

# Web 2.0 in SIP Multimedia Communications (IMS)

Tânia de Almeida Serrano

Instituto Superior Técnico - Taguspark  
Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal

E-mail: tania.serrano@tagus.ist.utl.pt

## Abstract

*The main objective of this work is to add a set of new features into a conference and collaboration service on the Internet. This service is in development at PT Inovação and is named Tagarela. The new features allows that, once inside a conference room, the users may share their desktop (desktop sharing), or a set of pre-added or selected images (slideshow sharing), having always a voice channel available.*

*In the scope of this work it was also developed an analysis of some existing webphones. This analysis was made with the intention of replacing a commercial solution, which was being used, by an open source solution. The analysis of those webphones was done taking into account a set of pre-defined requirements. After this process of analysis, the selected solution was integrated into the conference service interface.*

## Keywords

**Integration, Sharing of Information, Conference, Collaboration.**

## 1 Introduction

The recent evolution of Internet provides a set of features that allows people to share all kinds of information, using several types of applications. Nowadays, there are applications that allows the users to share their pictures (E.g., *Hi5* [1]), their movies (E.g., *YouTube* [2]), and even their own thoughts (E.g.,  *Blogger* [3]). It is also possible for users to choose how they want to organize their webpages (E.g., *iGoogle* [4]), adding and removing items, changing the webpage themes, etc. Now more than ever, is important develop new applications that allow users to create, manage and share the information that exists on the Internet.

We live in an age where is increasingly difficult to interact with people personally, because the people are mostly occupied. So, it is important to have a service that allows people to communicate, not just by writing text on a page, but also speaking and seeing the person with whom they are talking. Together with all these factors, the communication also has to be at the lowest possible cost. Now, imagine these features used not just to talk with one person, but to talk with a lot of people, at the same time, wherever they are. This whole set of features turns *web communications* more personal and attractive.

### 1.1 Motivation and Goals

Instead of having an open application for audio and video conversation, another application for chatting, and yet another one for file sharing, with this new service the user only needs to open a browser. This does not only saves computer resources, as yet provides a more organized work environment. This type of collaboration and conference services allows that, with only an Internet browser, the user can interact with several people.

So, the main goal of this work is to add features to a service, that has already available some resources, such as audio conversation, file sharing, video sharing and chat service. The new features will enrich the service providing two more applications, from where the users can share data with others participants, in a collaborative environment.

With the intent of reduce even more the number of needed applications, to interact with the service, another goal is to integrate a webphone. This will avoid the requirement of having a softphone opened to enter in a conference room, and to communicate with the others users.

This work can also be an important contribution for companies that have offices throughout the country, for example. Imagine that there is an employee of the company that is at an office at 300 km from the

headquarters of the company, and must have weekly meetings with the Director. Instead having to go to meetings, the employee may use the service to start a slideshow sharing with the Director, and make the presentation of the weekly reports. This will save time travel of the employee, as well as the money of the company spent on these trips.

## 2 State of the art

### 2.1 Web 2.0

The concept of "Web 2.0" was born of the idea that the web is increasingly important, because of the exciting new applications and sites that are emerging at the present-day [5]. Today, the term "Web 2.0" has been widely adopted to describe a new web stage where web applications are increasingly more collaborative and depend on contents distributed by the final users, instead of conventional static information repositories.

However there is still a disagreement about what Web 2.0 really means, because some people believe that the term is just a marketing buzzword and others believe that Web 2.0 is the new conventional wisdom.

Some examples of applications in Web 1.0 and Web 2.0 are Britannica Online (Web 1.0) vs. Wikipedia (Web 2.0), Personal websites (Web 1.0) vs. Blogging (Web 2.0), and Content Management Systems (Web 1.0) vs. Wikis (Web 2.0).

### 2.2 SIP

SIP (Session Initiation Protocol) includes some elements of two well-known Internet protocols: HTTP (HyperText Transfer Protocol) [6], that is used for web browsing, and SMTP (Simple Mail Transfer Protocol) [7] for e-mail. From HTTP SIP uses the client-server model, as well as the usage of URL (Uniform Resource Locator) [8] and URI (Uniform Resource Identifier) [9]. Of SMTP, SIP applies the text encoding scheme and the headers style that are used on SMTP messages. Some examples of these headers are *To*, *From* and *Subject*.

An advantage for SIP from using these elements is related with the possibility of extensions. That is, SIP can be changed to support new features and services, as call control services, mobility and interoperability with the telephony systems that already exists.

### 2.3 IMS (IP Multimedia Subsystem)

Currently, an important architecture that uses SIP is IMS. IMS is an overlay service network architecture that can be applicable to any kind of IP (Internet Protocol) network as, for example, CDMA2000

(Code-Division Multiple Access version of the IMT-2000 standard), GSM (Global System for Mobile Communications)/GPRS (General Packet Radio Service), UMTS (Universal Mobile Telecommunication System) and WLAN (Wireless Local Area Network) [10]. Its a global standard that is supported by 3GPP (Third Generation Partnership Project), 3GPP2 (Third Generation Partnership Project 2), ETSI (European Telecommunications Standards Institute), OMA (Open Mobile Alliance) and IETF (Internet Engineering Task Force), and can be considered as the Universal Service Delivery Platform for NGN (Next Generation Networks), also supporting FMC (Fixed Mobile Convergence). Because there is no real IMS services that are standardised, for now IMS should only be considered as a service enabler.

A more specific definition of IMS is that it is a platform for multimedia service control. It combines real time resources, such voice and video-conference, with non-real time services, independently of the radio technology that is used. The IMS architecture intends to provide the next group of advantages:

- support to sophisticated multimedia services;
- QoS (Quality of Service);
- FMC of services and network operations;
- support of Legacy services;

### 2.4 MVC (Model-View-Controller) Pattern

An important architectural pattern used in software engineering is MVC [11]. By applying this pattern it is possible to separate core business model functionality from the presentation and control logic that uses this functionality. Such separation allows multiple views to share the same enterprise data model, which makes supporting multiple clients easier to implement, test and maintain.

As it can be seen in figure 1, the *Model* represents the information (the data) of the application and the business rules used to manipulate the data. The *View* corresponds to elements of the user interface such as text, checkbox items, etc. Finally, the *Controller* manages details involving the communication to the model of the user actions, such as keystrokes and mouse movements.

## 3 Web Communication Services

In the context of this work two independent services called *Tagarela* and *WebHuddle* were used. The service *Tagarela* is a solution in development at PT Inovação for conference and collaboration, while *WebHuddle* is an *open source* service for web conferencing as well.

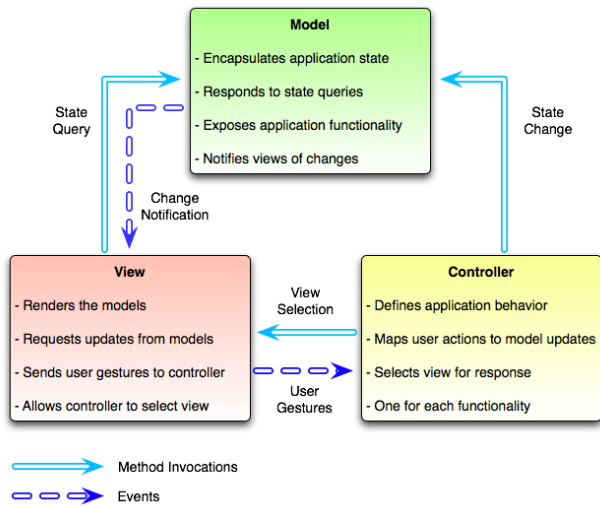


Figure 1: Representation of the MVC structure.

The relation between these services are the fact that the *WebHuddle* will be used to add two new features to *Tagarela*. These features are desktop sharing and slideshow sharing.

### 3.1 Tagarela

*Tagarela* is a Conference and Collaboration solution that is under development at PT Inovação, which is part of an architecture called *Shipnet* [12]. This architecture includes the family of products to give answer to the challenges and needs of the NGN, in the FMC scenario.

*Tagarela* provides a wide range of features [13]. These features include chat, video conferencing, file sharing and audio conversation. Under this work, two need features were added: desktop sharing and slideshow sharing. These features were implemented using the *WebHuddle* (it will be presented in section 3.2).

#### 3.1.1 Entities

In *Tagarela* there are different types of users, also called entities, and each of them have different roles in the service. They are organized hierarchically, being the Administrator the entity with more privileges.

- **Administrator:** Entity that is responsible for managing the service. The Administrator is also responsible for the creation of the *Tagarela* services classes, and by the definition of the rules of the use (type of shared resources, maximum number of participants, moderator, etc);
- **Session Creator:** Entity responsible for the creation of the *Tagarela* sessions (thematics and/or ad hoc conferences) and definition of the sessions

policies and respective mode of control, according to the privileges given by the Administrator.

Although the Session Creator is not necessarily a participant of the sessions created by him, he has the full control of the sessions. The only superior entity of the Session Creator is the Administrator of the service;

- **Moderator:** Is the entity responsible for management, in real time, the conferences. During a conference, the moderator has the ability to give, revoke and/or deny the ability to use the room resources (file sharing, video sharing, chat, download files that are being shared, etc), to participants. The moderator can also remove a participant from the conference.

The choice of the moderator and his moderation functions are defined according to the policies defined by the Session Creator. The moderator can also be chosen during the conference creation;

- **Participant:** This entity represents the user without moderation privileges, when he is in a conference room. Once the user is no longer in the meeting, he will become a simple user. The functionalities of participants control are defined by the Session Creator;

- **User:** Is the entity that executes the actions outside the sessions. For example, the user is who do the subscription and scheduling of an *ad-hoc* meeting.

### 3.2 WebHuddle

*WebHuddle* is a multi-platform *open source* application, based on a web server, for web and video conferencing [14]. The goal of analysing this application is to add two new features into a Conference Service that is under development at PT Inovação, called *Tagarela*. The features that were added to the Conference Service are the desktop and slideshow sharing. With these features the user will be able to share their desktop, pictures and presentations, with the users who are at the meetings.

#### 3.2.1 Architecture

The *WebHuddle* architecture is composed by a Server that publishes its services on the Internet, through JBoss [15], in a configurable URL (Uniform Resource Locator), and the users who access the available services. But the WebHuddle Server himself acts as a client and a server. This is, the client is the entity that receives the requests from the browser, and sends these requests to the server entity. Another important feature of *WebHuddle* is that it is based on the

MVC (Model-View-Controller) pattern (defined in section 2.4), because the *jsp* pages are used to render the *View*, the Servlet works as *Controller* and components EJB (Enterprise JavaBeans) act as the *Model*.

The *WebHuddle* Server is the entity responsible for the management of all the interactions between users. Furthermore, as the events are received, the server has to perform the required actions. With regard to information, this is stored in a DB (Database) of the type HSQLDB (Hypersonic SQL), and it has all the information about the registered users. For example, the pictures that the users have on their accounts, the meetings that they have attended and the contents (users and pictures) associated to each meeting. This information is stored in a file at the *WebHuddle* Server.

To make the service available on the Internet, it is required that the service has been deployed at JBoss, which then should be started. JBoss is an *open source* application server, based on J2EE (Java 2 Enterprise Edition) platform. It is implemented in Java and, because of this, it can be used in any OS (Operating System) that supports Java, just as what happens with *WebHuddle*. When the JBoss service is started, *WebHuddle* will be available on the Internet, at the *WebHuddle* Server IP address.

When the service is published on the Internet, a group of servlets, on the server side, will be available and, on the client side, there are a group of applets that are loaded on the client browser. A representation of the *WebHuddle* architecture is depicted in figure 2.

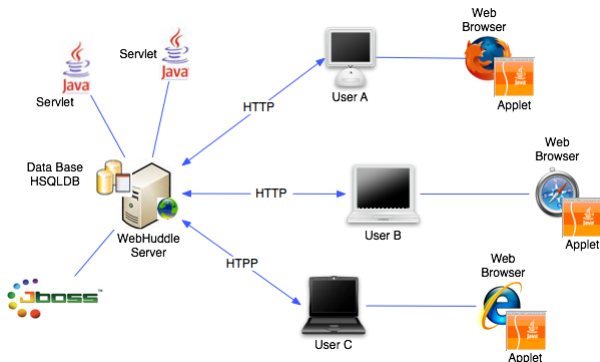


Figure 2: Architecture of *WebHuddle*.

To do the interface between the *WebHuddle* Server and the DB, EJBs are used. When the service is deployed it creates the DB tables.

### 3.2.2 Using Struts in WebHuddle

*Struts* is an *open source* framework to implement applications using the MVC design pattern. This framework provides the basic infrastructure for implementing MVC. This allows developers to concentrate on the business logic, without worrying about the others parts of the architecture [16].

The *Controller* is provided by *Struts* in the form of *ActionServlet* class. This class is used for handling all the requests. For *WebHuddle* to allow the use of this controller servlet, the code represented in figure 3 is included in the deployment descriptor ("web.xml"). In file "web.xml" must also be specified the mapping of *WebHuddle* URLs to this servlet. The code of figure 3 also defines that any URL with ".do" extension calls the *ActionServlet* to handle the request.

```
<servlet>
  <servlet-name>action</servlet-name>
  <display-name>action</display-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>/*.do</URL-pattern>
</servlet-mapping>
```

Figure 3: Example of code that allows to enable a controller servlet.

### 3.2.3 Entities

In *WebHuddle* there are only two entities that interact with the application. They are:

- **Moderator:** Is the entity that starts a desktop sharing or a slideshow sharing. It controls what the other users will see, during the sharing:
  - In one **desktop sharing** is responsible for deciding when to start and to stop the sharing. The moderator may also decide if wants to share all desktop or just a single area, bounded by a rectangle;
  - In one **slideshow sharing** is responsible for choosing the slides that wants to share, and the order of the presentation of the slides. The moderator also has the ability to do some kind of edition of the images, through the tools available at the sharing.
- **User:** Entity that joins into one desktop or slideshow sharing. It has no control on what is happening during the share.

## 4 Implementation

### 4.1 WebHuddle and Tagarela Integration

In this section it will be presented how the new features were added in the *Tagarela* service. In figure 4 are shown the changes made, in *Tagarela* architecture, after the integration process. In this figure it can be seen two new components of the architecture. These components will be presented next.

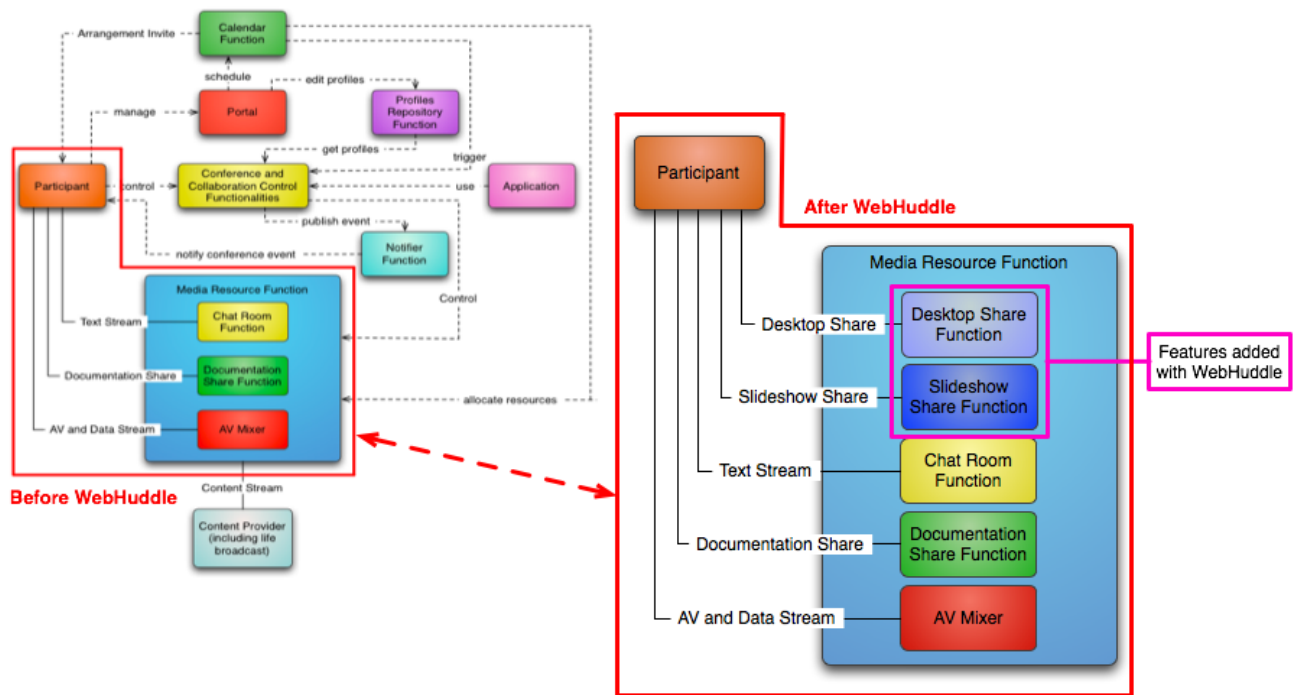


Figure 4: Changes to the functional architecture of *Tagarela*, after the integration of the *WebHuddle* features (desktop sharing and slideshow sharing).

- **Desktop Share Function:** this new group service functionality allows to control the desktop sharing. It provides the ability to show, to others participants, what is happening on the desktop of the user that started this type of sharing. This does not allow others participants to access the desktop that is being shared;
- **Slideshow Share Function:** this new group service functionality allows the users to share and control a set of pre-uploaded pictures and presentations.

It is important to make a distinction between two concepts. Both in *Tagarela* and in *WebHuddle*, there is an entity called *Moderator*. After the integration, the *Moderator* of *Tagarela* (from now on designated as *ModeratorConf*) may not correspond to the same as the *WebHuddle*'s (from now on designated as *ModeratorWH*). In brief, the *ModeratorConf* is the responsible for managing the conference room of *Tagarela*, and the *ModeratorWH* is the participant of *Tagarela* who starts the desktop or slideshow sharing.

The entities *ModeratorWH* and *ModeratorConf* do not need to be the same entity, because the desktop sharing or slideshow sharing do not need to be started by the *moderator* of a conference room.

#### 4.1.1 Examples of Features

**Start Desktop Sharing** Once inside the conference, a participant may decide to start one desktop sharing. To do this, it is only required that the room has the desktop sharing resource available, and must click in the "Start Desktop Sharing" button, at the *Floor Control* area. 🖥️

When a participant clicks on the "Start Desktop Sharing" button, there are made a set of verifications, to see if that request is possible to be attended or not. These verifications include to check if the participant has permission to use the desktop sharing, if there is no desktop sharing ongoing and if the participant is not already in a sharing.

**Request Permission** Once inside a conference room, the *ModeratorConf* can revoke any resource permission to the participants. To do this, he/she only needs to click on the respective resource associate to the desired participant, on the *Floor Control* area. In the case that the resource revoked is the desktop sharing or the slideshow sharing, and the participant wants to use the resource, he/she can click in one of two buttons: or in the resource next to his/her name, or in the button at the bottom of the *Floor Control* area.

When the participant request the permission to use a revoked resource, the *ModeratorConf* will receive the information shown in figure 5.



Figure 5: *ModeratorConf* receives permission request.

The *ModeratorConf* when sees the request permission may click on the resource and a window will be opened, as the one that is presented in figure 6. Now, the *ModeratorConf* may decide if wants to accept or deny the request permission.



Figure 6: Interface for *ModeratorConf* decide if wants to give permission or not.

## 4.2 Webphone Integration

In order to integrate a webphone into the *Tagarela* webpage it was done an analysis on two webphone SDKs (Software Development Kit) and then it was analysed one *open source* softphone. After implementing some changes on the softphone, it was created one *plug-in* that was then integrated in *Tagarela*.

**Webphone vs Softphone** First of all, it is important to make a distinction between a softphone and a webphone, so that their differences can be understood. Softphone is a software application, that is installed in the computer, and allows to make calls over the Internet, using the mouse or keyboard to dial phone numbers. To use a softphone, the computer must have a sound card, plus a speakers or headset, and a microphone. On the other hand, a webphone has the same feature as a softphone, but with the difference that it does not have to be directly installed in the computer. Instead, the webphone is integrated in the HTML (HyperText Markup Language) code of a webpage, and automatically downloaded by the browser whenever it is required.

This analysis was made with the aim of replacing an existing webphone solution, at *Tagarela*. The purpose of the webphone, inside the *Tagarela* webpage, is to allow the users to use *Tagarela*, without the need to launch a softphone application. Thus, if the users want to use *Tagarela*, they only need to open the Internet browser.

The two webphones that were tested and analysed were *Abbeyphone VOW (Voice Over Web)* of Abbeynet [17] and *eyeP Foundation* of eyeP Media [18]. The softphone which was used to be integrated in *Tagarela* is called *Emansip* [19]. This softphone is based in one *open source* softphone called *Linphone* [20]. The result of this analysis is presented in table 1.

	VOW da Abbeynet	EyeP Media	Emansip
IE 6	✓	✓	✓
IE 7	✓	✓	✓
Firefox	✓	✗	✗
Safari	✗	✗	✗
Opera	✗	✗	✗
Windows XP	✓	✓	✓
Windows Vista	✓	✓	✓
Mac OS	✗	✗	✗
Linux	✗	✗	✗
Support to earlymedia (RFC 3960)	N/A	N/A	N/A
Audio / Video Codecs	✓	✓	✓
Video Resolution: SD / HD	N/A	N/A	N/A
Installation of the plug-in in a transparent way	✓	✗	N/A
Installation/use to users without administration privileges	✓	✓	N/A
Support of SIP TLS	✗	✗	✓
Support of SIP SRTP	✗	✗	✓

✓ – Supported.  
 ✗ – Not Supported.  
 N/A – Not Available.

Table 1: Summary table with the result of the study of the webphones.

### 4.2.1 Emansip

With *Emansip* the users can make and receive audio and video calls, as with others softphones. The *Emansip* Toolkit is developed in C++ and it uses three main libraries: *amsip*, *eXosip2* and *mediastreamer2*.




In order to implement and integrate the webphone, in *Tagarela*, through the *Emansip* softphone, it was provided the *Emansip* toolkit. In the code of this toolkit there is a GUI (Graphical User Interface) to test the *Emansip* features, as register one SIP account, and make and receive calls with video included. Through this application, developed in **VB! (VB!)**, it was made some changes to the toolkit, as it is explained in the following section. After this, it was generated a **DLL! (DLL!)** that was then used to create an ActiveX object. This object was later used to create a webphone and integrate it into *Tagarela*.

An ActiveX object is a COM (Component Object Model) developed by Microsoft for Windows platforms. Through the use of COM runtime, it is possible to develop software components, that perform functions. Several Microsoft Windows applications,

such as IE and Microsoft Office, use ActiveX controls to build their feature set, as well as encapsulate their functionality as ActiveX controls, so that the features can be embedded in other applications. IE also allows the ActiveX controls to be embedded inside web pages.

#### 4.2.2 Webphone Implementation


In the *Tagarela* interface there is an area where it can be seen one icon corresponding to the state of the webphone registration. When using IE! (IE!), this icon is available and allows that, depending the color that it has, the user can see if the webphone account is:

- **disabled**  : indicates that the webphone is not registered;
- **registering**  : image informing that the webphone is being registered;
- **ready**  : when the webphone is registered, this is the picture that is shown;

In the *Tagarela* code, when is detected that the user is using IE and he has a webphone account at the DB, the application informs the user to wait for the registration of the webphone. Then, it is called a function, called *initWebphone*, with three parameters: *webUser*, *webPassword* and *webDomain*. This values are those that are obtained from the DB when the user has an entity started by "web:".


Inside the function *initWebphone()* is created an ActiveX object, with the name "plugin". After, this object is used to call a function, called *initialize()*, with the same parameters that were used for *initWebphone()*. This function allows to initialize the *plug-in* with the received parameter.

After the execution of the function *initialize()*, from the ActiveX object, is called a function, from this same object, to register the SIP account, *API\_am\_register()*. This function is only used to call another one, *am\_register\_start()*, at the *amsip* library, of the *Eman-sip* toolkit. The function *am\_register\_start()* configures the webphone to be registered with the SIP account of the user. When the initialization of the webphone is finished, the user is informed through the next message: "Your webphone is registered and ready to use!". With this, the user may use the webphone to join any conference room.

When the participant wants to join a conference room, using the webphone, only needs to click on the "Join the Conference" button (  ) and choose the account of the webphone. After this, the application starts the call to the webphone with the username equals to "webUser", and then the call will be answered. To make possible the integration of the webphone, into *Tagarela*, it was made a change to the way how the calls were answered by the toolkit. This is,

the calls received by the *plug-in* must be answered automatically.

Initially, when a call was received, the user needed to click on the "ClickHoldCall" button to answer it. Now, when the event of receiving a call is detected, by the function *API\_process\_event()*, the call is answered automatically. This is done by calling the function *ClickHoldCall()*, with the parameter "1". This parameter corresponds to the number of the line of the call, because the *Emansip* toolkit can have four lines working at the same time. But, because the *plug-in* only needs to work with one call, at a time, the line that is used is the number "1".

To end a phone call is used the function *ClickStopCall(1)*. The value "1" is used for the same reason explained previously. Inside *Tagarela* this function is called when the participant clicks on the "Leave Conference" button (  ), inside the conference room.

## 5 Tests

To assess the effectiveness of adding the three new features into *Tagarela*, several functional tests were performed. These tests only check the functionality of the product, since this work represents a proof of concept of the integration of two independent services.

To the realization of the tests two computers were used: one that was the server and other that was the client. The required tools were an Internet browser, Mozilla Firefox or IE, depending on the type of test that was being done, and a softphone or a webphone with a SIP account registered, also depending on the test.

### 5.1 List of Functional Tests

The list of the functional tests is presented bellow. These tests allows to verify if all the features added to *Tagarela* are working or not, doing a set of actions.

1. T1. Create new thematic conference, with desktop and/or slideshow sharing resources;
2. T2. Create new ad hoc conference, with desktop and/or slideshow sharing resources;
3. T3. Register account of webphone, after use IE to login in the *Tagarela* homepage;
4. T4. Use the webphone to join one conference room;
5. T5. Use a softphone to join one conference room;
6. T6. Start one desktop sharing;
7. T7. Start one slideshow sharing;
8. T8. Join one desktop sharing;

9. T9. Join one slideshow sharing;
10. T10. Ask permission to use the desktop sharing resource;
11. T11. Ask permission to use the slideshow sharing resource;
12. T12. Upload one presentation;
13. T13. Try to start a desktop/slideshow sharing, when there is one already underway;
14. T14. Try to join one desktop/slideshow sharing, when there is no sharing going on;
15. T15. Manage the folder of slides to use in a slideshow sharing (add or remove slides);
16. T16. Send/receive audio through the webphone, inside a conference;

## 5.2 Results and Analysis

Before presenting the test results, there are some details that need to be mentioned. Almost all the tests were made using two computers. One computer was the server, because is where the service was running, and the other computer represents the client of the service. The only exceptions were the tests T3, T4 and T15. Because these tests were the ones that included the webphone, and the *plug-in* was only installed on the server.

Another important information is about tests T10 and T11, because the *ModeratorConf* only can revoke a resource permission after the participant have joined the conference room. So, to do these tests, the participant need to join the conference, and wait that the *ModeratorConf* has removed the permission.

### 5.2.1 Number of Required Actions

In this section are presented the results of the usability tests. In figure 7 it can be seen the number of actions that are required, in order to perform the respective tests. Concerning to tests T1 and T2, although in those tests must write various parameters, this writing was considered only as one action.

Through figure 7 it can be seen that the tests T7, T12 and T15 are those that need more steps to be executed. This is no coincidence, because these tests are testing the functionalities that are associated with slideshow sharing. So, it is also need to choose the slides that wants to share, besides clicking on the same buttons that are used in desktop sharing. Apart from these three exceptions, the most tests only need nine steps to be executed.

Concluding, through figure 7 it may be seen that, concerning to the usability, the adding of the new features in *Tagarela* was successfully, because the new features can be used as easily as the original ones.

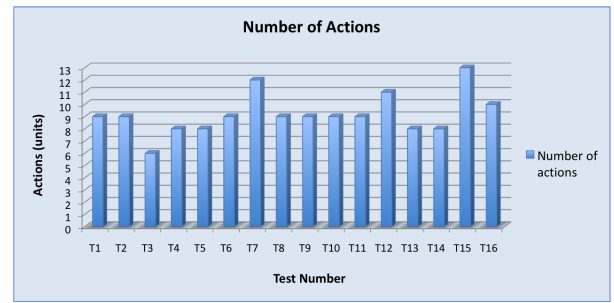


Figure 7: Chart with the representation of the tests of usability result.

### 5.2.2 Duration of the Tests

Another type of tests that was done had as purpose the measurement of time. For each test five measurements were made, with the aim of having a more reliable result. In figure 8 is depicted the maximum, minimum and average values of the measurements made for each test.

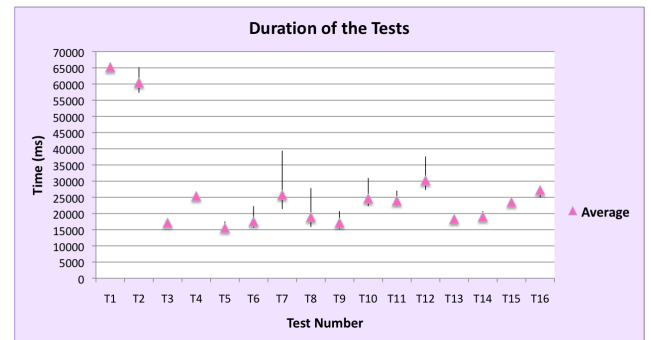


Figure 8: Chart representing the maximum, minimum and average duration of each test.

Through this figure it can be seen that T1 and T2 are the tests that need more time to be executed. This is due to the fact that the user must fill several parameters to create the conference room. However, T1 and T2 have a difference of a few seconds, because T2 has less parameters to fill, and so it takes less time.

There is also a considerable difference in the times of T6 and T7, because for T7 the service need to load the images to the applet, while to start the desktop sharing is only need to load the applet. And if the number of slides is big, more time will be needed to start the slideshow sharing.

Another conclusion that can be taken from figure 8 is that the times of test T4 and T16 are similar. This happens because the only difference between them is the fact that the user need to say something, to be heard at the conference. So test T16 is supposed to take longer than T4, but only a few seconds.

About tests T10 and T11, they also need more time to be executed because, once inside the conference



room, the participant have to wait for the *ModeratorConf* to remove the permission, and then ask for it.

Another conclusion that can be taken from figure 8 is that the execution time for test T12 depends on the number of pages that the uploaded presentation has. So, the greater the number of pages, the longer it will take for the presentation uploading. Finally, the time that the test T15 needs to be done, with successful, depends on the number of slides that the participants wants to modify. For this test, it was selected the folder that wanted to share, and then it was deleted only one slide.

## 6 Conclusions

During this document it was explained the technologies that were studied and used, how was implemented the integration of two specific services, and finally were presented the list of the executed tests to verify the functionality of the new features.

Through the tests it was verified that the new features are easy to use. They not only need relatively few actions to be executed, nor are slow to be made. The majority of the tests need around nine actions, including the login where the user need to write the username, password and desired language.

With this dissertation the *Tagarela* service was enriched with three new features: desktop sharing, slideshow sharing and a webphone. These features are important, because it is unusual for a conference service to have features that are not only audio, video and chat conversation.

### 6.1 Future Work

Once inside a conference room, when a participant wants to start one sharing, it is necessary to click on the respective button. But, before the sharing being started, the service must perform some verifications. These verifications are necessary to know if the participant has permission to use the resource, or if the resource is not being used already. Depending on the result of the verifications, the participant may be informed that can not use the resource, or that the sharing is already started.

A possible feature that could be studied and implemented as future work is replacing the way how the transmission of the content of the new resources is made. Instead of using HTTP (HyperText Transfer Protocol) to verify resource permissions and transmit the content of the sharings, it could be used SIP to do the verifications and the content of the sharing could be sent inside video stream.

After the integration of the video feature in the webphone *plug-in*, the information exchanged during the

desktop sharing and the slideshow sharing can be sent as video stream. In order to do this, instead of doing the resources admission via HTTP, this control could be done using SIP. A possibility of implementation could be through the following steps:

- when a participant wants to start a desktop or slideshow sharing, could be sent an INVITE message to all the participants that are in the conference room;
- in this INVITE message it will be sent, within the video header, the information about the sharing;
- when the participants receive the INVITE message, may decide if want to send the OK message, accepting the invite to join the sharing;
- when the *ModeratorWH* decides to end the sharing, the respective BYE message will be sent to the participants inside that sharing.

Besides this, it is also needed to end the development of the webphone *plug-in*. Due to the lack of time the *plug-in* does not have the video feature, only allowing audio conversation. One possible solution to implement this feature is, after capturing the video from the webcam, present it in the *Tagarela* webpage. Once inside the conference room, the communication with the webphone is established, transmitting audio and video to the room.

There is still another feature that could be implemented in future work. In the case that the service is used, for example, in a Call Center, whenever a customer calls with a question, and is used the desktop sharing, this interaction can be recorded. Then this recording will be sent to the customer, and if the customer ever again has the same problem, will not need to call to the Call Center, because will have access to the recording. The implementation of this feature could be done using the *WebHuddle*, because already has it available. But, for this work, the recording feature was removed, because it was not part of the work.

## References

- [1] <http://www.hi5.com/> (on-line September 2008)
- [2] <http://www.youtube.com/> (on-line September 2008)
- [3] <https://www.blogger.com/> (on-line September 2008)
- [4] <http://www.google.com/ig> (on-line September 2008)
- [5] O'Reilly, Tim, " *What Is Web 2.0* ", September 30, 2005.

- [6] IETF RFC 2616: "*Hypertext Transfer Protocol – HTTP/1.1*", June 1999.
- [7] IETF RFC 821: "*Simple Mail Transfer Protocol*", August 1982.
- [8] IETF RFC 1738: "*Uniform Resource Locators (URL)*", December 1994.
- [9] IETF RFC 2396: "*Uniform Resource Identifiers (URI): Generic Syntax*", August 1998.
- [10] Lauretti, Samuel R., "*Evolução das Redes de Telecomunicação: Arquitetura IMS*", December 06, 2004.
- [11] "*Java BluePrints: Model-View-Controller*", <http://java.sun.com/blueprints/patterns/MVC-detailed.html> (on-line August, 2008)
- [12] PT Inovação, "*SHipNET*", <http://www.ptinovacao.pt> (on-line July 2008)
- [13] PT Inovação, "*Conferência e Colaboração - Descrição Técnica*", Version 1.0.
- [14] <http://www.webhuddle.com>. (on-line November, 2007)
- [15] <http://www.jboss.org>. (on-line July, 2008)
- [16] [http://www.oracle.com/technology/sample\\_code/tech/java/j2ee/jintdemo/tutorials/Struts.html](http://www.oracle.com/technology/sample_code/tech/java/j2ee/jintdemo/tutorials/Struts.html) (on-line August, 2008)
- [17] <http://www.abbeyphone.com/>. (on-line May, 2008)
- [18] <http://www.eyepmedia.com/products/toolkits/>. (on-line May, 2008)
- [19] <http://www.antisip.com/>. (on-line June, 2008)
- [20] <http://www.linphone.org/index.php/eng>. (on-line June 2008)