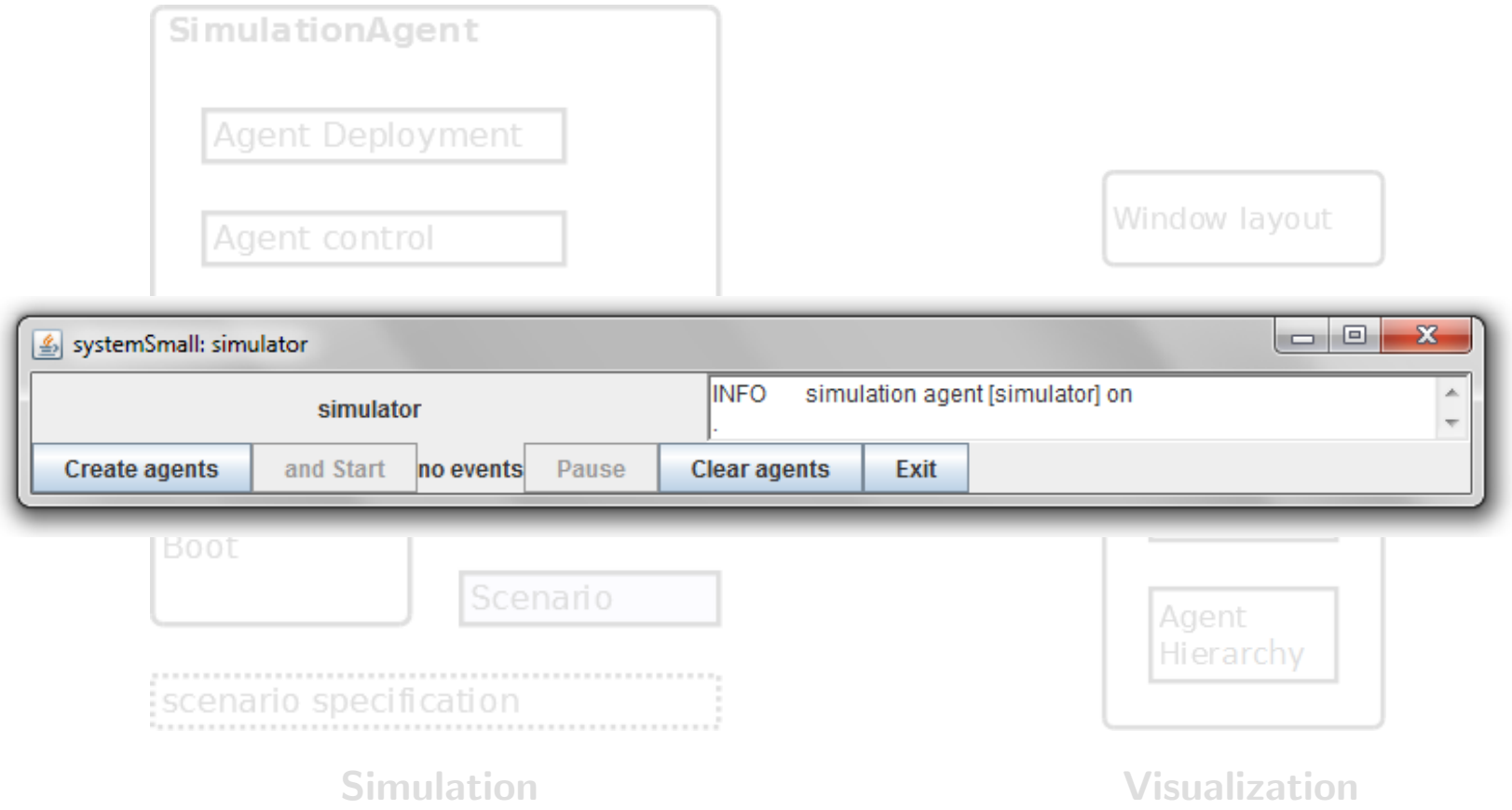
```
1   <scen:jadeConfig mainContainerName="Administration" platformID="SmartRoom" />
2   <scen:adfPath>scenario/2013/SmartRoom-EMAS</scen:adfPath>
3   <scen:agentPackage>agent_packages.example.smartRoom</scen:agentPackage>
4
5   <scen:initial>
6     <scen:container name="AliceContainer" create="false">
7       <scen:agent>
8         <scen:parameter name="loader" value="adf2" />
9         <scen:parameter name="class" value="StudentAgent" />
10        <scen:parameter name="name" value="AliceAgent" />
11        <scen:parameter name="parent" value="MASCourseAgent" />
12        <scen:parameter name="userName" value="Alice" />
13        <scen:parameter name="fixed" value="true" />
14        <scen:parameter name="GUI" value="UserAgentGUI" />
15      </scen:agent>
16    </scen:container>
17  </scen:initial>
18  <scen:timeline>
19    <scen:event time="2000">
20      <scen:CLAIMMessage>
21        <scen:to>SchedulerUPMCAgent</scen:to>
22        <scen:protocol>newSchedule</scen:protocol>
23        <scen:content>
24            ( struct message newSchedule ( struct knowledge scheduledTo CSCourse Room04 ) )
25        </scen:content>
26      </scen:CLAIMMessage></scen:event>
27  </scen:timeline>
```

Example **XML scenario** file

Simulation & Visualization Segments

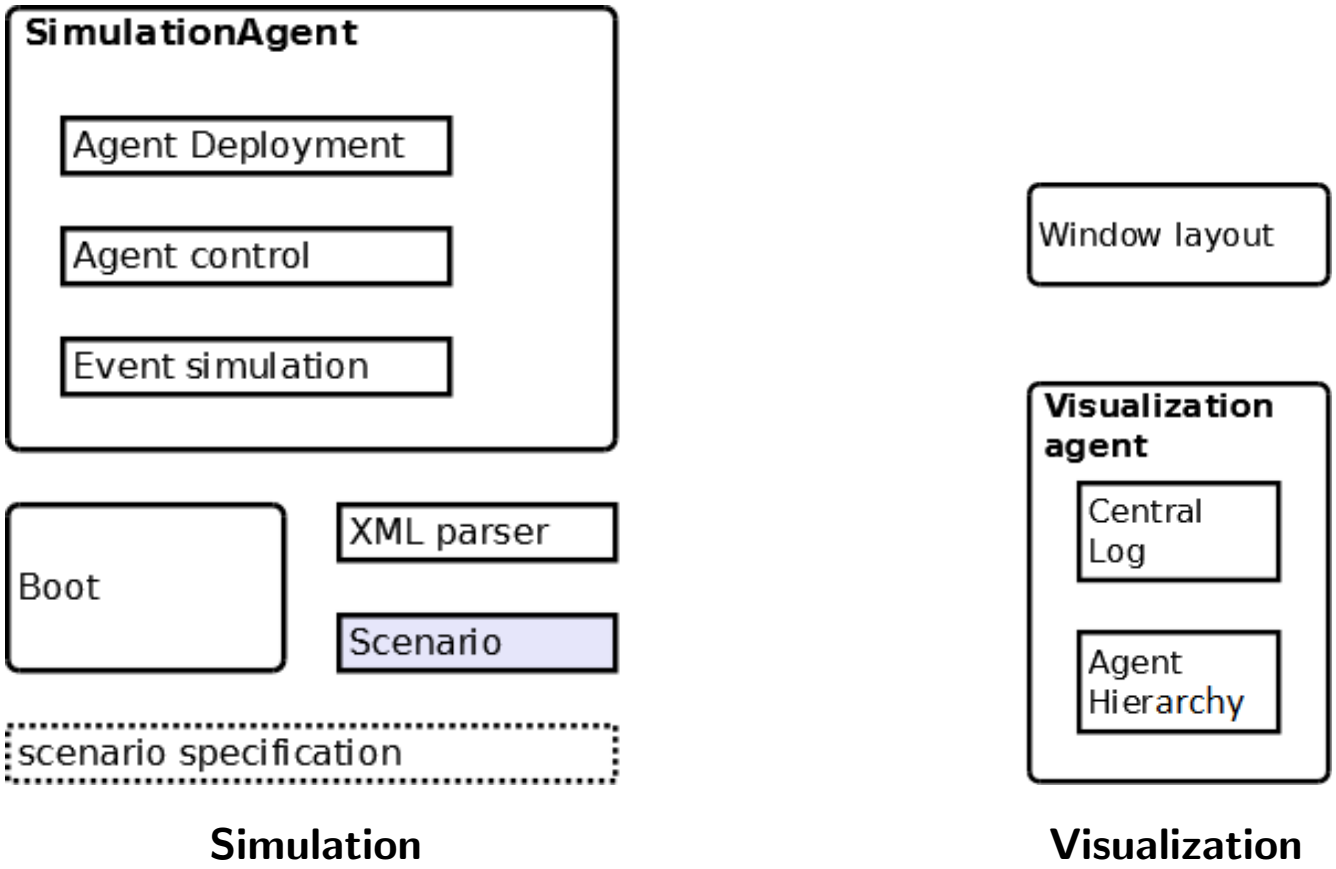


Simulation & Visualization Segments

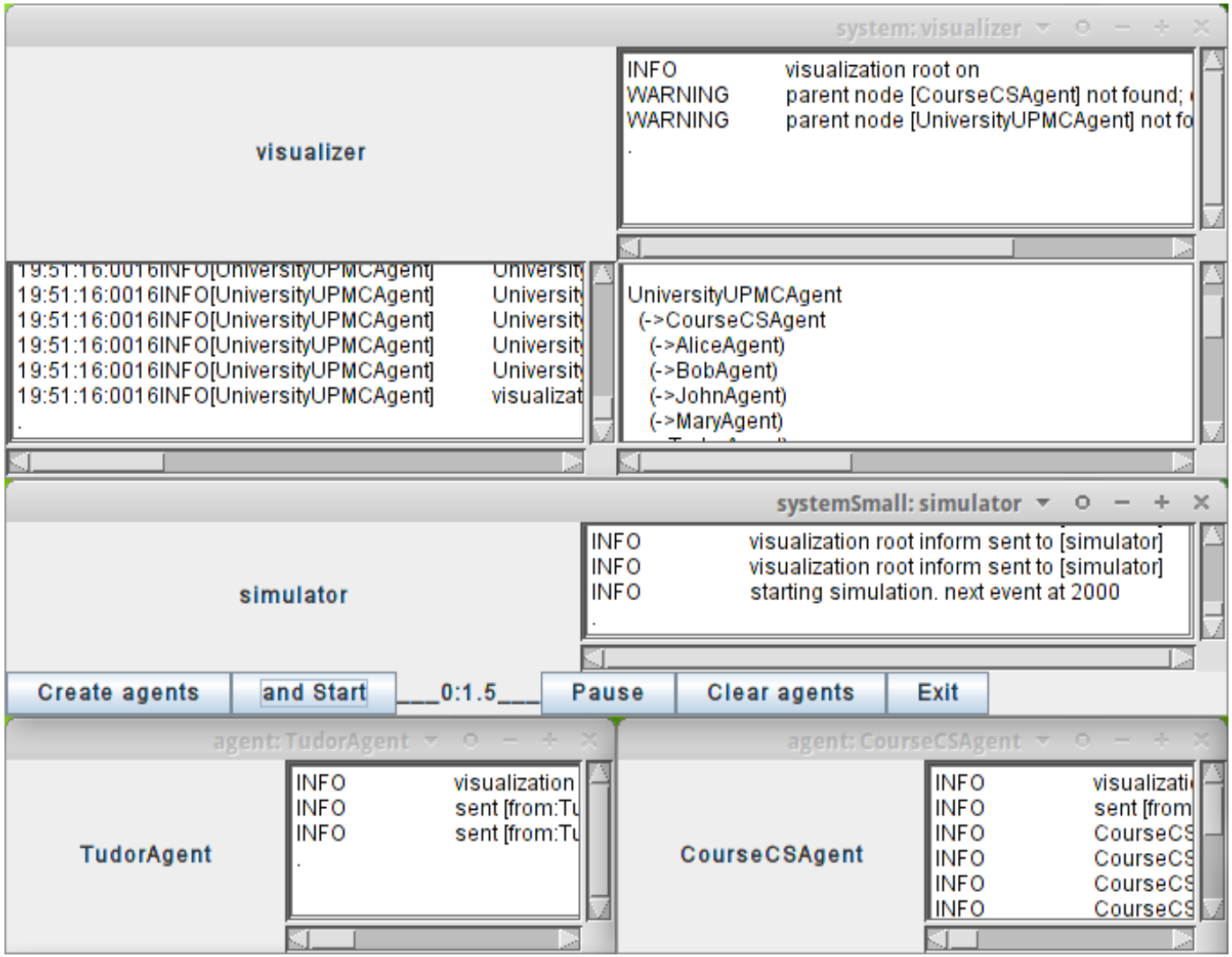


GUI of the **Simulator** agent

Simulation & Visualization Segments



Simulation & Visualization Segments



Window layout (simulator, visualizer & two user defined agents)

Outline

Syamisen (SmartRoom) Scenario

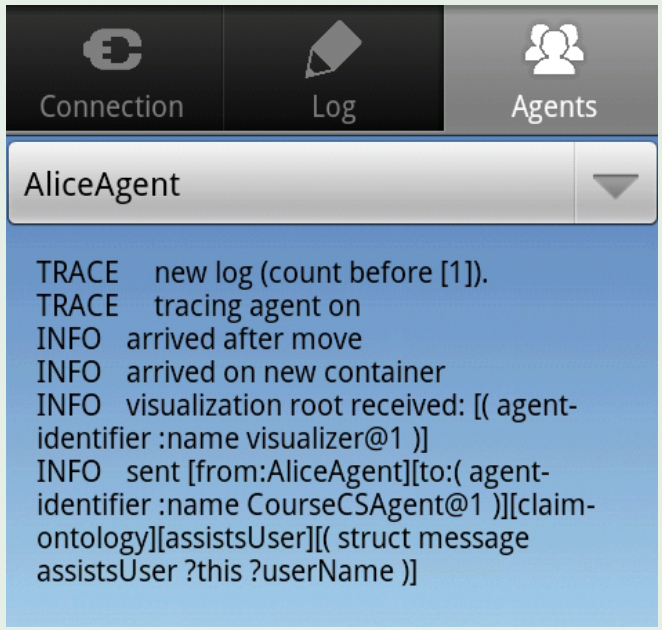
– S-CLAIM + tATAmI Baljak, V., et al., 2012;
– **Syamisen** (SmartRoom) **Scenario** (`http://webia.lip6.fr/~aodai/videos/aodai.wmv`):

- Alice is informed that the room for the CS course that she attends has changed;
- At the hour set for the course, the professor is in the room;
- Based on a global situation of the students, available on his PDA, he decides to start the course;
- The room is configured for the presentation and the presentation begins;
- After the course the students are involved in some hands-on activities;
- After a pre-established interval of time, the teacher evaluates the results of the activities;
- The course ends and everything turns off;
- The students leave feedback when the Feedback agent comes to their PDAs in order to ask for it.

Smart Room in NII, Japan

Syamisen - example

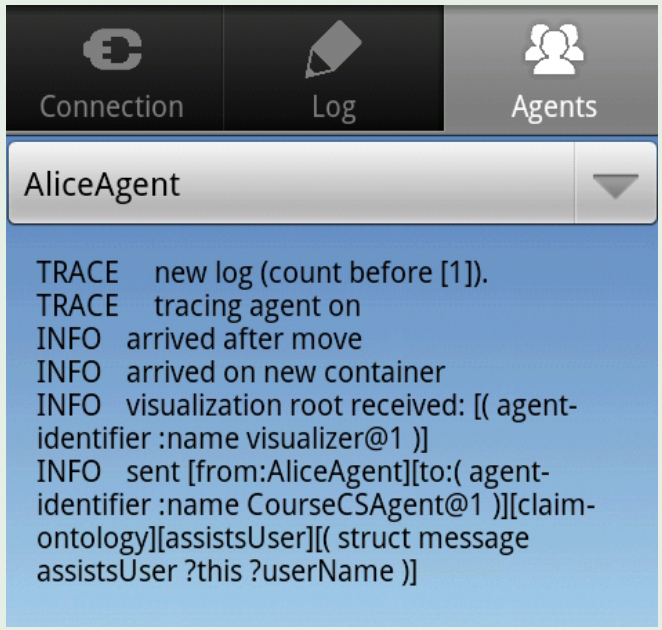
Example (Migration *PC* $\longleftrightarrow$ *Android* of an agent)



Syamisen - example

Example (Migration *PC* $\longleftrightarrow$ *Android* of an agent)

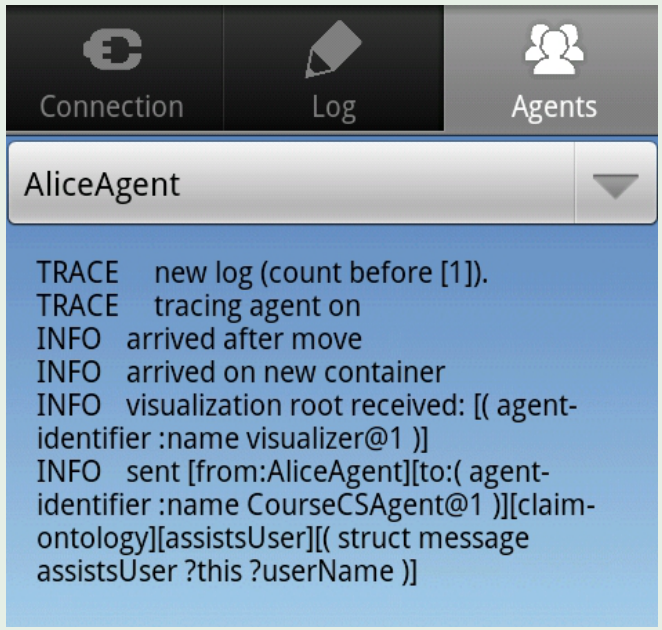
```
<scen:timeline>
  <scen:event time="2000" >
    <scen:CLAIMMessage>
      <scen:to>SchedulerUPMCAgent</scen:to>
      <scen:protocol>newSchedule</scen:protocol>
      <scen:content>
      ( struct message newSchedule
        ( struct knowledge scheduledTo CSCourse Room04 )
      )
      </scen:content>
    </scen:CLAIMMessage></scen:event>
</scen:timeline>
```



Syamisen - example

Example (Migration *PC* $\longleftrightarrow$ *Android* of an agent)

```
<scen:timeline>
  <scen:event time="2000" >
    <scen:CLAIMMessage>
      <scen:to>SchedulerUPMCAgent</scen:to>
      <scen:protocol>newSchedule</scen:protocol>
      <scen:content>
      ( struct message newSchedule
        ( struct knowledge scheduledTo CSCourse Room04 )
      )
      </scen:content>
    </scen:CLAIMMessage></scen:event>
</scen:timeline>
```

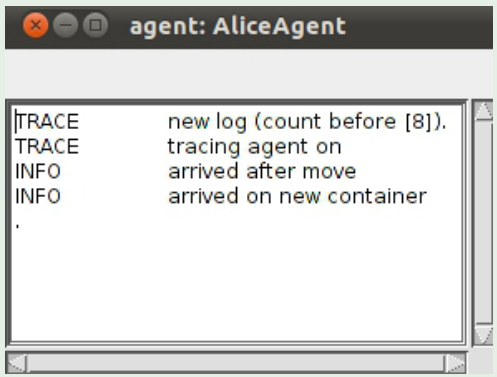


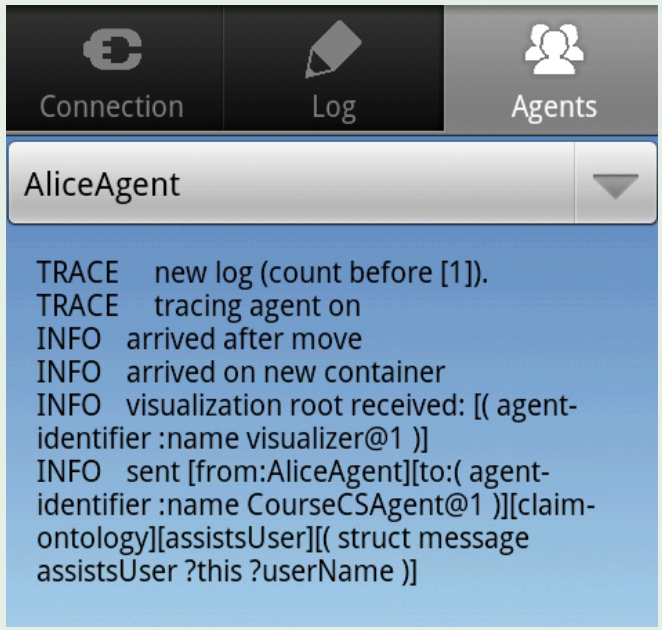

```
20:39:05:0624 INFO  [AliceAgent]:    sent [from:AliceAgent][to:( agent-identifier :name CourseCSAgent@1 )]
20:39:13:0767 INFO  [AliceAgent]:    AliceAgent must migrate to RoomContainer
20:39:13:0773 INFO  [AliceAgent]:    moving to [RoomContainer]
20:39:13:0777 INFO  [AliceAgent]:    moving to [RoomContainer@<Unknown Host>]
20:39:13:0783 TRACE [AliceAgent]:    log out (logs remaining [0]).
```

Syamisen - example

Example (Migration *PC* $\longleftrightarrow$ *Android* of an agent)

```
<scen:timeline>
  <scen:event time="2000" >
    <scen:CLAIMMessage>
      <scen:to>SchedulerUPMCAgent</scen:to>
      <scen:protocol>newSchedule</scen:protocol>
      <scen:content>
      ( struct message newSchedule
        ( struct knowledge scheduledTo CSCourse Room04 )
      )
      </scen:content>
    </scen:CLAIMMessage></scen:event>
</scen:timeline>
```






```
20:39:05:0624 INFO  [AliceAgent]:  sent [from:AliceAgent][to:( agent-identifier :name CourseCSAgent@1 )]
20:39:13:0767 INFO  [AliceAgent]:  AliceAgent must migrate to RoomContainer
20:39:13:0773 INFO  [AliceAgent]:  moving to [RoomContainer]
20:39:13:0777 INFO  [AliceAgent]:  moving to [RoomContainer@<Unknown Host>]
20:39:13:0783 TRACE [AliceAgent]:  log out (logs remaining [0]).
```

Pro-Con Debate

- successful diploma project with students at UPB;
- **workstations** + **Android** devices;
- **Scenario:**
  - **discussion groups** ('Pro' and 'Con') of students;
  - students **add opinions**, using their smartphones;
  - the students **change groups**, after a while;
  - their **opinions move automatically** together with the students.

Outline

Conclusion & Future work

Conclusions:

- we have introduced the **tATAmI platform**:
  - easy and quick implementation and execution of mobile MAS;
  - deployable on heterogeneous networks of Java and Android devices;
  - interoperable with web services;
  - supporting S-CLAIM, a light-weight AOP language;
  - offering a powerful context representation mechanism, through pattern-based knowledge matching and ambient calculus inspired agent management and mobility.
- we described **two case studies** of using tATAmI.

Future work:

- increase the **usability** – better documentation and maintenance;
- improve **web services** support **for Android** devices;
- creation of a **library of** commonly used **functions**;
- **goal-oriented** aspect of agents further explored.

Questions & Answers

Thank you!

References

- **tATAmI platform:** https://github.com/tATAmI-Project
- **S-CLAIM language:** [Baljak et al., 2012]
- **2nd version of tATAmI:** [Olaru, 2015]

Baljak, V., Benea, M. T., Seghrouchni, A. E. F., Herpson, C., Honiden, S., Nguyen, T. T. N., Olaru, A., Shimizu, R., Tei, K., and Toriumi, S. (2012).

S-CLAIM: An Agent-based Programming Language for AmI, A Smart-Room Case Study.

Procedia Computer Science, 10:30–37.

Olaru, A. (2015).

tATAmI-2 – a Flexible Framework for Modular Agents.

Control Systems and Computer Science (CSCS), 2015 20th International Conference on, pages 703–710.

tudorbenea@gmail.com