Profiling and optimization for Android applications on the tATAmI platform

*UNDERSTANDING THE T**ATA**MI PLATFORM AND THE S-CLAIM LANGUAGE*

Outline

- Intro
- The tATAmI Platform
- S-CLAIM
- An Example Scenario (ProCon Android App)

Intro

- **Starting with**
  - A collaborative effort of Andrei Olaru, Thi Thuy Nga Nguyen and Marius-Tudor Benea seeking **a platform for the deployment and testing of AmI applications**.
  - a diploma project that implements a simple scenario (The Android application)
- **The goal is**
  - optimize the platform
  - profile and solve performance issues in the Android app
  - optimize it in order to offer an enjoyable experience to the user

tATAmI

- **t**owards **A**gent **T**echnology for **A**mbient **I**ntelligence
- Designed and built having the following requirements in mind:
  - the use of a programming language for the high-level implementation of agents
  - a modular and extendable structure
  - deployablity on mobile devices
  - traceability and visualization
  - the use of scenario-based simulation
  - the possibility of integration with other platforms and protocols

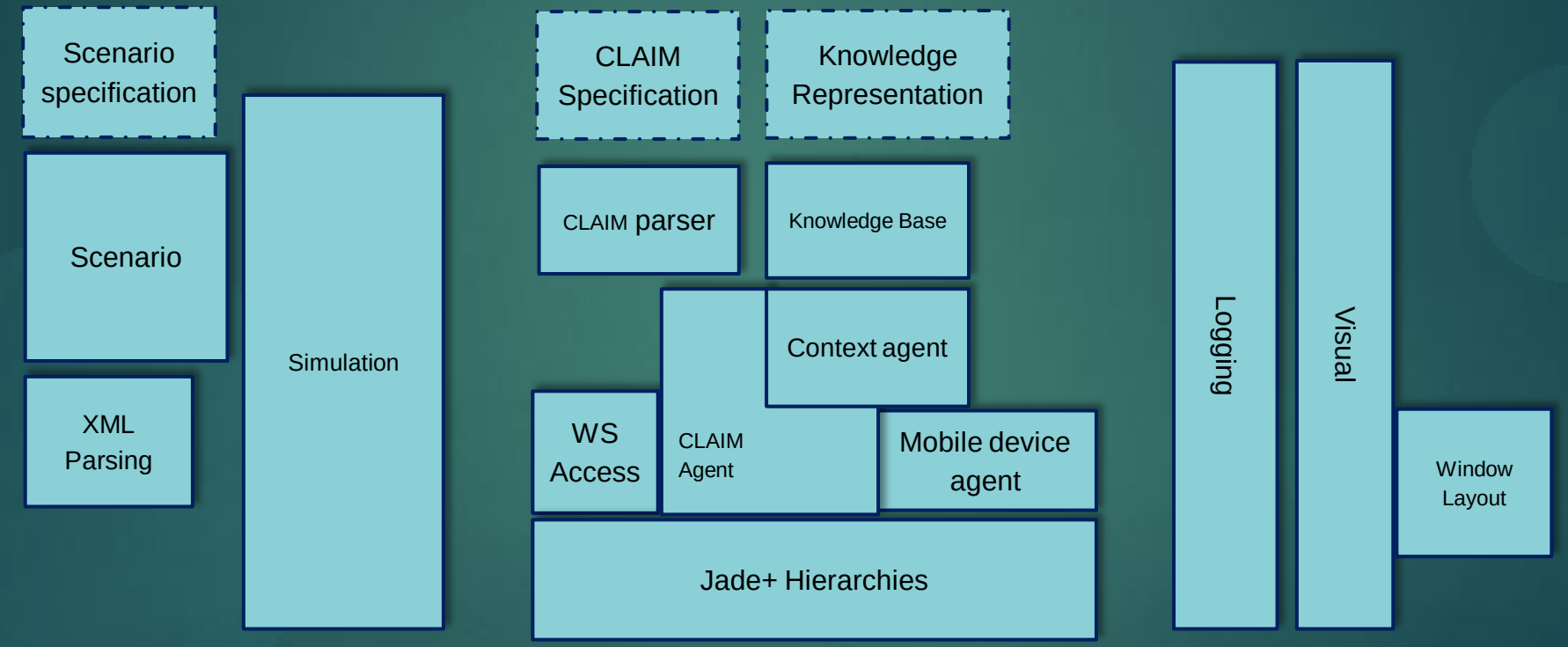
tATAml – Structure

- The **Core** Component:
  - Agent communication, mobility, and management - JADE agents are used.
  - Hierarchical mobility for agents - protocols and behaviors that allow agents to automatically move together with their parents.
  - Web service access
  - S-CLAIM interpretation and execution - a parser for S-CLAIM agent description files
  - Knowledge Base - an interchangeable component that allows access to knowledge through a standard set of functions
  - Context-awareness - use of context matching for problem solving and exchange of relevant context information.

tATAmI – Structure (cont.)

- The ***Simulation*** component:
  - serving for the repeatable execution of scenarios
  - Uses as input XML files that define the execution scenario.
  - Deploys the agents according to the scenarios
- The ***Visualization*** Component
  - Receives log reports and mobility events from agents
  - Displays all agent logs in a centralized, chronological manner
  - Provides components for the automatic layout of agent windows on the screen of the machine they execute on

tATAmI – Structure (cont.)



tATAmI - SCENARIOS

- specified by means of an XML file that contains info about:
  - The initial knowledge of Agents
  - events to generate

- The purpose of a scenario is to reproduce an execution
  - the mentioned information is all that is needed for this execution.

S-CLAIM

- **S**mart **C**omputational **L**anguage for **A**utonomous **I**ntelligent **M**obile agents
- An easy to use high-level declarative Agent-oriented programming language that was created to allow the representation of cognitive skills such as beliefs, goals and knowledge,
- Allows programmers to use the agent-oriented paradigm during the whole process of designing and implementing an AmI application, as it specifies only agent-related components and operations, leaving algorithmic processes aside

S-CLAIM - SEMANTICS

- Communication:
  - send
  - receive
- Mobility:
  - in
  - out
- Agent management:
  - open
  - acid
  - new
- Control primitives:
  - condition
  - if
  - Wait
- Knowledge management
  - addK
  - removeK
  - readK
  - forAllK

S-CLAIM - SYNTAX

- uses the notion of Blocks
  - (<keyword>)
- Variables
  - ?? re-assignable. or ?not
- 3 Types of behaviors
  - initial
  - reactive
  - proactive

```
(agent SimpleAgent ?destination
    (behavior
        (initial sender
            (send ?destination (struct message hello)))))
```

JAVA FUNCTIONS

- There are processes that cannot be easily performed with the default primitives (S-CLAIM), this is why the developer can attach one or more Java class files
- all java-functions share the same signature (except for the name, obviously); they take a vector of values as an argument and return a boolean
- for the agent to use java functions, it has to contain *a parameter* specifying the *.java file which includes the needed functions
- Examples in the next section

A PC/ANDROID SCENARIO (The ProCon Debate App)

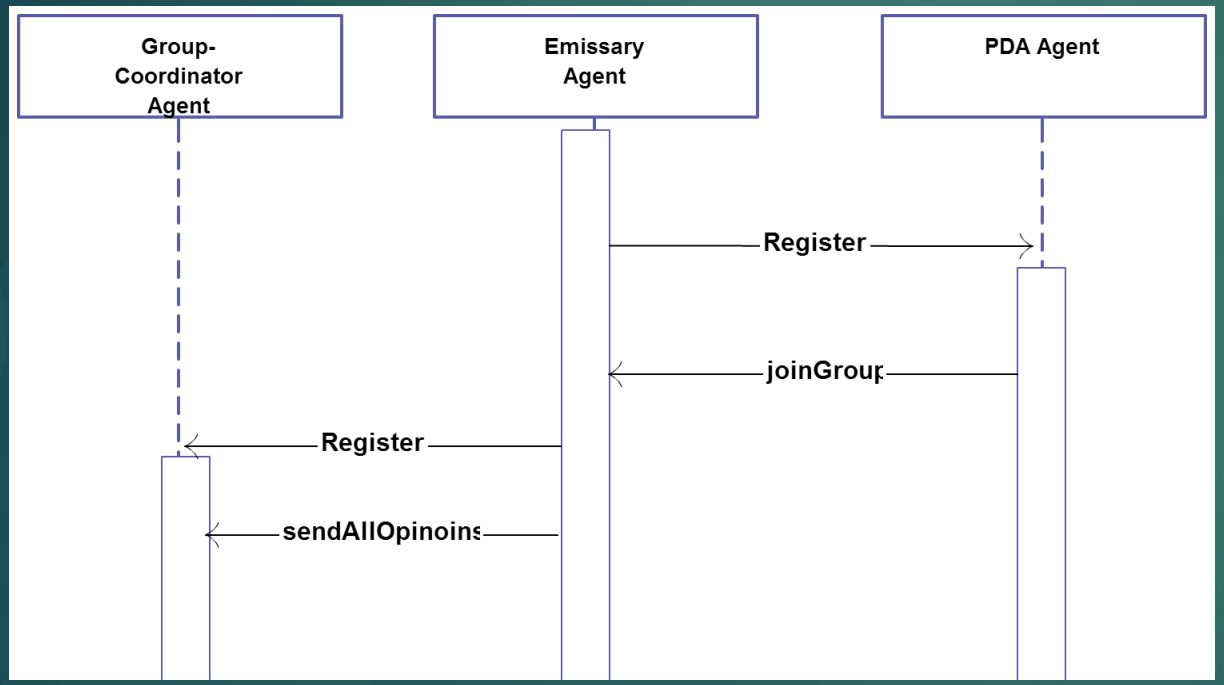
- Allows users to debate over a subject using their mobile java-based devices
- By typing their opinions and sending them
- After classifying each opinion as either positive (Pro) or negative (Con)

A PC/ANDROID SCENARIO (The ProCon Debate App)

- Agent Structure
- There are three types of agents in this application that do all the work:
- PDAAgent
- EmissaryAgent
- GroupCoordinatorAgent

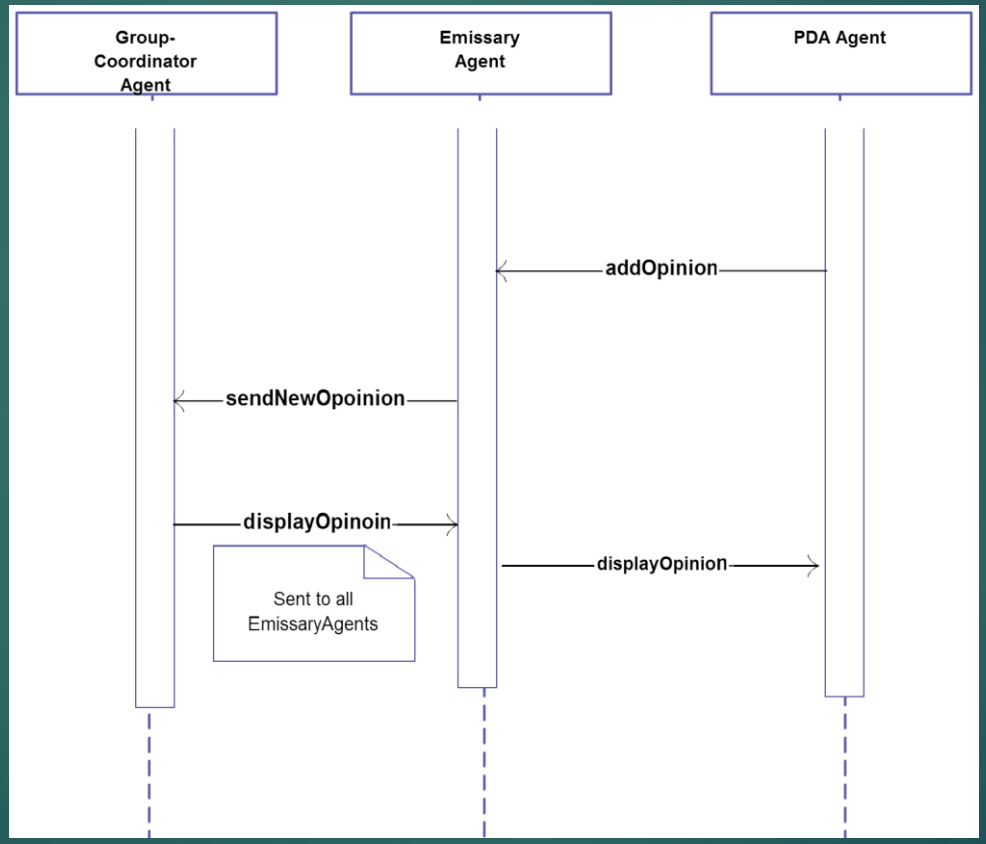
A PC/ANDROID SCENARIO (The ProCon Debate App)

- Joining a Group



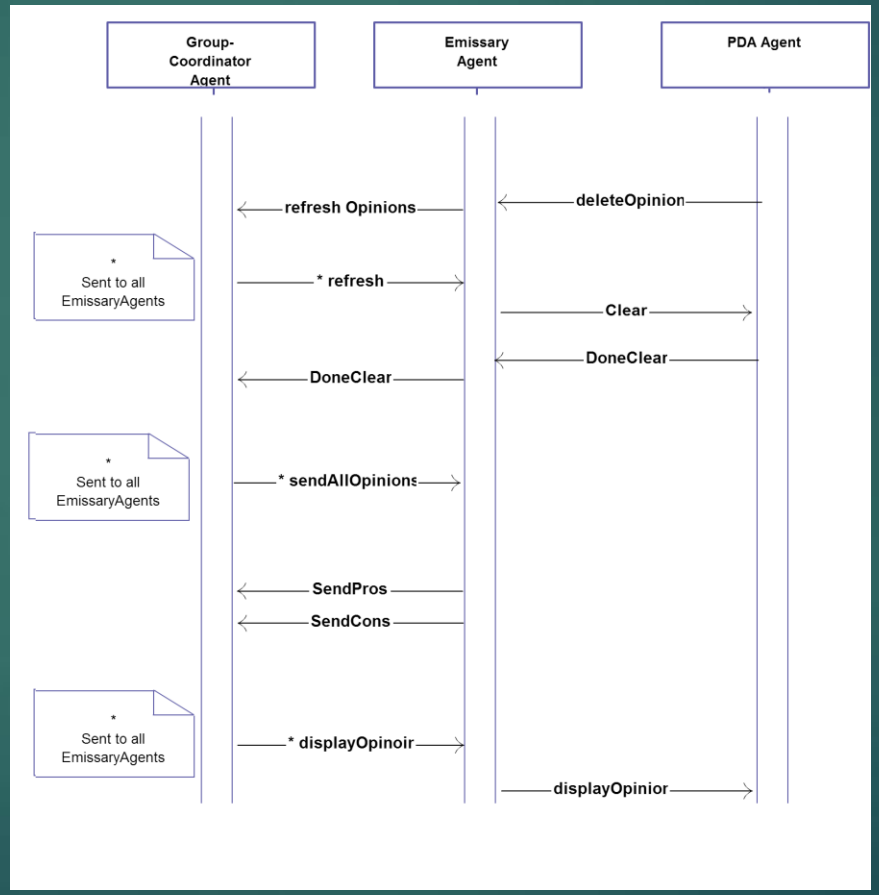
A PC/ANDROID SCENARIO (The ProCon Debate App)

- Adding an opinion



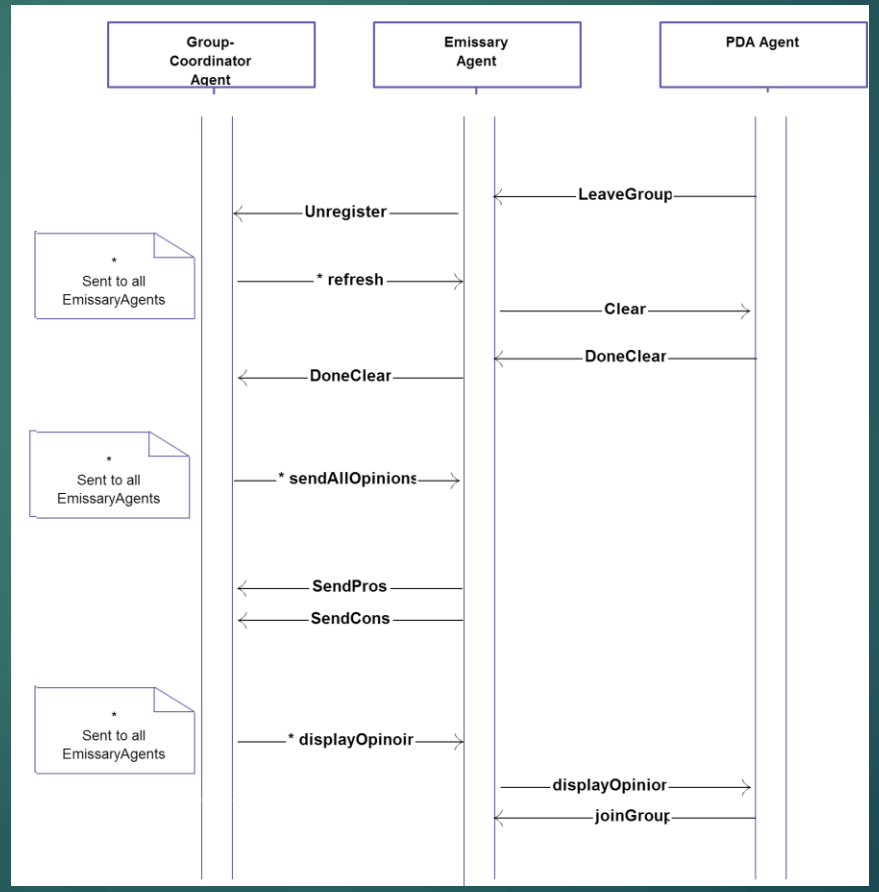
A PC/ANDROID SCENARIO (The ProCon Debate App)

- Deleting an opinion



A PC/ANDROID SCENARIO (The ProCon Debate App)

- Leaving a Group (and joining another one)



FUTURE WORK

- Since the tATAmi platform had recently been re-structured, the application must be ported fully to it making the necessary changes.
- More actions could be added to the Agents making the application richer.

- an editor that allows developers to write S-CLAIM code easily and elegantly would be a very nice addition, offering some of the following features:
- Open the specific type of Agent file (*.adf2).
- Color and suggest auto-completion for S-CLAIM keywords.
- Find the existing variables and method in the *.java/xml files in the same project and also color and suggest auto-completion for them while typing S-CLAIM code.
- A kind of a "run" command/visual-button for the file (of the specific type) to check if it follows some specific syntactic rules or not.
- Showing errors and the line numbers in which they occurred.