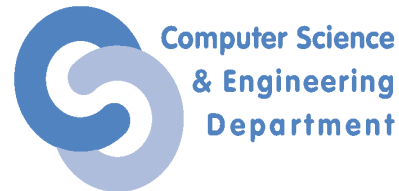


Universitatea POLITEHNICĂ din București
Facultatea de Automatică și Calculatoare,
Catedra de Calculatoare



LUCRARE DE DIPLOMĂ

Aplicație Android bazată pe agenți pentru participanții la conferințe

- implementarea agentului software folosind platforma tATAml -

Coordonator

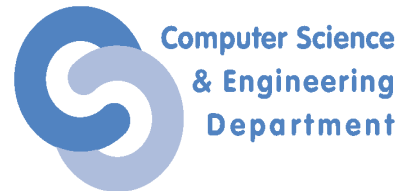
Ș.I. Dr. Ing. Andrei Olaru

Student

Ivona-Emma Sevastian

București, 2014

University POLITEHNICA of Bucharest
Automatic Control and Computers Faculty,
Computer Science and Engineering Department



BACHELOR THESIS

Agent-Based Android Application for Conference Participants

- software agent implementation using tATAmI platform -

Scientific Advisers

Andrei Olaru, PhD

Student

Ivona-Emma Sevastian

Bucharest, 2014

“Civilization advances by extending the number of important operations which we can perform without thinking about them.”

Alfred North Whitehead, Introduction to Mathematics (1911)

“If you look at history, innovation doesn't come just from giving people incentives; it comes from creating environments where their ideas can connect.”

Steven Johnson

“The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”

Mark Weiser

Abstract

In a world that is always in motion, where, to quote a well known phrase, “time means money”, we are always searching for solutions that will make our life easier. Ambient intelligence is a domain that has flourished in the past few years, to a point where our society depends on it. By passing some of the human responsibilities/ usual task to an intelligent system, we can avoid boring routines, giving the user the ability to better manage his time. Following this idea, we are going to focus our attention on improving user experience while participating in a conference, by the means of a Smart Conference application.

The fundamental feature of the application is that it makes it easy for the participants to organize their time and prioritize the events, during the conference. In order to create a viable application we should concentrate on the important parts: communication, user interaction, user action and location detection, decision making, intelligent system. Throughout this paper, we are going to focus on the implementation of the software agent using tATAmI platform. This implies integrating a new cognitive component able to react to environment changes, adapting agent’s behavior definition and execution to the new context and modeling their way of acting.

Table of Contents

Abstract

List of figures

Notions and abbreviations

1. Introduction

1.1 Ambient Intelligence

1.2 Smart Environments

1.3 Smart Conference

1.4 Project Goals

2. State of the art

2.1 Introduction

2.2 Existing Conference-Support Applications

2.3 Improvements

3. Project and Specifications

3.1 General Purpose

3.2 tATAmI Platform

3.3 Cognitive Component

3.3.1 Context Graphs

3.4 Overview of the Project

4. Application Architecture

4.1 Application Modules

4.2 Changes in tATAmI Platform

5. Implementation details

5.1 Languages

5.2 Improving tATAmI

5.2.1 Cognitive Component

5.2.2 Agent Behaviors

6. Smart Conference

6.1 Agent Model

6.1.1 Elements of the Conference Environment

6.1.2 Elements Properties

6.1.3 Example of Use Cases

6.2 Scenario Internal Representation

6.3 Case study: Smart Conference Scenario

7. Conclusions

7.1 Contribution Overview

7.2 Possible Improvements and Future Work

8. References

List of Figures

- 3.1 Simple example of CG representation
- 3.2 Simple example of CP representation
- 3.3 Server Architecture. Colored nodes represents the contribution described in this paper to the Smart Conference Application
- 4.1 Application modules - Android side
- 4.2 Application modules - Server side
- 4.3 Overview of tATAmI platform
- 6.1 Main Agent KB example

Notations and Abbreviations

| | |
|---------|--|
| Aml | - Ambient Intelligence |
| S-CLAIM | - Smart Computational Language for Autonomous, Intelligent and Mobile Agents |
| tATAmI | - towards Agent Technologies for Ambient Intelligence |
| CG | - Context Graph |
| CP | - Context Pattern |
| KB | - Knowledge Base |

1 Introduction

1.1 Ambient Intelligence

Philips was the first company that has brought into public attention the notion of Ambient Intelligence (Aml) in 1998 during a series of workshops. It relies on the presence of a digital environment that is adaptive and responsive to people. As Raffler stated: "A digital environment that proactively, but sensibly, supports people in their daily lives." [15]

Aml puts the emphasis on the user experience, on giving him better control over his daily actions. Its perspective changes the normal view on human-computer interaction, by gathering a variety of devices which are embedded in our environment. Main objectives of Aml are to improve human comfort, security, health, communication and to optimize daily tasks with minimal interaction from the user. Objectives are achieved by bringing together different resources: sensors, networks, graphic interfaces and areas of Computer Science: Artificial Intelligence, Ubiquitous Computing, in order to provide intelligent services to users in their own environment.

Aml is defined by its characteristics, for example: open, adaptive, context-aware, personalized, embedded in the environment, transparent. Systems are almost, if not totally, invisible to people, creating a pleasant, comfortable environment.

These systems should be able to exhibit specific forms of intelligence: recognize people and emotions, react based on human behavior etc. To achieve this, Aml combines different notions / areas. An Aml system should know the context (context-awareness concept) in which it is being used and be able to respond accordingly. "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". [2] This fact is stated by the "5W's and 1H principle" as well:

- who: every entity has an associated role. The system needs to distinguish between different entities in order to respond accordingly
- where: environment, physical space. Should be aware of its surroundings, in order to map/translate the information gathered from the exterior
- what: uniquely characterised objects/ events
- when: Time plays an important part, since some of the information is relevant just for a short amount of time, at certain moments. (For example: an Aml home system that knows when to wake you, when to turn the lights on or off. We wouldn't want to be woken up in the middle of the night. Of course, those examples are not relevant without the entire structure of the system, since its decision is not based only on time)
- why: Mainly, all of our actions have an end-point. (daily life could not be considered as open-ended) Following the same idea, Aml system's action tend to have a goal. Their actions are calibrated after it.
- how: actions done directly / indirectly by the system to achieve its goals

The principle of integrating different devices / resources into the environment and into our daily lives' background is one of the main conventions of Ubiquitous Computing. It is a concept that relays on the fact that "a computer is made to appear everywhere and anywhere... At their core, all models of Ubiquitous

Computing share a vision of small, inexpensive, robust networked processing devices, distributed at all scales throughout everyday life and generally turned to distinctly common-place ends.” [3] The area of Ubiquitous Computing has developed due to the advance of the smart technologies, such as wireless communication and networking, efficient sensors, machine learning and decision making, robotics, middleware and agent technologies.

Reeves and Nass argued that the interaction between human and machine should follow the example of interaction between humans: “Our strategy for learning about media was to go to the social science section of the library, find theories and experiments about human-human interaction – and then borrow...Take out a pen, cross out ‘human’ or ‘environment’ and substitute media. When we did this, all of the predictions and experiments led to the media equation: People’s responses to media are fundamentally social and natural”[9]

Due to the advance in technology, Ambient Intelligence is following an ascending path. However, as Pardeep Maheshwaree describes, “ambient use-cases are user-centric and not driven by the technology.”[4] It models on the people’s needs. It can be applied to a large number of everyday situations: from your house waking you up and providing help in small tasks, to health care usage and even self-driving cars (smart cars). Human-computer interaction has developed to a whole new level. As long as these systems will improve our lives, making us feel more secure, more comfortable, freeing us of repetitive/ long/ boring tasks, Aml will keep on advancing. New ways of improving existent systems or involving them in totally new situations will appear.

1.2 Smart Environments

As described above, a lot of use cases for Aml tend to interact with Smart Environments concept, which can be defined as the infrastructure of the system: “a small world where different kinds of smart device are continuously working to make inhabitants' lives more comfortable.” [10] It tries to replace and automate daily tasks in order to provide comfort. The first step is to gather information about the state of the environment, by the means of physical components. Then, the environment must use the information to reason about the goals and the outcome of the possible situations. At last, a change in the state of the environment may or may not be triggered.

The operations above exhibit one of its most important features: autonomy. An autonomous system is capable of making its own decisions based on prior experience. It can infer what the user wants to do, or what he wishes for, by following his actions, and can react accordingly.

“Within a home environment, ambient intelligence will improve the quality of life by creating the desired atmosphere and functionality via intelligent, personalized interconnected systems and services.” [7] But there are also other applications and environments defined for Aml, beside smart homes: smart conferences, smart hospitals, smart rooms, smart cars. In this paper our objective is to improve user experience while participating in a conference.

1.3 Smart Conference

What should a Smart Conference imply? Let's first take a look at a normal conference. It should be composed of speakers, attendees, organisers and chairs (in some cases, the one that helps with the organizing and keeps order). The environment will be represented by rooms, which can be: presentation rooms, meeting rooms and break rooms. Attendees register as they arrive and attend the presentations they prefer. During this time, they may (or may not) meet other colleagues with whom they share common interests, therefore building professional relationships.

In this scenario, there are some points that can (and will) be improved. For example, how will participant A know that at the same conference (or maybe presentation) there is participant B with whom he would enjoy having a talk? Or how would participant A know that another presentation is delayed? Wouldn't it be easier if they knew all these ahead of time?

Here is where Smart Conference concept shows its real value. For the environment to become an Aml one we need sensors (for tracking the position of the user, making it easier for him to get relevant notifications) and a cognitive component (which is the objective of this paper). Based on current events, situations, user interactions we will then be able to give suggestions to attendees. In order to achieve this, we will need a good infrastructure as the ground base of our system.

1.4 Project Goals

Our main goal is a Smart Conference application that guides the user during a conference and makes him feel more relaxed and confident, knowing that he will get the most out of this event. This application contains parts which, brought together, should take the user experience to another level. Let's first discuss about these components and then focus on the main idea of this thesis.

For the application to reach its goal it needs:

- sensors: for detecting and tracking the attendees in order to provide accurate information about the environment. For this purpose different technologies can be used: logical sensors (e.g. QR Codes, which require the user to actually check-in at Presentation/Meeting/Break Rooms)), physical sensors (e.g. iBeacons)
- means of communication between different devices and modules
- a way of representing the environment in order to interpret and react to changes
- a decision making process / component

The main idea of this thesis is represented by the implementation of the software agent. It should be able to interact with the environment, react based on attendees actions, interests, detect similarities between different users and situations and provide the most accurate and useful information to them.

I will summarize my contribution to the Smart Conference application in the next few lines, along with the final goal, which will be described in detail throughout the rest of the paper:

First of all, I have integrated a Cognitive Component based on Context Graphs (CG) in the application. I have used S-CLAIM language to define the agent's behavior. The language and the possible behaviors of the agent were adapted in order to interact with the new component. Until now, tATAmI used agents based on Jade. The novelty of our application stands in the fact that it is just partially dependent on the Jade agent.

Because we now have a way to define the agents, interacting with the environment is the next big step towards the final goal. We will discuss about how to use these elements in a Smart Conference scenario by illustrating the real world model that we've created, the behaviors and the agents.

In the next chapter we will present a quick overview of several state of the art contributions, and what can be done to improve the approaches to a normal conference. In section 4 we are going to present the application's architecture: its components and logic. Section 5 will get into more details regarding the implementation and technologies we have used. Last , but not least, is the discussion about how to model the agent in order to be integrated in a Smart Conference application and the presentation of an actual use case of the contribution (where and how we can use our implementation). An important part of the cognitive component is the way it stores the knowledge and how it interacts with it. By this, we mean how the elements of the world are translated into its internal representation (Here the world = the conference environment and we are going to represent it using Context Graphs).

2. State of the Art

2.1 Introduction

“We are rapidly going towards a world where computers become truly intertwined with our daily lives.”[8] If until not so long ago, we could distinguish between human or computer action/interaction, now the line becomes blurry. More and more computing devices are around us, helping and guiding us. As Mark Weiser said long time ago: “The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”[6]

Ambient Intelligence was created on the grounds of user centered design, concept that places the user in the center of the development, in order to create complex and intelligent environments. The main purpose of this Intelligent design is to improve the quality of life by making daily tasks easier for the user to handle and offering efficient ways to communicate and interact with other people, systems and environments.

Use of physical components (sensors) and smart devices (controllers) is indispensable for our real-world environment to exist. These are the elements that allow the agent to sense, infer about the environment and its components and select the right actions. The autonomy of a smart environment is one of its most important features, due to the fact that it is able to adapt to changes, and to the natural way of communicating with humans. Autonomy has certain benefits that influence the user, by offering him comfort and better control over his daily routines. One example is that the user is able to use the environment's devices remotely. The appliances used within these smart environments provide remarkable capabilities: a large quantity of information is shared between devices in order to create a complex image of the state of the environment, and better customize its behaviors.

Smart Environments have been gaining a lot of popularity in the past few years. Also, Conferences tend to be at a larger scale. The obvious step was the movement toward Smart Conferences applications.

2.2 Existing Conference-Support Applications

Amiando / Conference2Go

Amiando [11] is available for Apple devices (iPhone, iPod touch and iPad). It is designed as a multifunctional and professional mobile application, an approach for online registration.

The developer of Conference2Go describes it as being able to give “attendees easy mobile access to all the relevant information surrounding an event. In addition, this App offers event organizers numerous opportunities to better portray conferences, as well as their exhibitors and sponsors.” Its main goal is to provide an efficient way to present the event in order to provide the conference attendees with quick access to everything they need.

Features:

- communicate with attendees before, during and after the conference
 - helps in promoting the event in advance
 - update delegates during the show
 - do post-conference follow ups

- the attendees have live access to the organizers' posts
 - ideal for speakers, exhibitors
- easy and direct access to the schedule of the event
 - can easily be integrated into users personal calendar
- mobile networking
 - incorporates messaging platform and an attendee search tool
- social medias (ex: Twitter and Facebook) are incorporated into the app.

MySmartEvent

MySmartEvent [12] is an useful application, mainly designed for conferences and business events, providing an interactive user experience for participants.

Its developer describes an overview of the application development: "We know what running an event, business conference or seminar is like. We've run hundreds. The last thing you need is more work. So we've made sure setting up your own mobile app is an easy process" .

Main goal: "Keeping attendees interested, informed and engaged"

Features:

- the user can easily visualize details of the event, and information about his partners
- participants can follow the news feed of the event
- view profiles and interests of other attendees
- photos uploaded by the event organizer can be viewed during or after the event
- the public can be engaged into a voting procedure
- schedules personal meetings
- attendees have the possibility to download and send files

2.3 Improvements

Most of the conference application platforms are not developed to the degree of having their own intelligence and autonomy. They usually take advantage of the attendees' smart devices. Further we will talk about Smart Conference, concept that implies bringing the conference to another level. In order to achieve this state we are taking advantage of numerous resources and notions. Those will be physical (e.g. sensors used for tracking attendees during the conference, information which will be sent to the agent), and logical (examples of some of the concepts used: Multi-Agent Systems, Context Graphs etc). By combining these (and other resources) we will get a better understanding of the environment, thus achieving a state where we can provide useful, accurate suggestions.

As its name states, the concept of Multi-Agent Systems refers to multiple intelligent agents and their environment. For the purpose of this thesis, the agents will be actually represented by their cognitive component, while the environment will refer to the conference environment. The agents will be able to interact, provide useful information to one another. Also, they will be able to process the information , generate new knowledge and react accordingly.

3. Project and specifications

3.1 General Purpose

The subject requires the research and implementation of an agent-based application that assists the participants at a conference in their activity. The application will help participants with finding researchers who share common interests, with organizing meetings and with presenting their work to the audience, by detecting relevant situations and acting appropriately. It will be based on the S-CLAIM agent programming language and the tATAmI agent-based platform.

We will focus on the implementation of the software agent, while following the next steps: integrating a new cognitive component based on Context Graphs into tATAmI platform, adapting the language and the possible behaviors of the agent in order to interact with the new component, incorporating elements that will not depend on Jade in order to achieve a new, independent version of tATAmI, modelling the agents behavior, representation, etc. We will further develop this topic in the next chapters.

3.2 tATAmI Platform

The implementation relies on the tATAmI agent-based platform and on the S-CLAIM agent programming language. Those two combined, form an unified framework capable of implementing deployable, multi-agent systems. From its beginning, tATAmI was designed to have three main components: Agent, Simulation and Visualization. The Agent (which is the central component) was based on the Jade agent. For a more general approach, it developed to a newer version where an agent has independent components: Messaging, Movement.

The integration of S-CLAIM into the platform is an important aspect, as it describes the agent. It provides a simple way of programming them by introducing agent definition files (.adf2 files). S-CLAIM has derived from CLAIM language and tries to go beyond its limitations[16]. It was designed to have a basic syntax, easy to use and understand. Simple and concise, it can be used in a wide range of applications.

“The ambient intelligent applications are well known for their architectures, for the way they combine different components and most of all for their frequent changes in context.” [16] The context refers to the environment, but also to the agent’s behavior, and to any information that characterizes an entity (person/place) and/or its situation. So, an important part of the agent would be to be sensitive to the environment and to other agents. Following this idea, context-sensitivity seems a key element. It will allow to adapt, react / interact with other users/agents and/or situations it may encounter. Thus, we need a component which could structure and combine all of the above.

3.3 Cognitive Component

tATAmI platform included a cognitive component that was using a simple knowledge base (KB) on top of the Jade agent. We propose a more general approach using Context Graphs for the implementation of the knowledge base. The new representation is more concise, easy to interact with and provides other functionalities that we will find useful while implementing the cognitive agent: it is capable of finding similarities between users / situations, it is fast in retrieving the needed information etc.

3.3.1 Context Graphs

As previously discussed, an Aml system should be aware of the current context, in order to respond to different changes, situations, interactions. On the grounds of this statement, context-awareness is a central element in an Aml system. The concept originated in the field of Ubiquitous Computing while dealing with the relationship between environment changes and particular systems and devices. It refers to the observation of the context (which can be made through sensors, different interactions etc), its representation and reaction to external factors.

Andrei Olaru proposed a graph approach to storing context information that can be used in a multi-agent system, allowing agents to process that information by using context matching. CG provides a way of modeling relationships between different entities (places, people, objects etc) in a simple, intuitive manner. The concept is more complex, “making possible the isolation of application-independent context-related processes in a specific layer – a middle- ware that relies on a simple, flexible and generic context representation in order to perform tasks like situation detection, decision, and sharing of context information” [13] The graph will store relevant information for the current user / agent. Further let’s define CG and its components / usage while giving examples.

The formal representation of a CG is defined as:

$CG = (V, E)$, $V \subseteq \text{Concepts}$ (Strings or URIs, refers to particular entities)

$E = \{\text{edge} \mid \text{edge} = (\text{from}, \text{to}, \text{value}), \text{from}, \text{to} \in V, \text{value} = \text{relationship between those 2 concepts, the label of the edge}\}$

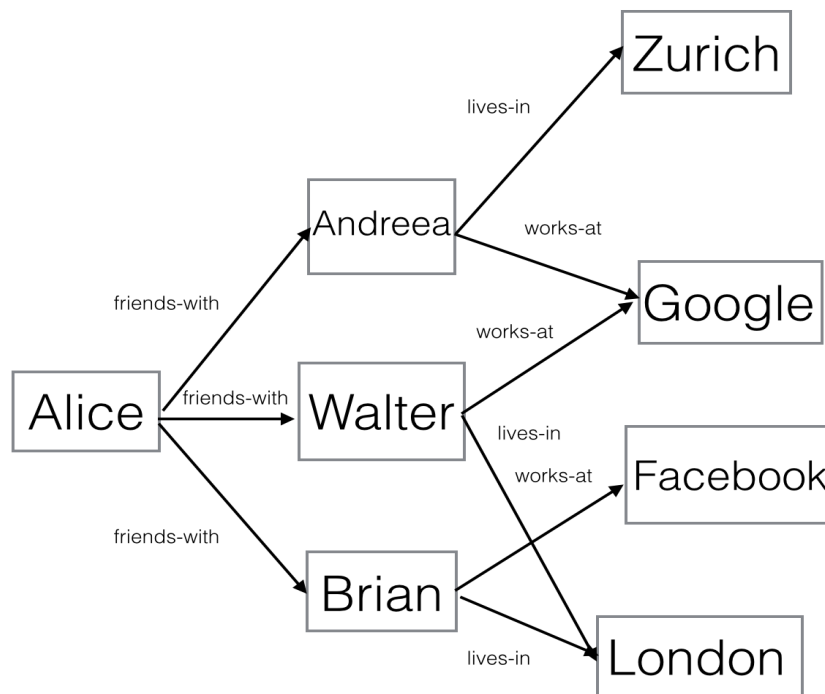


Fig 3.1 Simple example of CG representation

As seen in the figure 3.1, we are able to easily represent a real-life situation and understand it: we have 4 persons. Regarding Alice, we have information about her friends, while regarding Andreea, Walter and Brian, we know where they work and live. Based on the current situation, we would like to know if (for example) Alice can visit London and the Google Office with one of her friends. Context Patterns (CP) are used in order to detect particular situations: “A pattern represents a set of associations that are specified by the user, the applications, or are extracted by the agent from the history of context information.” [13]

The representation of the pattern is similar to the CG, but besides nodes, labeled edges, it may contain generic components: “?” labeled nodes or actionable edges. Beside these characteristics, the pattern contains a relevance element (how relevant the match will be for the current situation) and a persistence element (how long will it be available after the match is done).

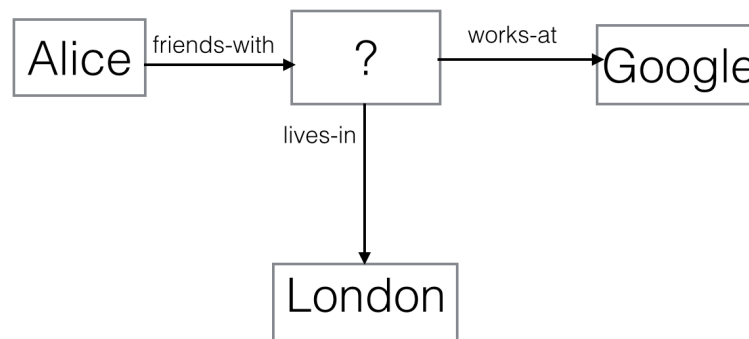


Fig 3.2 Simple example of CP representation

This pattern can be used for detecting if our question has a positive response. As we can see, the CP contains a pattern node ? which can match against any of the nodes, but related to the fact that the labeled edges point to actual concepts, it will match to the desired friend.

A formal definition of the match contains: (full match, solved part and unsolved part)

For a match to exist there are some conditions that must be met: all components of the pattern should match a different component in the CG with the same value if it is a non-generic element, or should respect the regular expression if it is a generic edge. CG are more complex, thus being a good fit for our goal. We are not getting into further details here, since we are going to describe our representation in more details.

3.4 Overview of the Project

Main requirements of the Cognitive Component are:

- to provide persistent information
- to facilitate the interaction with the knowledge base
- to detect relevant situations

Since S-CLAIM allows various representations of the KB (as long as they can be addressed by relations or association patterns), Context Graphs seem to be a good approach.

Beside the resources described above, our application has other components that can be visualized in figure 3.3 which will be described in the next chapter. As we can see, the green colored components are the ones we are putting emphasis on in this paper.

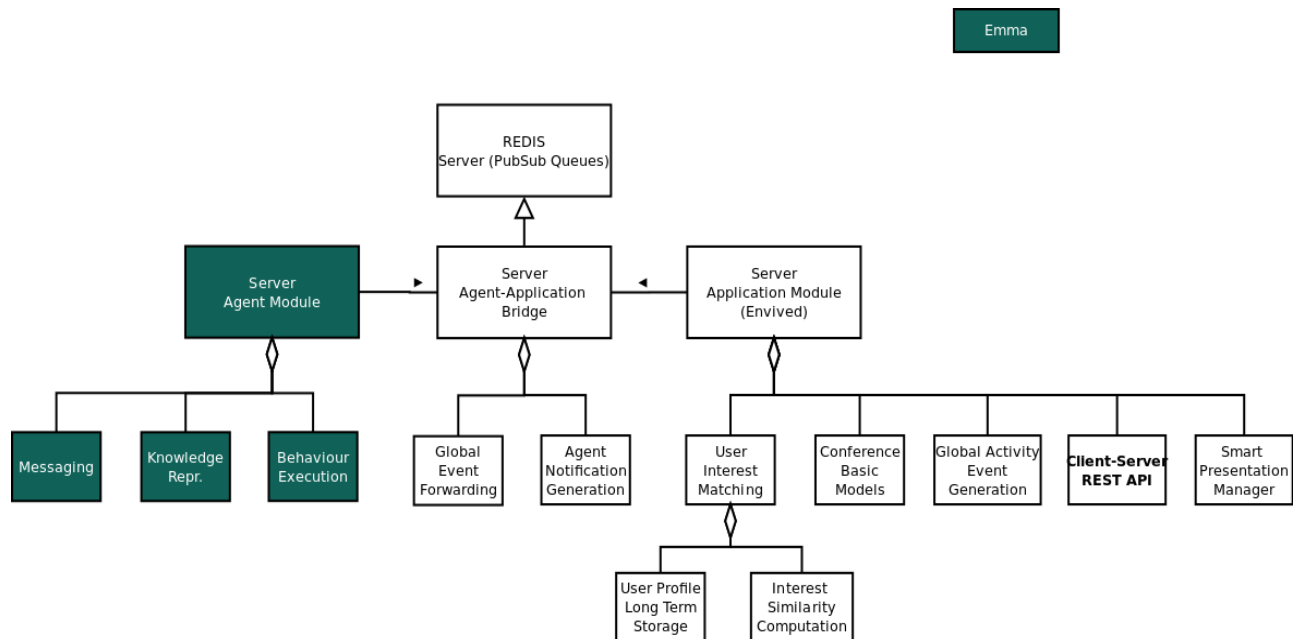


Fig 3.3 Server Architecture. Colored nodes represents the contribution described in this paper to the Smart Conference Application

Knowledge Component/ Representation

- implemented using Context Graphs

Behavior execution

- Execution of S-CLAIM and other behaviors
- Behavioral triggers (message receipt, other behaviors, context matching)

Messaging

- implemented by the means of S-CLAIM

4. Application Architecture

4.1 Application Modules

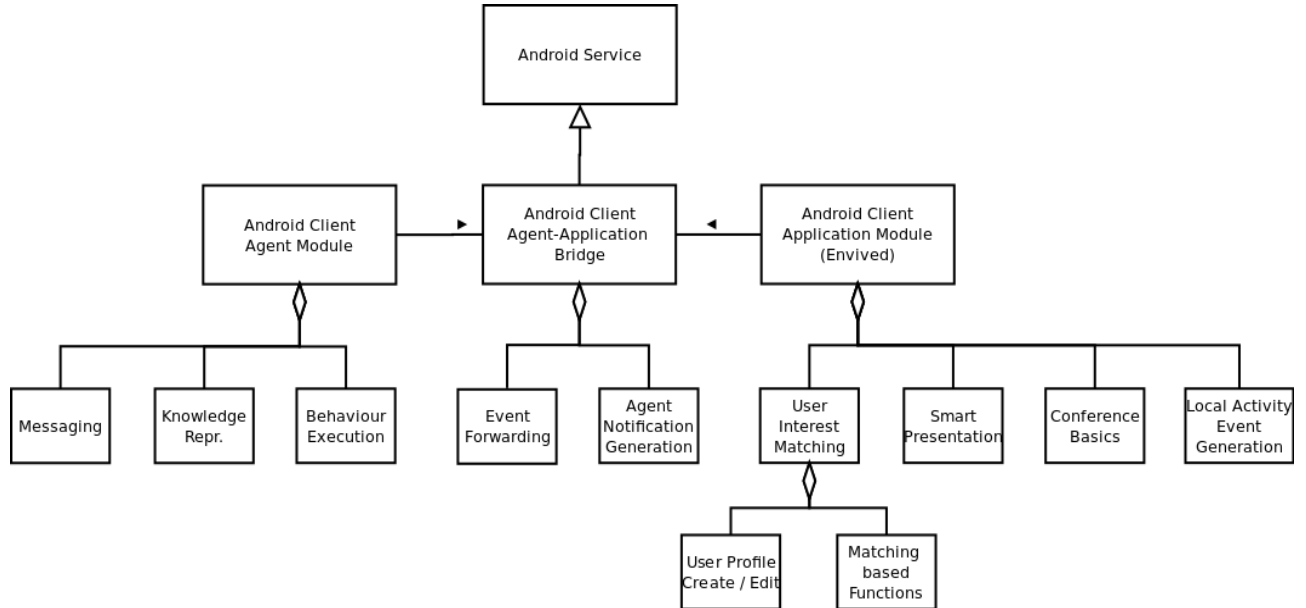


Fig. 4.1 Application modules - Android side

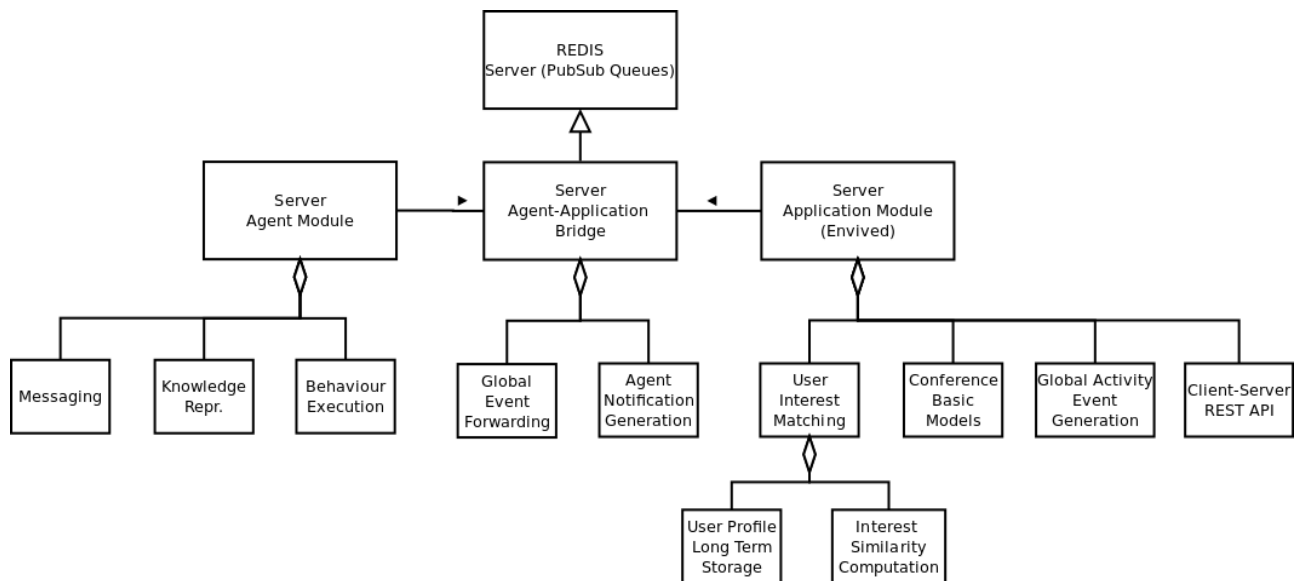


Fig. 4.2 Application modules - Server side

As seen in the fig 1 and 2, the application consists of a server side and an Android Client side. Module components are:

- Agent Module:
 - similar on the client and the server part
 - on the server side, we are going to keep complete relevant context, while on the client side, we will only retain relevant information for the user. The server is the one synchronizing the clients with the new information, using Agent-Application Bridge.
- Agent-Application Bridge: facilitates the communication between the agent and the application
- Application Module: for the client, this will represent the application that users will interact with

4.2 Changes in tATAmI Platform

As previously discussed, the Agent Module is implemented over tATAmI. The structure illustrates the relationship between its units:

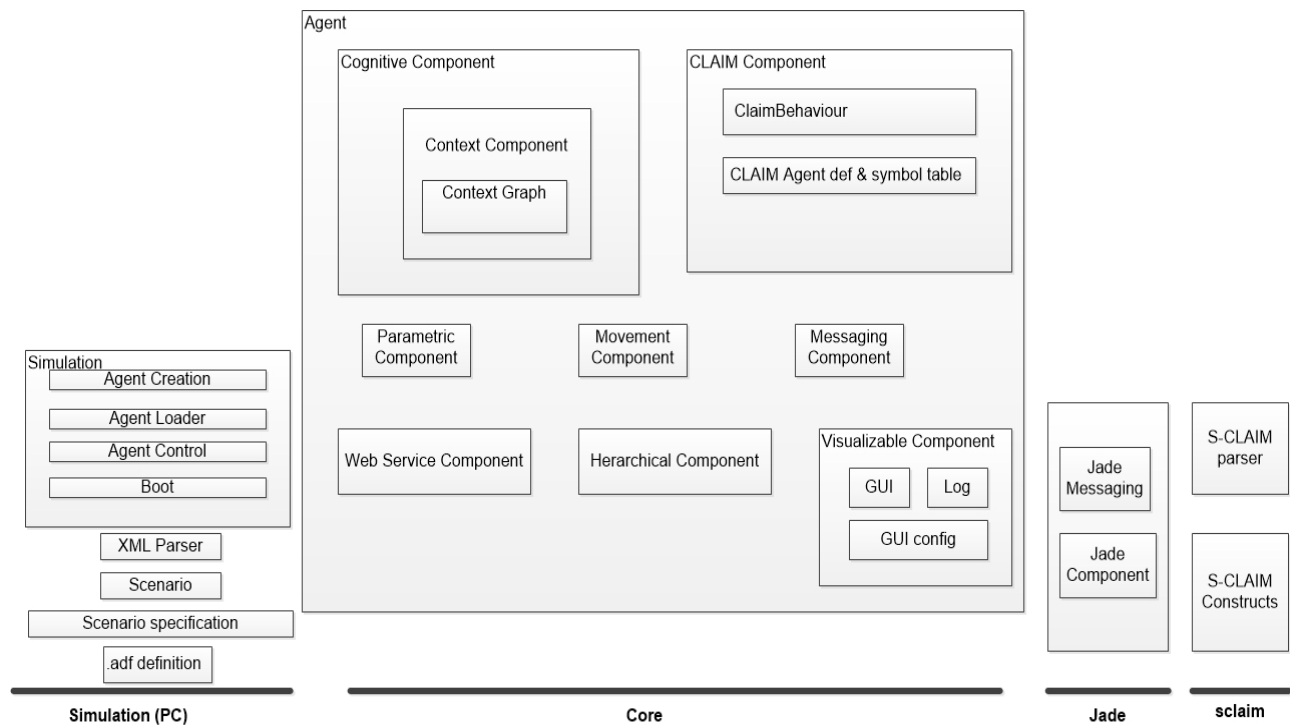


Fig. 4.3 Overview of tATAmI platform

As seen in figure 3, the agent will consist of several elements (all extending AgentComponent):

- Hierarchical Component
- Messaging
- Movement
- Parametric
- Visualization
- WebService
- Cognitive (the main component we are focussing on during this paper)
- Claim

During the implementation, parts of the tATAmI architecture changed.

The new concept moves towards the displacement of Jade, although not all the elements are currently implemented this way.

Despite the fact that the internal representation of the behaviors (which depends on the S-CLAIM definition) hasn't been significantly changed, S-CLAIM agent definition has been modified in order to adapt to the new representation of the knowledge base.

The old version of the cognitive component was based on a simple knowledge representation that allowed the information to be stored in a human readable format, independent one from another. By introducing CG representation, we are adding some new functionalities (eg: interconnected components).

If until now, the knowledge was handled as usual text, now we had to handle things quite differently. Let's look at a simple example invoking readK statement. Based on a pattern, it should return the first available match. In the older version (e.g.: `(readK (struct knowledge ?Ana goes ?Where)))`) we had to search through all the knowledge parts of the KB while patching word by word. After the cognitive component has changed (e.g.: `(readK (struct knowledge pattern ?Ana -goes-> ?Where)))`) we have to search in the CG for a possible match. This is done internally, after converting it to a Context Pattern (eg.: `"?#1 -goes-> ?#2"`)

Due to CG use and partial removal of Jade, we are updating the method of executing behaviors which we are going to discuss in the next chapter.

5. Implementation Details

5.1 Languages

As discussed above, our implementation is based on the tATAmI agent-platform which combines different technologies and languages. It is mainly written in Java and was based on Jade.

It uses an XML scenario format which contains the definition for all the agents and a short, parametric description of all their elements. In this manner, we can specify the components of the agent and add relevant parameters to them: add the initial KB of the cognitive component and provide the path for the agent behavior definition.

Agent behaviors are modeled using S-CLAIM language, which provides an easy method for describing agents actions.

5.2 Improving tATAmI

tATAmI was designed as a unified framework for implementing deployable multi-agent systems. It can be used in a wide range of scenarios, providing different capabilities for the agents.

5.2.1 Cognitive Component

The main part of the Cognitive Component is represented by the knowledge base. There are different ways it can be represented. We have chosen the CG representation because it provides additional features of significant importance to the project.

The implementation of every unit has to extend AgentComponent. The structure of the component is represented by the CognitiveComponent class, which contains common elements independent of the actual representation, and the ContextComponent class, which extends CognitiveComponent.

The CognitiveComponent offers different methods of using / interacting with the KB. We are able to add, remove information, but also to reach one / all match(es) of a pattern. While adding a new cognitive component, we are able to specify the starting KB through the scenario definition (.xml file). It will be added as a parameter (text representation).

5.2.2 Agent Behaviors

To ease the communication between the agents within the environment, we should be able to execute their behaviors. We are keeping their definition in .adf2 files, written in S-CLAIM programming language. In order to execute that information, we firstly use a parser that will translate S-CLAIM format into an internal mapping of the definition. Secondly, every agent will execute its behavior by following the steps stated in the files. Below are described the possible behaviors and how we have implemented them.

S-CLAIM provides four types of behaviors: initial (those that happen once, at the initialization of the cognitive component), reactive (as a result of an action), proactive (goal-driven) and cyclic. The reactive behaviors can be triggered by the following actions: receiving a message or fulfilling a condition. Every

behavior definition contains a set of statements that will be executed in that order. If one statement does not execute completely, we are going to repeat it until it succeeds.

We will discuss below the implementation of some behavior executions.

The agent definition refers to the interaction with other agents, or with its knowledge base. Different types of construction are available in S-CLAIM:

- function calls:
 - wait
 - java method calls
 - condition
 - input-output related
 - input
 - output
- interaction with the KB:
 - addK
 - readK
 - removeK
 - forAllK
- agent communication
 - send
 - recv
- other statements:
 - if
 - while

We will focus the discussion on the interaction with the knowledge base, since all the statements are based upon it. The knowledge can come from the exterior (information gathered by the sensors), from other agents, through communication, or can be created internally as a result of a decision making process. No matter how the information gets to the agent, it needs to be stored in the KB.

E.g.: `(addK (struct knowledge ?Presentation $\xrightarrow{\text{has-started}}$ STARTED)).`

The structure of the message will use CG notations. Variables using ?<name> will expand to their value, if it was previously stored internally. All the interaction with the KB is done through cognitive component. In the same manner, we can remove parts of the KB.

CG provides an easy way to determine relevant situations, by utilising patterns. Their format is almost the same as the graph representation, with the mention that it may contain non-generic components (NodePattern or EdgePattern). The pattern should describe a situation which we want to know if it exists, and should contain pattern components for the elements that we would like to find out. We are using patterns in the implementation / execution of different statement: readK, forAllK, condition.

readK has almost the same format as addK. It will return the first match found, if such one exists.

`(readK (struct knowledge pattern ?User $\xrightarrow{\text{attends}}$ PresentationName))` (If a match is found, the later use of ?User and ?RoomName will expand to the returned information, since we are internally mapping variables to the new found values. (by using symbol tables)

forAllK, is similar to readK, in the sense that it searches in the current users' KB, but in addition, it returns all possible matches. An example of usage would be the scenario of searching through all the attendees of a presentation that are in the presentation room, in order to mute their phone.

```
(forAllK (struct knowledge (?User attends → PresentationName)
  (if (readK ?User is-in → RoomName)
    (muteTelephone)
  )
)
```

The new values of the “?” variables, those that will be initialized with each match, will not be preserved after the execution of forAllK, since they are relevant only for the current function (local scope)

The agents will be able to interact with each other, through send and receive statements.

Below is illustrated the structure of these communication statements, through a simple example linked to an user attending a presentation.

```
(send (struct message UserName attends → PresentationName))
```

We are using this message format in order to be easily interpreted. When receiving this notice, the main agent will retain the information regarding the user. Receive statement will also keep the user agent identifier.

```
(recv ?agentName (message ?UserName attends → ?PresentationName))
```

simple example of using condition statement:

```
(condition (readK (struct knowledge pattern ?Presentation has-started → STARTED))
```

Java calls: We can use Java methods beside the functions that S-CLAIM provides. To achieve that, we are integrating Java code sources (containing the actual method) that will be defined in the xml scenario as “java-code” node. The path to the source code is retained in the “agent-package” element.

6. Smart Conference

In this chapter we are going to talk about how we can use the previously described improvements to integrate our application into a conference event, in order to add a “smart” layer to it. For this purpose we will first concentrate on how to model the environment and the actions /the reactions, in order to apply to this case and to be able to interact with the KB. We will then take a look at how the agent will be defined (using XML files) and at some possible use cases.

6.1 Agent Model

We can distinguish two different types of agents: user / attendee and main agent

- User Agent:
 - refers to the user / conference attendee
 - is uniquely identified
 - its knowledge base will contain relevant information for the current user
 - what he will attend
 - presentations
 - interests
 - others
- Main Agent :
 - refers to the main server
 - the knowledge base:
 - will contain all the relevant information regarding the environment and the users
 - will be used for matching, sending notifications, etc.

tATAmI platform supports Multi-Agent Systems which are able to communicate. The main agent will run on the server, while the user agent will run on each of the attendees mobile device.

6.1.1 Elements of the Conference Environment

For the purpose of translating the conference reality into an understanding representation for the cognitive component, we should first take into account the important elements:

- Space represents the actual conference location which can be divided into:
 - SessionRoom
 - MeetingRoom
 - CoffeeBreakRoom
- Presentation
- User / Attendee
- Session
- Notification

6.1.2 Element Properties (modeled using CG notations):

CG has the ability to represent by labeled edges the relationships between different entities/ concepts. Space is an extensive concept, that refers to the physical location of the conference.

- Space (while referring to space by the point of view of a conference. The actual representation will be through the room names)
 - isa → Room
 - isa → SessionRoom /
MeetingRoom /
CoffeeBreakRoom

A presentation will take account of the Space, Users, time in order to answer the questions: who, what, where, when. Short description:

Presentation (identified by the name of the Presentation)

- isa → Presentation
- speaker → UserName
- slides-URI → PresentationSlidesURI
- part-of → Session
- location → RoomName (may be unnecessary if we have the session)
- is-in → RoomName(while the presentation is taking place)
- start-time → startTime
- end-time → endTime
- has-state → STARTED (after the chair session checked it) /
FUTURE /
FINISHED /
STANDBY (is next presentation, in the current session)

Example of message send through Agent-Bridge

```
content : [{  
  "subject":<presentation_id>,  
  "edge": "isa",  
  "object": "Presentation"  
}]  
...
```

A Session implies a series of Presentations that usually take place in the same room.

Session (identified by the name of the Session)

- isa → Session
- chair → UserName
- presentation → PresentationName (multiple)
- location → RoomName
- is-in → RoomName (while the presentation is taking place)
- start-time → StartTime
- end-time → EndTime

The most important concept in our representation would be the User. He will be identified by the name/unique ID. We should have a way of storing all the users' actions / information.



User

- static information that might be relevant for future matching
 - interest → (to Domain/Field) (multiple)
 - affiliation → (potentially multiple)
 - research-field → (to Domain/Field, subset of interests)
- other
 - presenter-of → PresentationName (static)
 - chair-of → SessionName (static)
 - attends → PresentationName (the user is in the room where the Pres takes place)
 - attends → SessionName (see above)
 - to-attend → PresentationName/SessionName ("the user intends to attend")
 - is-in → RoomName
 - in-area → Speaker/ Audience

Notification - requires the system to send a notification

- to → UserName or SessionName or RoomName or values
- name → NotificationName
- value → Text

Setting

-  SettingName
-  Value

6.1.3 Examples of Use Cases:

The cognitive component should be able to respond to the next questions:

- When should the presentation slides be prompted?
- What should be done when the chair session prolongs the presentation?
- How will the attendee be notified about the starting of the presentation?
- other

Some examples of Patterns / S-CLAIM code associated with the above examples

We are going to discuss in further detail the whole scenario in the next session. Here, we are giving some examples of how the agent should react to different stimuli/stimulus.

When should the presentation slides be prompted? Since there will be a chair for the current session, he will mark the presentation as started after the presenter is in front of the audience / finished his introduction. At that point, the slides should be prompted (the speaker had to previously upload the presentation). The chair application agent module (represented by one of the UserAgents) will notify the MainAgent about this change.

Beside prompting the slides, when receiving information from the chair, the Main Agent will send the URI to all the attendees of the current presentation. (This will be useful in case they want to add notes, suggestions, questions)

```
[presentationOn: ?presentationName
  (-speaker-> ?UserName
    (-is-in -> ?RoomName)
    (-in-area -> SpeakerArea)
  )
  (-status/!2-> STARTED) // /! = actionable
  (-is-in -> *?RoomName -isa -> SessionRoom)
  (-part-of-> ?SessionName -chair-> ?ChairName)
  -slides-URI -> *?PresentationSlidesURI
]
```

MainAgent:

```
(reactive presentationStarted
  // by the previous pattern
  (condition (readK (struct knowledge pattern (text representation
of previous pattern)))
    // or
    // receive message regarding this change
    (forallK (?user1 -is-in -> ?roomName)
      (addK (?user1 -attends -> ?presentationName))
      (if (readK (?presentationName -slides_uri -> ?URL)
        (send ?user1 (message attendingPresentation ?URL))
      else
        (send ?user1 (message attendingPresentation)))
    )
  )
  (forallK (?user1 -to-attend-> ?presentationName)
    (send ?user1 (message presentationStarted))
  )
)
```

What behavior will be triggered when the chair session prolongs the session? The MainAgent should update its information regarding the start and the end time of all the current and following presentations from the same session.

Example of simple MainAgent knowledge base represented as Context Graph

we are using KB as root node in order to have a connex component.

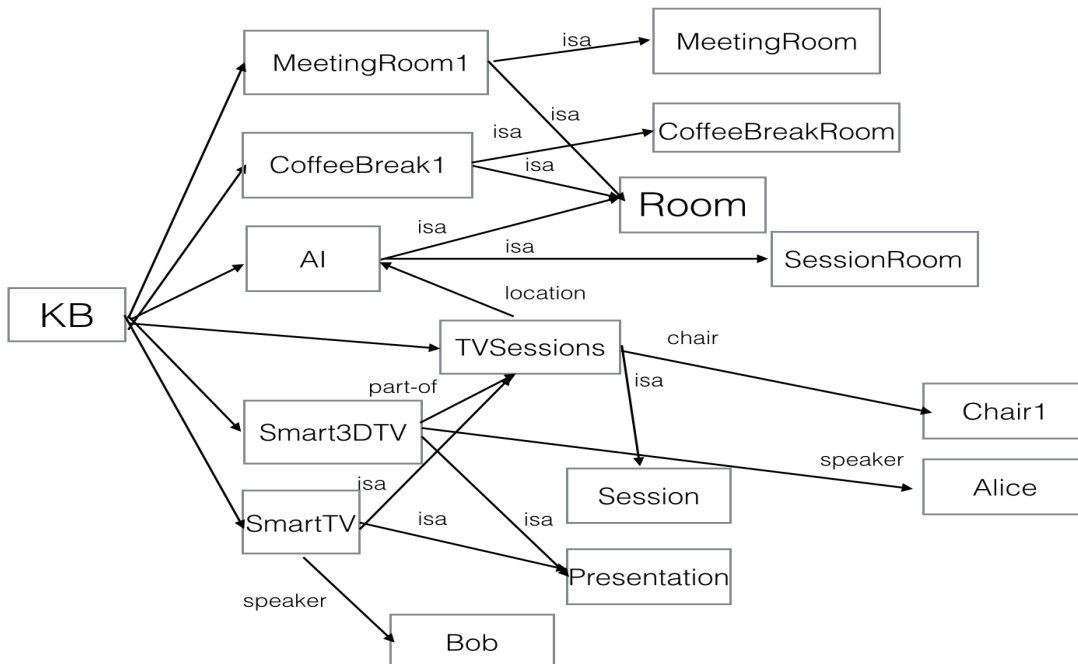


Fig. 6.2 Main Agent KB example

6.2 Scenario Internal Representation

tATAmI uses scenarios as a ground base for illustrating the components, such as agents, their parameters, values etc. Xml format was chosen as the appropriate one for implementing the scenarios, because of its easy-to-read and write functionalities.

We will now take a look at some examples of xml nodes that will prove their usefulness in creating our scenario:

For describing a scenario we will need a scenario element, which is the main node (the root node). This will contain all the relevant information related to the current described scenario.

- `<element name="scenario">`

An important element is the platform that the agents are based on. We are usually using Jade, but tATAmI could accept other types of platforms

- `<element name="platform" maxOccurs="unbounded" minOccurs="0">`
- `<scen:parameter name="name" value="jade" />`

In order to describe the agent we will use an agent element, that will contain all the relevant information linked to it.

- `<element name="agent" maxOccurs="unbounded" minOccurs="0">`

The essential part of the agent is represented by its components, which can be: messaging, movement, hierarchical, and others.

- `<element name="component" minOccurs="0" maxOccurs="unbounded">`
 - `<scen:component name="cognitive">`

A component is characterised by its properties, cited as parameters:

- `<element name="parameter" minOccurs="0" maxOccurs="unbounded">`
 - `<scen:parameter name="java-code" value="Agent" />`

Here is an explanation of the above structure: for the cognitive component, one parameter will be the initial knowledge base, as for the visualizable component, one parameter will be the GUI referring to the source of the gui interface etc

Since the agent can be seen as a Composite Agent, the loader element will be:

- `<element name="loader" maxOccurs="unbounded" minOccurs="0">`
 - `<scen:parameter name="loader" value="composite" />`

Other relevant characteristics for an agent are:

- `<element name="agentPackage">`
 - contains the S-CLAIM agent definitions and Java code attachments (which we will further discuss)
- `<element name="name">`
- `<element name="java-code">`
 - refers to the java code source (will discuss later)
- `<element name="parent">`

The cognitive component will have the starting knowledge base that contains static information about the conference environment, such as:

- presentation name, place, date
- presenters
- space related information (presentation rooms, meeting rooms)

6.3 Case Study

This chapter exhibits a possible conference scenario and points out where and how we are using the new approach, focusing on how the agent will react to different actions and how the design will improve user experience.

Preparation

Before the beginning of the conference, all the information about the event is made available to the attendees. Since the MainAgent should contain all/most of the environment elements, these will be added to its knowledge base.

Methods of adding this information:

- graphic interface
- user registration
- initialization of the agent (by adding the information in the scenario xml)

Register

Alice, one of the participants at the conference, gets to the reception desk. Here, she is asked if she would like to register and install this new Smart Conference Application. She agrees and completes the registration form (which contains basic information about her: interests, affiliations, the role at the conference). Based on the information she provided, a personal profile will be created and relevant information will be added to the user knowledge base.

For every new attendee, a new User Agent will be created, containing relevant, unique information (role, interests etc). If she is a presenter, the application will ask her if she wants to provide the presentation URL. In this case, the application will manage setting up her slides when she's due to present.

Another participant, Radu, installs the application. He registers as a chair session. This will give him additional functionalities, allowing to manage sessions.

Ad-hoc Meeting Notifications

Using the research interest based similarity component, the application will notice that Alice, Radu, Bob and George have high match. It will propose an ad-hoc meeting (in one of the meeting rooms)

Alice's Presentation

Before / At the start of a presentation, the users will get a notification, if they are not in the presentation room.

After the introduction, the chair (Radu) will mark her presentation as started. This will change the status of the presentation (in the main agent KB)

Since Alice already uploaded her presentation, the slides will be automatically prompted. For all the attendees, their phone will be muted (if they accepted this) and will get the link to possibly download or see the current presentation.

By the end of the presentation, if there are more questions/ it is taking longer than expected, the chair can prolong the presentation by X minutes. This will change the starting and ending time for all the other presentations that will occur in the same room. At the same time, the attendees of those presentations will be notified.

7. Conclusions

The software / cognitive agent should improve the overall conference experience, to a point where the user can rely on the application. Although there are enhancements that can be applied to the tATAmI platform in order to make it even more stable, the new approach has set the ground for future developments .

7.1 Contribution Overview

The goal described in the introduction of this paper has been achieved. The new cognitive component is able to process information gathered from the environment. Combined with the new design of the behavior execution, it can react to environment changes and send useful suggestion to the user. The agent prototype (as described in the last section) can be adapted to any conference.

7.2 Possible Improvements and Future Work

An important improvement of the Cognitive Component would be the use of interest-based similarities and suggestions, which would certainly improve the “smart” aspect. It uses the personal profile (provided by the user) and sensor collected data in order to provide appropriate suggestions to the user.

Any conference may be an opportunity for the participants to meet new people, form groups that share same ideas and vision, and finally develop professional relationships. Following this idea, the application might become the starting point of successful collaborations.

Also, the integration of physical sensors (eg.: iBeacons), would bring an improvement, since attendees would no longer need to personally check-in, but they can be automatically checked-in based on their location.

The application can also offer a short overview of each of the presentations and speakers, based on multiple factors: the number of participants, the responsiveness, the feedback received, number of questions asked etc. This way, the application will also create a profile of the speaker and of the approached subjects.

8. References

- [1] Philips Research – Technologies, “The technologies of Ambient Intelligence”, <http://www.research.philips.com/technologies/projects/ami/breakthroughs.html>, Sept. 2014.
- [2] G. D. Abowd, D. Anind, P. J. Brown, N. Davies, M. Smith, and P. Steggles, “Towards a better understanding of context and context-awareness,” *Handheld and Ubiquitous Computing*, Springer Berlin Heidelberg (1999), 304-307.
- [3] Wikipedia: The Free Encyclopedia, 7 Sept. 2014., http://en.wikipedia.org/wiki/Ubiquitous_computing, Sept. 2014.
- [4] Pardeep Maheshwaree, “The Future World of Ambient Intelligent Services - Mobile-phone-centric Perspective”, Helsinki University of Technology
- [5] Dey, Anind K., Gregory D. Abowd, and Daniel Salber. "A context-based infrastructure for smart environments." *Managing Interactions in Smart Environments*. Springer London, 2000. 114-128.
- [6] Weiser, Mark. "The computer for the 21st century." *Scientific american* 265.3 (1991): 94-104.
- [7] Philips Research – Technologies, “Ambient Intelligence”, <http://www.research.philips.com/technologies/projects/ami/vision.html>, Sept. 2014.
- [8] Augusto, Juan Carlos. "Past, present and future of ambient intelligence and smart environments." *Agents and Artificial Intelligence*. Springer Berlin Heidelberg, 2010. 3-15.
- [9] Reeves, Byron, and Clifford Nass. “How people treat computers, television, and new media like real people and places.” CSLI Publications and Cambridge university press, 1996.
- [10] Cook, Diane, and Sajal Das. *Smart environments: technology, protocols and applications*. Vol. 43. John Wiley & Sons, 2004.
- [11] <http://www.bedouk.com/news/conference2go-app-is-customizable-for-your-event.108640>, Sept. 2014.
- [12] <http://www.mysmarteventapp.com>, Sept. 2014.
- [13] Andrei Olaru, “Context-Awareness in Multi-Agent Systems for Ambient Intelligence”
- [14] Olaru, Andrei, and Adina Magda Florea. "A graph-based approach to context matching." *Scalable Computing: Practice and Experience* 11.4 (2010): 393-399.
- [15] J. Augusto, “Ambient Intelligence: The confluence of pervasive computing and artificial intelligence” in *Intelligent Computing Everywhere*, A. Schuster, Ed. Spring, 2007, 213-234
- [16] Baljak, V., Benea, M.T., Seghrouchni, A.E.F., Herpson, C., Honiden, S., Nguyen, T.T.N., Olaru, A., Shimizu, R., Tei, K., Toriumi, S., “S-claim: An agent-based programming language for ami, a smart-room case study”, *Procedia Computer Science* 10, (2012), 30 – 37

