SoundLog

Make More Noise!

André Filipe Mateus Ferreira

Master Thesis in
Information and Software Engineering

Advisor: Daniel Gonçalves

Jury

Chairman: João Madeiras Pereira
Members: João Cachopo

October 2010
Abstract. Sound is traditionally used for its content along with automated speech recognition techniques to understand what is said. But sound, besides its semantics, can be used to gather useful information. In this paper we propose SoundLog, a system that analyses sound and crosses it with contextual and autobiographic information to gather insights from different situations (public presentations and meetings). Our goal is to provide users knowledge regarding those situations, both in real-time and later on, without the need of training or use of devices. This can help them to correct or adapt their behaviour, and to review past events in meaningful ways. For that, we start by describing the state of the art with works that analyse sound and then works that analyse oral and textual conversations. Through different but relevant criteria, we compare all of them concluding that none of the works do exactly what we desire. Afterwards, we describe all the SoundLog’ requirements and how we have implemented them. We end the master thesis with the system evaluation.

Keywords. Low Sound Features, Subject Identification, Speech Recognition, PowerPoint Presentation, Scenarios, Framework and Interface
Table of Contents

1 Introduction ................................................................................................................................ 1
1.1 Motivation ................................................................................................................................ 1
1.2 Goals ......................................................................................................................................... 2
1.3 Contributions .......................................................................................................................... 3
1.4 Master Thesis Structure ....................................................................................................... 3

2 State of the Art .......................................................................................................................... 5
2.1 Works that Analyse Sound ..................................................................................................... 5
   2.1.1 SoundSense ..................................................................................................................... 5
   2.1.2 Noise at the Heathrow Airport ...................................................................................... 6
   2.1.3 NoiseSPY ..................................................................................................................... 7
2.2 Works that Analyse Oral or Written Conversations .............................................................. 9
   2.2.1 The Chat Circles Series ................................................................................................. 9
   2.2.2 Conversation Clusters .................................................................................................. 10
   2.2.3 Conversation Clock ...................................................................................................... 11
   2.2.4 Visualizing Remote Voice Conversations .................................................................. 12
   2.2.5 Visualizing Conversation (Loom) ............................................................................... 13
   2.2.6 Influencing Group Participation with a Shared Display ........................................... 14
   2.2.7 Designing Interfaces that Influences Groups Processes ........................................... 15
   2.2.8 Visiphone .................................................................................................................... 16
   2.2.9 In Situ Speech Visualization in Real-Time ................................................................. 16
   2.2.10 Conversation Votes .................................................................................................... 18
   2.2.11 Egocentric Analysis and Visualization of Instant Messaging Activity ................. 18
   2.2.12 CrystalChat: Visualizing Personal Chat History .................................................... 20
   2.2.13 Human Semantic Interactions in Meetings ............................................................... 22
   2.2.14 PrimaVista ................................................................................................................ 24
2.3 Discussion ............................................................................................................................. 26
   2.3.1 Criteria Description ...................................................................................................... 26
       Number of People .............................................................................................................. 26
       Time Period ..................................................................................................................... 26
       Temporal Notion ............................................................................................................. 26
3.1 Scenarios .................................................................................................................. 33

3.2 Requirements ............................................................................................................ 34
  3.2.1 Capture .................................................................................................................. 34
  3.2.2 Visualization ......................................................................................................... 34
  3.2.3 Analysis .................................................................................................................. 35
  3.2.4 Others ..................................................................................................................... 35
  3.2.5 Requirements Summary ....................................................................................... 35

3.3 System Overview ....................................................................................................... 36

3.4 Framework ................................................................................................................ 37
  3.4.1 Sound Capture ....................................................................................................... 37
  3.4.2 Low Sound Features ............................................................................................. 37
  3.4.3 Filters ..................................................................................................................... 38
  3.4.4 Record System ....................................................................................................... 38
  3.4.5 Voice Identification ............................................................................................... 39
  3.4.6 Framework Application Test .................................................................................. 40
  3.4.7 Speech Recognition ............................................................................................... 41

3.5 PowerPoint Add-In .................................................................................................... 43

3.6 Python Interface (Scenarios) .................................................................................... 45
  3.6.1 Layout and System Flow ......................................................................................... 45
  3.6.2 Capture Package ..................................................................................................... 48
  3.6.3 Plugins Package ..................................................................................................... 50
    Noise Plugin ................................................................................................................... 53
    Subjects Plugin ............................................................................................................. 53
    Words Plugin ............................................................................................................... 54
    Slides Plugin ............................................................................................................... 55
André Ferreira v

6 References .............................................................................................................................. 83

7 Annex ...................................................................................................................................... 87

7.1 SoundLog Evaluation ........................................................................................................... 87

7.1.1 The Ten Usability Heuristics – Jakob Nielsen, 1994 ...................................................... 87

7.1.2 Heuristic Evaluation Guide ........................................................................................... 88
   Heuristic Evaluation Tasks ..................................................................................................... 88
   Heuristic Evaluation Questionary .......................................................................................... 89

7.1.4 Prototype Evaluation ....................................................................................................... 91
   Class/Lecture Evaluation Tasks ............................................................................................. 91
   Meeting Evaluation Tasks ...................................................................................................... 91
   Evaluation Satisfaction Questionary ...................................................................................... 92
Figures List

FIG. 1: VISUALIZATIONS OF THE HEATHROW AIRPORT STUDY 6
FIG. 2: NORSONIC VS NOISESPY SYSTEMS NOISE LEVEL MEASUREMENT 7
FIG. 3: NOISESPY MOBILE APPLICATION FORM 8
FIG. 4: A) GOOGLE MAPS NOISE VIEW AND B) GOOGLE EARTH KML NOISE VIEW 8
FIG. 5: CHAT CIRCLES INTERFACE 9
FIG. 6: TALKING IN CIRCLES INTERFACE 9
FIG. 7: CONVERSATION CLUSTERS INTERFACE – TOPIC VISUALIZATION 10
FIG. 8: CONVERSATION CLUSTERS INTERFACE – THREAD VISUALIZATION 11
FIG. 9: CONVERSATION CLOCK INTERFACE 11
FIG. 10: VISUALIZING REMOTE VOICE CONVERSATIONS INTERFACE – HISTORY 12
FIG. 11: VISUALIZING REMOTE VOICE CONVERSATIONS INTERFACE – VOLUME AND PITCH 12
FIG. 12: VISUALIZING REMOTE VOICE CONVERSATIONS INTERFACE – CONTENT 13
FIG. 13: LOOM INTERFACES – A) DOTS FOR POSTS; B) USER POSTING TRACES; C) MOOD 13
FIG. 14: SHARED DISPLAY INTERFACE 14
FIG. 15: SECOND MESSENGER INTERFACE 15
FIG. 16: PEOPLE PARTICIPATION RATE 15
FIG. 17: VISPHONE INTERFACE 16
FIG. 18: HIDDEN WORLDS INTERFACE 17
FIG. 19: RE:MARK INTERFACE 17
FIG. 20: MESSA DI VOCE INTERFACE 17
FIG. 21: CONVERSATION VOTES INTERFACE 18
FIG. 22: ANONYMOUS VOTE AT CONVERSATION VOTES 18
FIG. 23: EGOCENTRIC SYSTEM INTERFACE 19
FIG. 24: CRYSTALCHAT INTERFACE - (A) TOP VIEW; (B) SIDE VIEW 20
FIG. 25: CRYSTALCHAT INTERFACE - DIAGONALLY OPEN CONVERSATIONS 21
FIG. 26: CRYSTALCHAT INTERFACE - TEMPORAL SPACING 21
FIG. 27: CRYSTALCHAT INTERFACE – TRANSLUCENT PLANS THAT DEFINE MODES 21
FIG. 28: CAPTURE SYSTEM – ENVIRONMENT OVERVIEW AND PARTICIPANT DEVICES 22
FIG. 29: INTERACTION VISUALIZATION – MEETING BROWSER OVERVIEW 23
FIG. 30: PRIMAVISTA WEB INTERFACE: LIVE VIEW (LEFT) AND PAST VIEW (RIGHT) 24
FIG. 31: PRIMAVISTA PAST VIEW: WORDS PANE TAG CLOUD 25
FIG. 32: SOUNDLOG SYSTEM 36
FIG. 33: TIMBRE OF A DRUM NOTE 39
FIG. 34: SOUNDLOG APPLICATION DEVELOPED TO TEST AND TO REFINE THE FRAMEWORK 40
FIG. 35: XML SPEECH RECOGNITION FILE 42
FIG. 36: SPEECH RECOGNITION APPLICATION 43
FIG. 37: SOUNDLOG POWERPOINT ADD-IN (TOOLBAR BUTTON, FRAME AND PROGRESS BAR) 44
FIG. 38: SOUNDLOG APPLICATION OVERALL ARCHITECTURE 45
FIG. 39: SOUNDLOG APPLICATION TOOLBAR 46
FIG. 40: SOUNDLOG LAYOUT (TOOLBAR AND PLUGINS) 46
FIG. 41: NEW PRESENTATION TOOL 47
FIG. 42: LOAD PRESENTATION TOOL 47
FIG. 43: SOUNDLOG INTERFACE - CAPTURE PACKAGE 48
FIG. 44: PYTHON COMMUNICATION WITH FRAMEWORK (C# DLL) 48
FIG. 45: SOUNDLOG INTERFACE - PLUGINS PACKAGE 50
FIG. 46: PLUGIN – A) PAN AND ZOOM (RED); B) SWAP (BLUE); C) SLIDERS (GREEN) 51
FIG. 47: SOUNDLOG LAYOUT – FIVE SLIDE TRANSITIONS AND ‘FRAMEWORK’ HIGHLIGHTED 52
FIG. 48: NOISE PLUGIN (SELECTED) 53
FIG. 49: SUBJECTS PLUGIN (SELECTED) WITH ANDRÉ IDENTIFIED. 53
FIG. 50: WORDS PLUGIN (SELECTED) 54
FIG. 51: PLUGIN (SELECTED) AND WITH WORDS FALA AND FRAMEWORK HIGHLIGHTED 55
FIG. 52: SLIDES PLUGIN (SELECTED) 56
FIG. 53: PLUGIN (SELECTED) WITH MORE SLIDES THAT CAN BE VISIBLE PER INSTANT 56
FIG. 54: GROUP OF MIDDLE SIZE SLIDES IMAGE 56
FIG. 55: SOUNDLOG SLIDE INFORMATION 57
FIG. 56: TIMELINE PLUGIN (SELECTED) 58
FIG. 57: TIMELINE PLUGIN (SELECTED AND ONE WORD HIGHLIGHTED) 58
FIG. 58: SOUNDLOG INTERFACE – TOOLS PACKAGE 59
FIG. 59: SOUNDLOG VISUAL REPORT 61
FIG. 60: SOUNDLOG TEXTUAL REPORT 61
FIG. 61: SOUNDLOG HELP TOOL 62
FIG. 62: PROTOTYPE EVALUATION - BES ESDATA MEETING 2 68
FIG. 63: SOUNDLOG NOISE PLUGIN TESTING (SNAPS EVERY TWO SECONDS) 70
FIG. 64: SOUND LEVEL IN THE CLASS TEST 71
FIG. 65: SOUND LEVEL IN THE MEETING TEST 2 71
FIG. 66: WORDS RECOGNIZED WITH IN DIFFERENT SLIDE TRANSITION INTERVALS (5, 3 AND 1S) 72
FIG. 67: SOUNDLOG RECOGNIZED WORDS RATIO PER INTERVAL 73
FIG. 68: SUBJECTS IDENTIFIED IN MEETING TESTS 1 AND 2 74
FIG. 69: SUBJECT IDENTIFICATION IN THE TWO MEETINGS 75
FIG. 70: SOUNDLOG TIME TEST 76


Tables List

TABLE 1: COMPARISON OF THE DESCRIBED WORK 29
TABLE 2: SOUNDLOG SYSTEM REQUIREMENTS 35
TABLE 3: SOUNDLOG USABILITY HEURISTIC NUMBER OF ERRORS REPORTED 65
TABLE 4: PARTICIPANT PROFILE 68
TABLE 5: SOUNDLOG TESTS LOCATION AND NUMBER OF PARTICIPANTS 69
TABLE 6: ATTRIBUTES TESTED PER TYPE OF SCENARIO 69
TABLE 7: SATISFACTION QUESTIONARY RESULTS PER PARTICIPANT – FROM 5 (BEST) TO 1 (WORST) 78
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KML</td>
<td>Keyhole Markup Language</td>
</tr>
<tr>
<td>MSN</td>
<td>Microsoft Network</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel Frequency Cepstrum Coefficients</td>
</tr>
<tr>
<td>VQ</td>
<td>Vector Quantization</td>
</tr>
<tr>
<td>SAPI</td>
<td>Speech Applications Programming Interface</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>PDF</td>
<td>Portable Document Format</td>
</tr>
<tr>
<td>H</td>
<td>Heuristic</td>
</tr>
</tbody>
</table>
1

Introduction

This chapter describes the context of the problem identified in the low use of sound as an important source of information capture; we present our motivation and system goals, report its contributions and describe the organization of this dissertation.

1.1 Motivation

The sound captured through a microphone is a rich information source that can be used to withdraw exact inferences about the speaker, its environment and social events. It is a modality that supports a wide set of features, such as conversation detection, activity recognizer, localization classification, social net structure discovery. However, it is traditionally used for its content, resorting to techniques like Automated Speech Recognition to understand what’s said.

But the sound itself, regardless of its semantics, can provide information. For example, in the course of a class, if the level of noise increases, this can indicate that some slide generated comments or that the students lost interest. This can be used to create a log of the student’s interest in the several parts of the class, to improve it where needed. Similarly, in a meeting or lecture, it can indicate when a polemic subject was referred. If we go deeper, and analyse the sound’s frequency, we can identify when a person is speaking and by analysing the timbre, we can recognize who spoke.
1.2 Goals

Despite sound’s potential as a robust way to obtain information, microphones and inferences based on it have not been given much attention in previous research. Our goal, in SoundLog, is to create a system that analyses sound and relates the information obtained from it with contextual and autobiographic information in order to gain insights about several situations. More than just providing the framework to analyse the sound in rich and relevant ways, we present that information to the users in meaningful ways, based on different scenarios. We will take care to do so in such a way as not to unduly distract the users from their tasks but still make the information readily available if needed. Our final goal is to create a system that don’t need any type of training (for example training for voice identification or speech recognition) and the use of any type of device (head sensors, video cameras, etc.). With the application in place, users only have to start/stop the information capture. So, our goals can be summed in building a system that captures relevant information to gain insights from different scenarios, in order to improve them through the analysis of the captured information.

In order to find out what kind of information can be relevant, we surveyed several previous works in the area. We focused on solutions that analyse the sound itself, oral and written conversations. We decided to analyse oral and written conversations because we wanted to see how to represent dialogues (and both sound and text are orthogonal to the visualization). From the first we determined the most important features (volume for a possible identification of what’s problematic or understood; frequency to differentiate between voice and music or noise, timbre for voice identification, etc.), that can be extracted from sound captured in realistic contexts. From the works that analyse oral conversations, we conclude that voice identification (with subsequence recognition of speaker) and speech recognition must be made in all scenarios (to see who spoke and what was said). The works that analyse textual conversations showed that all the collected information should be logged and how the exchanges between participants can be shown.

Having identified the requirements and effective sound analysing system should possess, we decided to create a reusable framework that analyses the sound characteristics and provides an abstraction layer that enables the quick development of sound-based applications. We will then consider three different scenarios: public presentations (class and lecture) and meetings. These scenarios will allow us to test the versatility of the framework by making use of a wide range of sound-based information, in different contexts. We will create a tool that has an extensible structure on which different details will be given more emphasis depending on the scenario that is considered (e.g. who spoke is more relevant in a meeting than in a lecture). The interface will adapt accordingly. There will be a real time interactive visualization during the events. Afterwards it will be possible to interactively analyse the logs, and produce reports.
1.3 Contributions

In this work, our main contributions are:

- The identification of which sound features we must collect that might be relevant to understand the surrounding context.
- An effective way to visualize sound-based contextual cues in such a way as to provide deep insights to the user.
- A prototype system embodying our ideas, which allows sound to be analysed in three different scenarios: meetings, classes, and public presentations.

1.4 Master Thesis Structure

This document has the following structure. We start by describing the state of the art through works that analyse sound (section 2.1) and works that analyse oral and written conversations (section 2.2). We used this analysis to demonstrate the identified problem and to study how to implement SoundLog system. Then we describe SoundLog implementation (chapter 3) by describing the scenarios where our system is applied, its requirements and architecture. After the complete description of our solution, we present evaluation results (chapter 4) and we end the master thesis with the conclusion (chapter 5) enumerating all the contributions and the future work.
SoundLog!
2

State of the Art

The state of the art is divided into two parts. The first part has studies that analyse the sound and the second part is the next logical step to take following the analysis of sound, and is the analysis of oral and written conversations. We decided to study both types of conversations because we wanted to see how to represent dialogues, and being sound or text it’s orthogonal to the visualization. All the papers presented focus on the scenarios of the application (visualization of the final work) helping us to understand how they should be made.

2.1 Works that Analyse Sound

In this section we present works that analyse sound itself. It helped us to achieve some of the system capture requirements.

2.1.1 SoundSense

*SoundSense* [18] is a scalable framework for modelling sound events through mobile phones. The system runs on mobile phones, uses its microphone to capture sound (without any other auxiliary system), and can identify general types of sound (e.g. music, voice).

Initially a sound is categorized as voice, music or environment sound (which consists of everything else). Then deepens each part (e.g. gender of voice or music). In the case of environment sound, this is learned over time. The sound is not immediately saved, first it’s regarded as a significant sound, but only if it’s repeated several times. When this happens, the user can label or reject the sound.

The sounds in real life are usually a number of different sounds and not just one. *SoundSense* decided not to break them into different sounds; prefers to get the mix of sounds as a signature event. The sound with higher energy level is then considered the principal one. Differences between actual conversations and television are not noted.
With this type of mobility is possible to: detect conversations, recognize activities, classify locations and discover social network structure of a person.

In addition to the described, a strong point of this system is the way it captures the sound. They use resources beyond the volume of a sound as this depends on the direction or location of the phone to the same (and may even vary in more than 30%).

The fact that these microphones don’t capture information above 4 kHz results in the loss of important data. Another important aspect is that mobile phones can’t have heavy and complex algorithms, and so there is some trade-off in the system implementation.

2.1.2 Noise at the Heathrow Airport

This study consists of a set of views that show the sound levels in the area of Heathrow Airport. It focuses on finding the noise created by the vast network of hubs that are in the airport, specifically looking at sources that create noise and then leaving it in a visual format (Figure 1).

**Fig. 1: Visualizations of the Heathrow Airport Study**

- **Spheres**: a ring representing the degree of noise recorded at locations near the airport. A full circle represents two minutes (the average time between the landings of two aircraft).
- **Matrix**: it represents the links between each location, in and around the airport. The location and type of sound are recorded in periods of two minutes.
- **Density**: each column portrays noise levels recorded in and around the airport. Each line represents when the noise level reached above the average of 57 dB. The scale on the left hand side represents every ten seconds. All locations can be traced via the location map. Starting left with location 01 the columns run in sequential order to location 15. A complete column represents two minutes.
- **Map**: is a dB noise map of Heathrow Airport runways 1 and 2. The map shows areas with a specific dB level.

---

2.1.3 NoiseSPY

NoiseSPY [14] is a noise monitoring system that allows users to explore a city area while collaboratively collecting and visualising urban noise levels in real-time. This user study has two goals:

- One was to demonstrate that a mobile phone enabled sensing can be used at city level where it’s possible to share data with other users;
- And the second one was to look at individual journeys to access personal exposures level allowing to determinate if the users are prone to the risk of serious noise pollution.

In order to determinate the sound level, the frequency of the audio samples is calculated and afterwards is applied a weighted filter. With this filter, the sound level meter is thus less sensitive for high and very low frequencies. Figure 2 shows the noise level results that the Norsonic System calculates (a comprehensive system for monitoring and analysis of environmental noise) versus the results of the NoiseSpy system. This shows a small difference between both noise level calculation methods.

![Fig. 2: Norsonic vs NoiseSPY systems noise level measurement](image)

The NoiseSPY system employs standard client-server architecture. The software application runs on a mobile phone (currently any of the Nokia Series 60 3rd generation) and it’s written in Native Symbian C++. By default, the noise is sampled every one second and the noise data is combined with the last GPS location information provided. The GPS information can be taken using the mobile phone or any other external GPS-receiver. When the GPS connection is established, a log file with the timestamp is created and the NoiseSPY starts running. All the data is also combined with the MESSAGE project\(^2\), which provides information about weather, traffic, pollution and environment conditions (for example, bad weather can make more noise).

---

\(^2\) MESSAGE project – Obtained from http://research.cs.ncl.ac.uk/message/test/ in October 2010
In figure 3 there is shown the mobile software application form used to capture the noise information.

For introspection and awareness, it was created an online interface that maps the noise captured (into maps). To do this, the captured sound was mapped to the Google Maps (Figure 4a), the Ordnance Survey OpenSpace and Google Earth KML which automatically opens Google Earth to view a 3D model of the measurements (Figure 4b).
2.2 Works that Analyse Oral or Written Conversations

In this section we present works that analyse oral or written conversations. It allowed us to identify some visualization and analysis requirements.

2.2.1 The Chat Circles Series

The approach taken was to start with a minimalist environment carefully designed (*Chat Circles*) and then making gradual improvements in the design of the environment. The five created projects were made for different market niches (*Chat Circles*, *Chat Circles II*, *Talking in Circles*, *Chatscape* e *TeleDirection*) [11].

In *Chat Circles* [23, 9] each person is represented by a coloured circle with its name near the same (Figure 5). The words written by the person appear in the circle, which increases its size and brightness every time you write new words. Written words will decrease, over time, until they disappear. The colour and the names help distinguish between people. The movement on the screen of *Chat Circles* is important because people can move to different locations to participate in different discussions. This project introduced the concept of auditory notion, people only see participants who are close to them, and others just appear as hollow circles (though they see the increase and decrease). Here, a person can get out of a discussion, and everyone sees it. There is also a log stored with timelines that allows a person see what has been written by other people (inside its “auditory range”).

*Chat Circles II* is a revision of the previous project. It introduces background images, traces of actions and a complete map of the area. The images introduce topics for discussion and you can see them at a distance similar to that of auditory range, although this distance may vary (depends on administrator decision). The action traces show where participants walked and the places where they spoke. There's also the concept of mini map that provides an overview of space (which allows the visualization of hot spots).

*Talking in Circles* [20, 21] is an online speech interface based on the *Chat Circles* model. The circle increases and gets brighter with the voice volume. The notion of auditory range is still implemented and the sound decreases as the speaker gets away from us, or vice
versa. There is also the notion of scribbles that consist on symbols (e.g. ?, !, etc.) that can be put in a circle while someone else is speaking, not interrupting them and giving some personality. Like in *Chat Circles II*, it has topics, but in this case the reach area is coloured (Figure 6).

In *Chatscape* [2] users can create simple behaviours in their icons (which consist of geometric shapes). The icons appearance can change according to the features of nearby users or features that they give to us. It has automatic movement (we can choose to follow or to reject someone).

The TeleDirection [10] changes the context of the interaction of a virtual chat to a real environment. Users have material (e.g. camera) that allows them to start conversations with other people and to carry out operations on the environment (which is indeed the preferred option). It has a video mode where approaching a word of a user means that we vote on it (this means that we accept the user goal). Outside the video movie, the words correspond to conversations between users.

### 2.2.2 Conversation Clusters

It is a system used in meetings that through Speech Recognition, creates sets of conversation topics (*Conversation Clusters*) [3]. The sets are arranged in a shared tabletop allowing its visualization by participants of the meeting (Figure 7).

The Conversation Clusters aim to capture the context of a given time, helping people get an idea of what went on during the meeting. Several mechanisms are used so as not to be diverted attention (the important is the meeting); and for the latest topics be quickly identified.

![Fig. 7: Conversation Clusters Interface – Topic Visualization](image)

When the meeting topic changes, a new one is created and putted on the centre of the tabletop. The others that are no longer used are moved away from the centre. Zoom commands are allowed in order to see with higher or lesser detail.
There is also the thread visualization type, where there are lines of topics that were discussed over time (from left to right). These lines can unite or divide itself (Figure 8). The parallelism of lines means that they talked about a common theme (e.g. the repair of a car has the topic car and finance). It is therefore assessed the relationship of the threads over time and, as in the previous case, the zoom commands are also permitted.

It should be enhanced that, given a set of recognized words, the topics are determined using an algorithm that accesses Wikipedia and relates what was said trying to determine their relationship (topic). Thus, the main constraints are both the possible lack of information (Wikipedia doesn't have everything) and the fact that this system is specific for meetings (not conversation).

2.2.3 Conversation Clock

The Conversation Clocks [4, 16] visualize a conversation up to four people seated at a table that has a projection surface. During the group interaction, microphones capture the sound stream of all participants. The visualization of such flows is then projected, in real time, on a projection table.

The flow of each person sound captured has a different colour and is placed in a circular timeline in the form of rectangles (whose size depends on an average amplitude of speech). Smaller rectangles within these represent side conversations, comments, etc. Yet, it is worth noting that points represent times when no one speaks, giving a sense of continuity and that the system is active (Figure 9).
The circular timeline begins in the centre and goes clockwise. Each circle represents one minute of conversation. When a minute finishes, its circle is contracted the centre for that another may appear, removing in that way a proper causal notion in a long conversation.

2.2.4 Visualizing Remote Voice Conversations

Visualizes remote conversations through the volume, pitch and content. It gets to portray sarcasm, subtle clues and questions in a way that its transcription can't [19].

This project contains three types of display which are focused on:

- Conversation History: stores the colour of the participant who spoke on squares (of its colour) from left to right. When two people talk at the same time, the least noisy one is represented by a smaller square within the other square (Figure 10). With this, we can see who controls the conversation;

![Fig. 10: Visualizing Remote Voice Conversations Interface – History](image)

- Volume and Pitch of the Conversation: also with the colour of the participant, represents the volume and pitch of the conversation in a graphical form where the xx axis consists on the pitch and the yy axis consists on the volume. You can figure out which of the participants laughed more, or who speaks, in average, louder or lower (Figure 11).

![Fig. 11: Visualizing Remote Voice Conversations Interface – Volume and Pitch](image)
• Conversation Content: is saved by words taken from the talk in chronological circles. In every minute that passes words get away from the centre of the screen (Figure 12). Each time a word is repeated, the word moves to the centre and its size and brightness increases (losing the chronologic notion). If a word is said by more than one person, its colour starts to be a combination of the participant’s colour who spoke.

Fig. 12: Visualizing Remote Voice Conversations Interface – Content

2.2.5 Visualizing Conversation (Loom)

Loom [9] is a visualization tool for Usenet groups (users post text messages in forums that are grouped by subject). The aim is to provide a visual interface of the various groups and at the same time try to find existing social standards.

In the most basic setting of Loom dots represent individual posting (Figure 13a). This view reveals interesting patterns: we can easily sport most vociferous members of a group and can see patterns of activity, daily intervals vs. those whose participation is more irregular.

Fig. 13: Loom Interfaces – a) dots for posts; b) user posting traces; c) mood
Another setting of \textit{Loom} traces the connection between sequential posts in a thread (Figure 13b). Here lines connect the thread as it passes from person to person. This view reveals individual patterns. We can see stand alone messages. Some are simply comments or announcements; others are postings of users known for their persisting and annoying provocations. By using colour to highlight relevant patterns, we can easily see which groups are places of long, intricate, never-ending discussions and which are sites of quick exchanges. Silence, in the real world, is an important communicative device. In a text-based medium, silence can be too subtle to serve such purpose. By visualizing the conversational patterns, these elusive social cues can be seen.

The last setting is \textit{Loom} as a visualization of mood. It classifies groups and each message with angry, peaceful, informative or other flag. In the Figure 13c, red dots represent angry postings - clearly the predominant mood of this group. As a signature or portrait of the group, this image is quite striking - one can quickly ascertain that if disputation is not of interest, this is a place to avoid. Yet for understanding the interactions within this group, this classification scheme is too general. One incorporating gradations of anger would reveal more of the underlying patterns.

\subsection{2.2.6 Influencing Group Participation with a Shared Display}

The goal of this work is to build interfaces that help groups to improve their interaction processes and aim to encourage them to include a wider range of views in their discussions in order to promote a better decision [8].

The interface provides information through a shared display during the group discussions indicating how much each person participated in relation to other members. There is no content extracted through speech recognition, there is a simple detector that detects when a person speaks during the meeting. The application sets a different colour for each participant and demonstrates how one person participated through a bar chart. Also shows in a bar on the top, through circles with the corresponding colours, who spoke in the last 30 seconds (Figure 14).

It is considered that a person speaks, when it’s detected the sound level of one’s speech for over than 30 milliseconds (the sound level is calculated every 10 milliseconds and the speech sound level of a person are calibrated at the beginning of the meeting). This simple implementation quickly eliminates expressions like “oh” or “a”, among others.
This interface is intended to be a tool for personal reflection and also to impose acceptable group behaviour. To make this message more clearly, the bar chart also has lines of participation: "under" to below 12.5%, "participating" between 12.5% and 37.5% and "over" above 37.5%.

2.2.7 Designing Interfaces that Influences Groups Processes

The aim of this work is to build and evaluate collaborative tools that persuade behaviour change in a group of individuals. To achieve this purpose, this project has three specific research objectives: (1) build an interactive environment for deploying collaborative applications, (2) run quantitative assessments of group collaboration on environments of face to face meetings, and (3) establish a set of standards to build applications that assist groups with the task of improving the quality of their collaborations [6].

The first application presented is the Second Messenger [7], which increases the interaction face-to-face by presenting on a screen the verbal comments made during a discussion. It is therefore a combination that has the speech recognition and semantic analysis technologies to show in real-time a text summary of the comments in the group (Figure 15).

The presented words can be manipulated and arranged with a mouse. In addition, the Second Messenger filters the messages getting more words of people who speak less and less of those who talk more, providing a wider range of opinions. When a user speaks to its microphone the recognized speech is analysed and filtered. After filtering, the text appears at the top of the screen and begins to fall toward the centre of it.

The other interface features, through a histogram, the number of comments spoken by each participant (Figure 16). The objective is to understand if the behaviour of the group changes, if the interface does not remove the attention of the same. Through studies they have found that the most active users began to speak less, as less active users began to talk more. Another interesting point was that the group's performance in solving tasks was higher.
2.2.8 Visiphone

Visiphone [15] is a hearing conversation graphical interface designed to support seamless and ubiquitous connections between people in different locations.

Its main purpose is that these types of casual conversations are pleasant, very much like a conversation when one person is giving company to another and not when they are in discussion. Therefore, the solution is a visual telephone, one that grows and speech through graphs (Figure 17). Works like a normal phone but provides a graphical and dynamic rhythm of the conversation. It attracts the attention of the user to respond to the speaker; and it provides a focus for which the person must talk.

The Internet use makes the calls free and the fact that's a hands-free phone allows a conversation with a larger number of people. The Visiphone also has the advantage of being in motion while the connection is active, giving the indication of presence. People do not feel obliged to be always talking because they know that the call did not fall (but still do not know if someone is on the other side of it).

The main setback of the system is that, despite having been tested by hundreds of people in public places, bringing very positive reviews, there were no formal studies that could confirm its potential.

2.2.9 In Situ Speech Visualization in Real-Time

This paper presents three works of interactive art (Hidden Worlds, RE: MARK and Messa di Voce) [17] that seek to answer the question: “if we could see our speech, what might it look like?” As such, the discourse is designed to provide, in a distinct and plausible way, an interactive fictional universe in which the speech is visible.

Hidden Worlds is an interactive audiovisual installation (or an augmented reality system) that allows participants to see the voices of each other, which are made visible in the form of animated graphic pictures that emerge through the mouths of the participants when they speak. The figures are related to the amount of volume, pitch and timbre and can only be viewed through glasses that record the 3D graphics in the real world.
It's allowed the participation of up to six people and there is a preview pane of the figure shadows for people who do not have the glasses (Figure 18).

**Fig. 18: Hidden Worlds Interface**

*RE: MARK* is a small installation designed for only two participants. As in the *Hidden Worlds*, presents an interactive visualization of users' speech but instead to focus on the aspects of sound, focuses more on the symbolic domain that has been written or spoken.

The sounds are spoken for a couple of microphones and analysed through a system of phonemes recognition. If a phoneme is recognized (e.g. "oh, ah", "ee", etc.), this one is displayed. Otherwise, it’s projected a geometric form that conforms to the tone of voice. Projections are always made from the shadow of the participants head in the display (Figure 19).

Both interfaces presented are quite simple and have as target audience children up to 10 years of age. In order to do something more sophisticated and appreciated by all, the project *Messa di Voce* has born. It consists of a concert where the speech, shouts and songs produced by two vocalists are augmented in real time through interactive visualization software (Figure 20). The audio is analysed after being collected through the microphones of the singers and there is a wide range of different graphical visualizations (including particle systems, mesh of elastic springs, simulations of fluids and clouds, etc.).

**Fig. 19: RE:MARK Interface**

**Fig. 20: Messa di Voce Interface**
2.2.10 Conversation Votes

Conversation Votes [5, 16] is an extension of the Conversation Clock. As the last, through visualization provides indication of the domain, changing shifts, mimicry, and other aspects that allows people to make an assessment in the third person of their participation. In addition, the first provides each participant with two buttons, which allow good or bad feedback to be shown in the visualization. Participants may encourage current speaker giving positive votes to continue or negative votes to stop.

The visualization consists of a continuous indication of phonetic activity. Each person is represented by a colour and a bar when it speaks. The current history is presented in the form of a horizontal bar and the past is represented vertically (Figure 21). The system accommodates up to 4 people sitting at a round table.

The size and brightness of the bar increases or decreases depending on the votes. A vote, which is given anonymously, is viewed through a point and is an indicator of activity, containment or reference point in the conversation (Figure 22).

2.2.11 Egocentric Analysis and Visualization of Instant Messaging Activity

This work focuses on privacy issues using the communication of an individual with its contacts [1]. To do so, (a) analyse the social and semantic aspects of the communication; detect features such as frequency, dominance, density and longevity; and (b) build a compact structure where you can view all these resources.

The structure consists of a ring that has the individual at the centre represented in black. All its contacts are around it and are represented by garnet circles, whose size varies depending on the number of communications. Points on the radial lines from the centre of the ring (the individual) and the contact represent conversations between both (timeline) and the more brilliant they are; the greater is the conversation activity (Figure 23).
As shown in Figure 23, this type of visualization also allows a tag search in the conversations. When you do a search for a certain tag, the conversations that have it appear in the upper right, and the contacts on them are selected (its circle colour changes to blue).

Through the slider we can choose the days we want to visualize and through the timelines we can select a conversation to read (by clicking in one of the points of the radial lines). The Frequency is checked by the circles size of contacts and the existing holes in the timeline that connects the individual to its contact. The Dominance is represented by the percentage that the individual talks with the contact and the dominant words. This is displayed in the upper right corner when you select a contact. When there are many conversations in one day, the points of a timeline increase to the sides, forming rectangles, and through this, we can easily determine the Density. Longevity is seen through the use of the slider. As we increase the time interval, if a timeline continues to have points, it means that there are, for a long time, conversations with the contact.

Initial test study indicated that users liked the interface and that the chosen resources capture the key features of communication.
2.2.12 CrystalChat: Visualizing Personal Chat History

CrystalChat [22] focuses on social interaction centred on a person, supporting exploration of their own history. This program creates a visual interaction, which incorporates the current message, in a structure that integrates its chat patterns of interaction through a time track in its history. Combine the social networking and the temporal time in a 3D representation structure.

It aims to visualize the personal history of the chat and for this we defined the following design goals:

- Reveal the chat activities of the person using data available in the log;
- Make individual messages visible and keep the connection between its visual symbol and its textual content;
- Indicate the beginning and end of conversation (opening and closing the chat window);
- Represent interactions with different people in groups visually distinct;
- Provide a structure that connects the interactions with all people;
- Be possible to indicate time in order to be able to compare durations of conversations;
- And provide an indication of the emotional content of the dialogue through the emoticons used.

Currently, the CrystalChat uses the MSN Messenger logs made in .xml and that contains temporal, textual and activity (close and open window, etc.) information. If necessary, could be applied to any other chat program that provides this type of information.

To allow a coherent view of all these criteria, it was used a hub and spoke type structure (where the hub is the central person). When the structure is viewed from above, each spoke represents a different contact. These spokes are composed by circles, where each circle corresponds to a different conversation (Figure 24a). If the structure is viewed from the side, each spoke gives details about the conversations with that contact (Figure 24b). When this happens, there are several coloured circles. Each circle corresponds to a different message and, depending on the colour, to a certain person (to each person is assigned a different colour).

Fig. 24: CrystalChat Interface - (a) Top View; (b) Side View
We can explore every conversation in detail by selecting them. The conversation opens diagonally to keep the temporal order and to see what was written (Figure 25). The colour of the text has the same colour of the person who wrote it and how many more characters are written, the darker the colour matching.

In order to present the information in a more precisely temporal level, authors defined the temporal spacing mode. This mode of representation has a line for each existing conversation in one day, and if there aren’t any conversations has a blank line (Figure 26).

To improve the visualization of the structure, they use translucent colour plans. To give them an extra use, the conversations are arranged depending on their content (Figure 27).

When there are a large number of spokes and / or conversations, they begin to experience problems of concealment. To avoid these problems were defined filters. With these, conversations are filtered to a certain period of time. So, find out who is the person who starts conversations with greater frequency or find out the number and frequency of messages becomes a very simple task.
2.2.13 Human Semantic Interactions in Meetings

In this paper is proposed an approach for capture, recognition and visualization of human interactions [24]. They consider that participants interactions involved are consumed with semantic (i.e., user intention or attitude towards a topic) and are different from physical intentions (for example turn talking or addressing).

With this in mind, they developed a system which consists on the following three layers:

- **Interaction Capture**: consists in the physical layer that deals with capture environment, devices and methods. Video cameras, microphones and motion sensors, are used to record the meeting content and tracking participants’ head movement (Figure 28);

- **Interaction Recognition**: consists in the structural layer which analyses content extracted from the audio-visual and motion data. Speech recognition engine extracts the speech features from audio data. Motion Data Processor is responsible for analysing people’s head gestures. Annotator is used for label features from these audio-visual data.

- **Interaction Visualization**: consists in the presentation layer, offers an interactive user interface for browsing human interactions as well as meeting content (Figure 29).
It consists in seven rows where at the top is the timeline. Samples of the overall movie are presented in a second row. The following four rows represent the semantic interactions of the four participants. The rectangles represent interactions. Above the rectangle is named the type of interactions and that can be also identified by its colour. The longer the rectangle, the longer the participant talked. The dashed arrow between interaction rectangles means that one interaction triggered the other. This means that interactions without any arrow pointing to them indicate that they were spontaneous. Finally, a blue bar indicates the attention given by the remaining participants.

The visualization tool also provides zoom in and zoom out functions. It also provides a quick way to reproduce a segment of a video, for that you only need to select the segment and press the Play button. With this system, exploring how people interact can be used to enhance a variety of pervasive systems.

Fig. 29: Interaction Visualization – Meeting Browser overview
2.2.14 PrimaVista

PrimaVista [13] is a system that contains different visualization techniques to support a meeting. It makes automatic speech recognition with multi-sensory data captured from the meeting into a unified visualization interface that provides real time interactions and also browsing of past meetings.

PrimaVista’s design has four goals which were formulated on the basis of other design frameworks and through observation of current meeting practices:

1. Supports awareness through visualization;
2. Unobtrusive data collection;
3. Prevention of privacy issues;
4. And allow input from meeting participants.

The PrimaVista system uses wireless microphones hooked into an audio mixer to capture all the audio from the meeting room, which in turn feeds the audio streams. These streams are processed to calculate the audio level and then fed into the automatic speech recognition. This system only supports Finnish and all the participants have previously trained to give better recognition.

All the data collected is passed to the PrimaVista web service (composed by php scripts and jQuery javascript animations). After some tests, they decided to implement an interface that provides a Live View and a Past View (Figure 30).

![Fig. 30: PrimaVista Web Interface: Live View (Left) and Past View (Right)](image)

The Live View is continuously updated to reflect the current state of the system. The layout is divided into: the words pane (Figure 30 – 1) that contains the automatic speech recognition results; the comments pane (Figure 30 – 2) that contains comments made by participants; the input box (Figure 30 – 4) where the users insert their comments; and the user pane which shows the current logged ones. There’s also a drop-down menu (Figure 30 – 5) to select the speaker model (done by the previous train) and an indicator for the system status (Figure 30 – 6). In this view are only shown the last ten minutes of the recognized words.
The Past View allows browsing meeting data in more detail. The words and comment panes retain the same functionality as in the Live View, however an additional tag cloud visualization option is included in the words pane (Figure 31). The size of the word depends on the number of time that it was recognized.

**Fig. 31: Primavista Past View: Words Pane Tag Cloud**

The key difference to the Live View is the inclusion of an interactive timeline display at the top of the Past View. It provides automatic speech recognition results, user generated comments, audio data and a new eye gazer feature which indicate the number of people (eyes) following the presenter at a given time. Each vertical line maps one minute of a given time and the height of the bar represents the quantity of data in units per pixel. Finally there's an interactive slider that allows the user to skim through all the meeting.
2.3 Discussion

Throughout this survey, we describe various systems and their interfaces. In this section we will be comparing the various models mentioned and for this we have defined a set of criteria, related to what we want to have in the final work, which will be described below. It’s also presented a comparative table and is given an overview of the area.

2.3.1 Criteria Description

Number of People

The number of people who use the system at the same time affect the way that the visualization of the results is carried out. Some sort of identification may have to exist, for each person to be noticed (e.g. assign different colours for them).

Time Period

If a system analyses a conversation, and demonstrates it through a circular timeline, that increases every minute, this means that the system has a time period of one minute. If there's not a temporal analysis of this type and the system only runs one time, it is considered that the system runs "Once". That said, it is important to note that, as the SoundLog system will examine a range of time (either long or short), it is important to define one or several periods of time (e.g. minutes, hours, etc.).

Temporal Notion

Identify sounds, voices, etc. and analyse what was identified is the first part of the final work. After that comes the visualization of it. For all to make sense, and to be articulated in a coherent manner, there must be a temporal notion. This criterion, coupled with the type of view, helps us understand how the visualization system must be made.
Sound and Speech Identification

One of the expected results of the final work is to realize how low sound features (like frequency, volume or timbre) can give us a rich description of what surrounds us. Thus, the identification of sounds (e.g. aircraft, cars, music, noise, etc.) and of speech is a very important criterion that could not be neglected. In this criterion, there are some works that do not identify any of these types. However, they are described as producing interesting views and / or introduce important concepts.

Sound and Speech Visualization

It's the most important part of the final work and one of the most influential in the work presented. Since that is the criterion that has more dependencies, is closely linked with the previous because they determine how the visualization is made. Table 1 considers that there is visualization, when there is some kind of sounds and / or speech demonstration. With that exclude, for example, systems that determine rates of orality, systems that label identified sounds and / or speech and systems that cluster or distribute sets of words.

Visualization Type

We choose Visualization Type as a criterion to know the different types of visualization that exist and why they have been applied. It helps to understand what is needed for each situation in which the system is used.
<table>
<thead>
<tr>
<th>Number of People</th>
<th>Time Period</th>
<th>Temporal Notion</th>
<th>Sound and Speech Identification</th>
<th>Sound and Speech Visualization</th>
<th>Visualization Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoundSense</td>
<td>One</td>
<td>Once</td>
<td>Yes</td>
<td>Both</td>
<td>None</td>
</tr>
<tr>
<td>Noise at the Heathrow Airport</td>
<td>One</td>
<td>Two Minutes</td>
<td>Yes</td>
<td>Sound</td>
<td>Sound</td>
</tr>
<tr>
<td>NoiseSPY</td>
<td>One</td>
<td>Once</td>
<td>Yes</td>
<td>Sound</td>
<td>Sound</td>
</tr>
<tr>
<td>Chat Circles I &amp; II</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Talking in Circles &amp; Chatscape</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>None</td>
</tr>
<tr>
<td>Tele-Direction</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>Both</td>
<td>None</td>
</tr>
<tr>
<td>Conversation Clusters</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>None</td>
</tr>
<tr>
<td>Conversation Clock</td>
<td>At Most Four</td>
<td>One Minute</td>
<td>Yes</td>
<td>Speech</td>
<td>Speech</td>
</tr>
<tr>
<td>Visualizing Remote Voice Conversations</td>
<td>Several</td>
<td>One Minute</td>
<td>Yes</td>
<td>Speech</td>
<td>Speech</td>
</tr>
<tr>
<td>Loom</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Influencing Group Participation</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>None</td>
</tr>
<tr>
<td>Second Messenger</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>None</td>
</tr>
<tr>
<td>Visiphone</td>
<td>At Least Two</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>Speech</td>
</tr>
<tr>
<td>Hidden Worlds</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>Speech</td>
</tr>
<tr>
<td>RE:MARK</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>Speech</td>
</tr>
<tr>
<td>Messa di Voce</td>
<td>Two</td>
<td>Once</td>
<td>Yes</td>
<td>Both</td>
<td>Both</td>
</tr>
<tr>
<td>Conversation Votes</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>Speech</td>
</tr>
<tr>
<td>Egocentric System</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>CrystalChat</td>
<td>Several</td>
<td>Once</td>
<td>Yes</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Human Semantic Interactions in Meetings</td>
<td>At Most Four</td>
<td>Once</td>
<td>Yes</td>
<td>Speech</td>
<td>Speech</td>
</tr>
<tr>
<td>PrimaVista</td>
<td>Those who Trained</td>
<td>Once</td>
<td>Yes</td>
<td>Both</td>
<td>Both</td>
</tr>
</tbody>
</table>

Table 1: Comparison of the Described Work
### 2.3.2 Criteria Discussion

According to Table 1, the number of people depends heavily on whether the project is related to sound or conversation analysis. When there is simply sounds analysis, such as the work *SoundSense* and *Noise at the Heathrow Airport*, only one person is required to properly run the system. In the conversation analysis, whether oral or textual, although one person can run the system, you always need more people to obtain data / information to enable the proper execution of the system (e.g. execute the egocentric ring of the *Egocentric System* without conversations in the MSN Messenger history will not produce any final information). The same applies to our system. Whatever the scenario where it is applied, the system is being used, even indirectly, by more than one person. In a class will be the teacher and students; in a lecture the speaker and the listeners; and in a meeting will be the people on it. This implies that the system must be developed with the ability to distinguish people voice. It has to identify and recognize different voices.

Regarding the time period, this is a criterion that is closely connected with the temporal notion. All works that have a time period must necessarily have a temporal notion. Only the works *Noise at the Heathrow Airport, Conversation Clocks* and *Visualizing Remote Voice Conversations* have a given time period (respectively two, one and one minutes). Based on the scenarios that will be applied by the system, it was only decided to launch a time period of minutes if there isn’t a slide presentation. That is, as all scenarios usually have one or more slide shows, it was decided that these will be defined as the time period measure. The system will be divided according to the view time of each slide.

Although most of the works don't have a time period, this does not preclude having a temporal notion. At this point, all works have temporal notion, the notion that there is time and that it is running. This notion can be based only on a circle that increases when a person speaks and, over time, decreases (*Chat Circles I, Chat Circles II, Talking in Circles* and *ChatScape*), may also be based on a system where new data is popped when a person speaks (*Clusters Conversation, Conversation Clock* and *Conversation Votes*). It may also be based on works that have timestamps. This means that although the work may not have a clock or something similar implemented in the system, it has, in one way or another, the notion that time passed because different visualizations happen over the course of it (action → reaction). The analysis of this criterion and its related works led us to the implementation of a real time interactive visualization. The noise demonstration, the voice identification and the speech recognition are some of the points to be implemented in the system.

Passing to the Identification and consequent Sound and Speech Visualization, it is already evident that all the works that do not have any of those criteria are those that analyse textual conversations (*Chat Circles I, Chat Circles II, Loom, Egocentric System* and *CrystalChat*).
Through them, we conclude that not only just real-time visualization is important. There must be some mechanism for a posterior visualization. Thus, event registration (logs) must be made. With that in mind, a status report is created, at the end of the scenarios execution that provides information that could be compared and analysed in a later occasion. On the other hand, works that have the Identification part but lack at the Visualization consist on works: that only list the identified sound or speech (*SoundSense*); that apply effects on existing objects in the system (e.g. increasing or decreasing circles); that list identified words (*Conversation Clusters*); or orality percentages (*Influencing Group Participation*). From the works that identify Speech, only the *Conversation Clusters*, the *Remote Visualizing the Voice Conversations* and the *Second Messenger* use some kind of mechanism for speech recognition; the work *RE:MARK* recognizes phonemes. Nevertheless, we decided that the system must have the speech recognition mechanism because by combining it with the existing words on the slides, it’s a quick way to understand what happened in each scenario (e.g. subject taught in a class or demonstrated in a lecture and addressed topics in a meeting).

Although some works do not have Sound and Speech Visualization, it doesn’t mean that they don’t have a Visualization Type. This is proven by checking that all the above works have a given value on this criterion. Lists, Spheres, Maps, Projections, Graphics, etc. are some of the examples in the described work. Through these we can get a sense of what exists and we can find many ways do the Visualization part of the final work.

From the related work, we identified several interesting requirements for a sound-based analysis system. Voice identification can provide us with the information about who asked questions, who answered them and for how long all the intervenient spoke and for this an analysis of the low sound features frequency and timbre must be made. To gain context, we need to make a speech recognition mechanism and if we want an efficient one, we need to recognize the most important words. For that, we need to get them through the slides presented in the different scenarios (if the slides exist). To avoid the addressing of polemic or complicated themes in the future (both in public presentations and in meetings), it is important to analyse the level of noise during the scenarios. For all this to work, the system must have a real time interface and must produce reports at its end for future analyses. A nice way to show all the information obtained is by the use of distinct plugins like in *PrimaVista*, but having a better way to navigate through the information, using some pan and zoom commands like in *Human Semantic Interactions in Meetings*.

However, none of the described works is identical to what we intend to implement. None of them makes use of all the information that sound can provide (low sound features, voice identification and speech recognition) – usually they only use one of them. Also, despite most of the works have a real time interface; none have it applied to different scenarios. The status/past
report is never produced and in some works there’s the necessity to do training or use of devices.

2.4 State of the Art Summary

In this chapter we studied works that analyse the sound itself and works that analyse oral and written conversations. We also made a comparison between them and concluded that none of works do exactly what we intend. However, this study, allied to the goals of the system, helped us to achieve a set of requirements that we’ll explain in the next chapter.
Implementation

In the previous chapter we describe the state of the art, describing previous works that try to address the same problem that concerns us in this document. Those works allowed us to identify the proper scenarios on which we can use our application and also a list of requirements that systems using sound as a basis for contextual insights must fulfil. In this chapter, we’ll start by describing those scenarios and requirements, talking about what information needs to be captured, and how to visualize and analyse it.

After that we describe the overall architecture of the system and then we will speak specifically of each major part. This consists on a framework for information capture and an interface for visualization and analysis of that information.

3.1 Scenarios

The scenarios consist on a public presentation (class and lecture) and on a meeting. We decided to address these scenarios because we can extract enough information through our analysis, thus helping to improve them.

In the public presentation scenario, the important thing is to realize what parts of a presentation have generated more controversy and what were better perceived. We can use the volume analysis to determine complicated parts of a scenario and subjects identification to determine these parts. It is also useful to find out who made more questions and if the answers were long (for this we must have a voice identification mechanism). Another important point will be to discover what slides were more complicated to describe and at the same time try to figure out when there were delays in the presentation.

In the meeting scenario we must have an idea of what happened during a particular meeting, by identifying relevant topics of discussion. Having a sense of who’s talking about and how much time it spoke (or even speech percentage itself) it’s also relevant. That can be done with a combination of subjects identification and speech recognition,
3.2 Requirements

In this subsection we describe the system requirements, taking in mind the information obtained from the state of the art and all SoundLog system goals. We divide the system requirements into different categories (capture, visualization, analysis and others) and we provide a summary of them.

3.2.1 Capture

Part of our goal is to discover which information may be used from sound itself. To accomplish this, and according to the described works, we start by capturing the sound low feature characteristics: volume, frequency and timbre.

To get more information from sound, we need to discover who said it and what is said. For this, we need find a way to identify different voices from the different subjects and we need to use an automatic speech recognition engine that recognizes what is said. By doing this, we can easily discover who spoke during a meeting and calculate oral rates (since in a public presentation the oral speaker is the most intervenient one). We also contextualize all the scenarios though the different recognized words.

In order to get even more contextual and auto-biographic information, we need to go one step forward and capture information that comes from presentations. Information about what happened in the different slide presentation is a quick way to discover problematic slides (problematic parts of a class or lecture).

3.2.2 Visualization

So that we can see all the captured information, we first need to build an interface that shows all this data.

The interface also needs to be able to support the three different scenarios (class, lecture and meeting). This can be acquired by developing a different set of data visualization plugins.

Each plugin will demonstrate different data and then we can choose which plugins to select, accordingly to the type of scenario. All the information must be accurate and for that, the system needs to have temporal notion. Knowing what happened is important, but without knowing when happened it becomes useless.
3.2.3 Analysis

To improve the different types of scenarios, we need to get different tools to properly analyse what happened. But for that, we first need to save the data captured from the scenarios. The system needs to capture the information, pass the visualization to the different plugins (to visualize it in different ways) and finally save all the data to allow future analysis. For an easy and fast analysis, we need to access all the information from a scenario in a quick way.

3.2.4 Others

We also have to work in a way that no special device (besides the computer itself) is needed in order to not deflect the user and other participant's attention. By doing this we are capable to use our system in any situation.

To make it easy to install the application we intend to build it in such a way that will make it independent. Our desire is to build an independent information capture interface that allows users to use it anywhere and analyse it later on.

3.2.5 Requirements Summary

So, in order to build a reliable system, we need to implement all the requirements defined in table 2:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Capture</td>
<td>Sound Low Features</td>
</tr>
<tr>
<td>2</td>
<td>Capture</td>
<td>Voice Identification</td>
</tr>
<tr>
<td>3</td>
<td>Capture</td>
<td>Speech Recognition</td>
</tr>
<tr>
<td>4</td>
<td>Capture</td>
<td>Presentations Info</td>
</tr>
<tr>
<td>5</td>
<td>Visualization</td>
<td>Real-time capture</td>
</tr>
<tr>
<td>6</td>
<td>Visualization</td>
<td>Plugins</td>
</tr>
<tr>
<td>7</td>
<td>Visualization</td>
<td>Temporal notion</td>
</tr>
<tr>
<td>8</td>
<td>Analysis</td>
<td>Record everything</td>
</tr>
<tr>
<td>9</td>
<td>Analysis</td>
<td>Pan and Zoom commands</td>
</tr>
<tr>
<td>10</td>
<td>Analysis</td>
<td>Functionalities</td>
</tr>
<tr>
<td>11</td>
<td>Analysis</td>
<td>Status Report</td>
</tr>
<tr>
<td>12</td>
<td>Analysis</td>
<td>Plugins Combination</td>
</tr>
<tr>
<td>13</td>
<td>Others</td>
<td>No Training</td>
</tr>
<tr>
<td>14</td>
<td>Others</td>
<td>No Attached Devices</td>
</tr>
<tr>
<td>15</td>
<td>Others</td>
<td>Independent Application</td>
</tr>
</tbody>
</table>

Table 2: SoundLog system Requirements
3.3 System Overview

The solution that we propose assume the format of a C# Framework and a Python application that accomplishes the goals and requirements detailed in the previous chapter. These requirements can be divided in two types: the capture of information and the visualization and analysis of it. This separation eventually dictates the architectural decisions that we have taken.

So, in order to capture the information, we decided to implement a generic, reusable, Framework. This Framework (.dll) meets all the capture requirements previously described, was implemented in the C# language. Through it we can get the low sound features, make voice identification and speech recognition. In order to get info from presentation, we develop a PowerPoint add-in that provides us with images and textual content.

Afterwards, we develop the interface in the Python language. This interface uses the Framework to get the information and with some processing we can visualize and analyse it. The interface consists on different plugins where each of them demonstrates different types of information. We then can choose which plugins to use in each type of scenario (classes, lectures or meetings).

We had also always in mind, throughout the development of this system, that we wanted to implement an independent application that doesn’t needed any training or devices to work properly. These requirements are ones of most importance because all the systems that we describe that were trying to do something like we have done were always too dependent (programmatically or physically). This makes them impossible to use them in a real live class, lecture or presentation.

So, the SoundLog system consists in a generic Framework that captures information, along with the PowerPoint add-in, and in a python interface that provides an abstraction to the different scenarios (Figure 32).

![Fig. 32: SoundLog System](image)
3.4 Framework

After having defined the system requirements, we started by building a framework in the C# language. This framework was developed in order to contemplate all the capture requirements described. With the state of art, we concluded that the low sound feature characteristics that we have to calculate are the sound volume, frequency and timbre. Sound volume is important because it helps us to identify parts of the scenarios that were more complicated. Both frequency and timbre features are used to calculate subject’s identification.

3.4.1 Sound Capture

The first thing that we have done in the Framework was to capture the sound from the microphone and calculate the volume of it. For that, we decided to collect a 44.1 kHz sampling rate, with 16 bit depth and 2 channels giving a bit rate of 1411.2 Kbit/s. We decided to use this sampling rate because the high frequency limit of human hearing is about 20 kHz. Thus, and because of the sampling theorem\(^3\), the sampling rate must be twice the maximum frequency one wishes to reproduce; the sampling rate had to be at least 40 kHz\(^4\).

3.4.2 Low Sound Features

With the sound samples captured, we needed to find a way to calculate the sound volume (or sound level) in dB and the sound frequency in Hz. That was done by passing the discrete audio signal captured into a digital signal that can be properly analysed. To do this we had to apply a Fourier Transform algorithm. Since the normal Fourier Transform consists in an \(O(N^2)\) algorithm, we decided to use an optimized version of the algorithm called Fast Fourier Transform (FFT)\(^5\). For this part, we didn’t reinvent the wheel and decided to use the Math.NET Iridium – Numeric Foundation dll\(^6\).

The FFT algorithm returns a vector where each position has a certain value (amplitude). Walking along the vector, we calculate the positions that have the highest and lowest amplitude. The maximum amplitude obtained corresponds to the volume (in dB) and its position on the FFT vector, after applied an Hz scale (sampling rate / FFT vector length), corresponds to the

---


fundamental frequency (in Hz). We determine, at every moment, both volume and fundamental frequency. And we validated our calculations in an artistic way, by playing notes in a flute\(^7\).

### 3.4.3 Filters

Because both calculations are made around every 200ms, we continuously make their average. This happens because the interface captures information every second, and not every 200ms. So we defined a configurable range (one second by default), which returns the frequencies and volumes (maximum and minimum) obtained.

Since human voice frequency covers a range of 60 Hz to a peak in the 1-3 kHz region (though reaching frequencies up to 6000Hz), when we get volume and frequency from the Framework, we can provide a VOICE or NONVOICE argument. Thus, the calculation is limited up to 6000Hz in the first case, and is made from 6000Hz in the second (by limiting the Fast Fourier Transform vector). Through the fundamental frequency and the limits of the human voice, is it possible to identify whether someone spoke in a certain moment.

If needed, we can get the low sound features within another frequency range, the Framework has filtering mechanisms: low-pass, high-pass and band-pass filters. The filter system is pretty basic, is to annul the amplitude of the vector obtained at the locations that the filter indicates (e.g. in a 3000Hz low-pass, it annuls all frequencies below 3000Hz). With this, and taking the voice limits of the fundamental frequency as parameters, we can focus on what is said and remove background noise from other frequencies.

### 3.4.4 Record System

The Framework allows sound recording: with start, pause/resume and stop commands. The samples collected from the microphone are written to a temporary file and at the stop command; this file is converted into a .wav file. The user can select both the directory and the file name of the recorded sound. The recording is also available after filter application. For this, the Inverse Fast Fourier Transform algorithm\(^8\) is executed to transform digital sound into analogue sound. This does the exact opposite of the FFT algorithm. This is useful if we record sound in a loud place. If we apply a band-pass filter with the human voice frequency as limits, the voice becomes clearer.

---

\(^7\) Flute Notes Frequency Obtained from http://www.phys.unsw.edu.au/jw/flutes.v.clarinets.html#Z in November 2009

3.4.5 Voice Identification

With the low sound characteristics volume and frequency being captured, and knowing when a person speaks because of the frequency, our next step was to discover who spoke. For that we turned into another low sound features, the timbre. People can speak with the same frequency but the timbre usually is pretty different.

Through research, we identified that the most common way to obtain timbre in real time was to use the *Mel Frequency Cestrum Coefficients* algorithm [12]. Because we didn't found it in any framework, at least available for use, we decided to implement the algorithm. This was promising because the first step of the algorithm is to calculate the Fourier Transform, which is one step that we already have done. After that we had to apply triangular overlapping windows in the spectrum obtained in order to map its powers onto *Mel scale*. After some extra calculations, we obtain the timbre that consists in a large set of values (Figure 33).

![Fig. 33: Timbre of a drum note](image)

Since it’s not reasonable to compare the timbre large number of values, we apply the *Vector Quantization* algorithm [12] which reduces the size of this set to only sixteen values. Of the sixteen, only the largest of them needs to be considered because the other fifteen are too small and can be discarded. We consider this acquired value to be the timbre obtained in real time.

However, even if a person speaks for a while, the value is not always equal. For that reason, we implemented an algorithm that consists in the calculation of the last timbre values average. Through the last twenty values we make the average of the first ten and another average of the last ten. If the difference between them is less than the defined threshold (now equal to 1.5) we then compare it to the array of identified voices. We don't consider the other fifteen values because they are too small (lesser than 0.1 and bigger than -0.1).

The identified voices array contains the averages obtained that are different (not with a comparison threshold of 2) from the ones that already are in the array. So, in one hand, if the array is empty or the average obtained is different from the ones previous defined, a new voice is identified and the last ten timbre average value is considered to be the voice timbre. On the other hand, if the last ten timbre average is similar to one of the voices, that voice is identified. We did a lot of testing and concluded that these two threshold values are the correct ones. If we make them bigger, we may identify the voices more times but we also identified them incorrectly.
After identifying a voice, this is going into the identified voices array. This array is persisted and every time the framework is initialized the array is populated. We decided to save (persist) the array because the same user usually deals with the same voices. In that way, an identified voice in one scenario is already identified in the future scenarios. This is useful because we can label the identified voice in the Python interface. So, if one voice is labelled in one scenario (e.g. John), if its timbre is identified in future scenarios, we already have its voice label and John will instantly be identified.

It is crucial to identify different voices in any of the scenarios. In a public presentation we can determine who is speaking, how long it spoke, if other people are speaking (e.g. students in class or people in the lecture), who made more questions, if there were long responses, etc.; in a meeting we can compute in real time the percentage of speech of each person.

### 3.4.6 Framework Application Test

During the implementation of this framework, before building the necessary .dll we develop an application in the C# language (Figure 34). This application allowed us not only to test but also to refine the framework, in order to pass it to the scenarios in perfect conditions.

![SoundLog Application](image)

**Fig. 34: SoundLog Application developed to test and to refine the Framework**

The left panes of this application provide information about: sound low features Fundamental Frequency and Volume; identified voices and recognized words. Only the top pane was offering information at this time, the other two panes were used to test the VOICE and NONVOICE
parameters previously described. Below these panes, we can define an interval for each information capture. In this case, the information in the panes is updated every five seconds. It provides averages of the low features and the identified subjects and recognized words.

The timbre (that allows to discover who spoke), is displayed on the right pane. In this example we can see two different voices. As you can visualize, the timbre values aren't exactly equal for each voice, despite pretty similar, and this is the main reason for the implementation of an algorithm that reviews the last timbre averages.

The top right corner is the filtering system. We could apply high-pass, low-pass or band-pass filters defining the high or low limits with two different sliders. The bottom right corner consisted on the sound recording system. The recording and filtering system were allied in such a way that we could define a filter with the voice fundamental frequency limits to remove sound from different frequencies, thus getting a more clear sound of the voice.

3.4.7 Speech Recognition

Another important task is to discover what was said. The speech recognition has not been forgotten and this is done through the Microsoft Speech API (SAPI) 9. We built it to recognize the English language because Portuguese is still not available. We decided to use the Microsoft speech API because it's already available in Windows OS (Vista or above).

The speech recognition could be implemented by two methods: full recognition or select recognition. With the full recognition mode, the speech recognition tries to recognize everything that is said. It accesses a global dictionary to get the better recognition possible to a phrase. With select recognition, a grammar that consists in a set of words is passed to the speech recognition and these words are the only ones that are going to be recognized.

We made some tests without any speech training (it is possible to make the speech recognition application to better recognize what a person says if the person makes some speech recognition training). It was easy to understand that the full recognition system was not the method to choose. On the other hand, the select recognition method was pretty good. It recognized correctly everything that we said. So, we obviously decided to implement the second method, the select recognition.

---

9 Microsoft Speech API (SAPI) 5.3 Obtained from http://www.microsoft.com/speech/default.mspx in November 2009
To implement this method, we needed to build a grammar. Then, the SAPI context-free grammar compiler compiles the XML grammar into a binary grammar format. The compiled binary grammar is loaded into the SAPI run-time environment from a file, memory, or object (.DLL) resource and finally the words in the grammar begin to be recognized.

The file consists in an .xml that can be parsed by SAPI in order to be compiled and afterwards recognized (Figure 35).

In the GRAMMAR tag we specify that language id (409 means English).

```xml
<GRAMMAR LANGID="409">
  <RULE NAME="soundLog" TOLEVEL="ACTIVE">
    <OPT>
      <DICTATION MAX="INF"/>
    </OPT>
    <LIST>
      <P>sound</P>
      <P>speech</P>
      <P>voice</P>
    </LIST>
    <OPT>
      <DICTATION MAX="INF"/>
    </OPT>
  </RULE>
</GRAMMAR>
```

The RULE content describes what is to recognize. Many rules could be built and the TOLEVEL property defined the first to be used. In our case we just needed one.

In the LIST tag there are the words that we want to recognize. The words that would be recognized using the grammar described in Figure 35 are sound, speech and voice.

The OPT tag describes optional speech that can be said but is not recognized. So, if we say the phrase “I love this sound man!”, the word sound is recognized and the “I love this” part of the phrase will fall into the first OPT tag and the “man!” part of the phrase will fall into the second OPT tag.

We started to build grammars with more rules but when we discovered the OPT tag, we simplified the grammar and obtain the best recognition. This .xml file is build by the Python application. With the skeleton of the grammar file defined, we only needed to get the list of words to recognize. We started by getting the most frequent words that we encounter in the PowerPoint presentations but we had problems with the most common ones. We removed those words (the Portuguese and English ones) and also words with less than three characters and by doing it we got a nice set of words.

In the meeting scenarios, where usually there isn’t any presentation, we faced the problem of not obtaining any word to recognize. So, we defined a tool in the Python interface that collects all the textual content from different files and applies the same algorithm to find the set of words.
to recognize. Without this tool, scenarios without any presentation wouldn't have speech recognition.

With the Speech Recognition feature we can discover what is said, gaining contextual information for the different scenarios.

With the capture of the sounds information made and tested with the C# application, we transformed the application into a .dll that could be used by the interface built in Python language. The interface uses the Framework dll to get the captured information. Then, using different plugins, it allows the information visualization and analysis in different ways.

When we start the capture of information in the Python application, the framework is also initialized. Since the beginning of it the Speech Application is started (Figure 36). This application is always hidden and it’s only active when we pass a grammar to be recognized.

![Fig. 36: Speech Recognition Application](image)

During the heuristic evaluation tests, users complained about the Speech Recognition application and for that reason we decided to hide it (minimize) every time it starts during the Python application. This also prevents errors that could happen if users closed the application.

### 3.5 PowerPoint Add-In

Before we start the build of the interface, and to obtain more contextual information, we develop a PowerPoint add-in to collect information from presentations (Figure 37). We extract the textual content of the slides (plus notes) and also images of them. These images are then shown in the Python application every time that is made a slide transaction in the PowerPoint. Both the slide transactions and also the file presentation name are also provided to the Python interface. This is done by writing info in a file that the interface will read every second.
In the left of the figure we have the toolbar button. If we press the toolbar it will appear the SoundLog frame (middle). With \textit{Capture images} option selected, all the slides will be saved in a directory converted into images (that are used by Python). The \textit{Parse Recognition} option is to get the textual content of the presentation. All the presentation’ text is saved in the same directory in a .txt file. This file is then used by Python to create the .xml grammar for Speech Recognition.

The \textit{Progress} frame in the right shows the progress of the information captured from the presentation. The Progress frame appears when we press the \textit{Get Info} button of SoundLog frame. All the data captured from the presentation (slide images and textual content) is saved in a directory located in the $\%$Public$\%$ Windows global environment (usually C:\Users\Public\). If the directory doesn’t exist when we the interface is capturing information (through the framework), the Progress frame is automatically presented when we enter in the PowerPoint presentation \textit{Full Mode}.

We started by creating a directory with the same name of the presentation. This was wrong because we can have the same presentation name in two distinct presentations or we can change the presentation and that won’t be replicated in future scenarios. To pass this, whenever the presentation enters in full mode, we use an executable that we built to make the hash file of all the presentation (name and content). So, the directory created will have the hash result and different presentations (on name or location) will have different directory names. Since we might improve the presentations, it was crucial to distinguish presentations that had the same name but different content.

During any PowerPoint presentation, if the SoundLog system is running and capturing information, this add-in also provides information about slide transitions. All transitions are logged into a file that is parsed by SoundLog. The information provided when a transition occurs consist in the hashfile directory name, the presentation real name and the slide to which the transition was made. All this information is processed and afterwards visualized in the interface.
3.6 Python Interface (Scenarios)

In order to improve the scenarios, we must visualize the information that can be captured by both the framework and the PowerPoint add-in, in order to analyse it afterwards. So, the interface deals with the Visualization and Analysis requirements of the SoundLog system. Needless to say that will also deals with the Other requirements, which are present during all the code created. We started by getting the captured information, from the Framework or the PowerPoint Add-in available in the Python interface.

So, we need to get the captured information, we need to visualize it and finally we need to analyse it. This made that our python application was divided into three packages (Figure 38).

![Fig. 38: SoundLog Application Overall Architecture](image)

The Capture Package makes available to the interface all the information captured. This package is also responsible to provide the needed information to the Framework or the PowerPoint Add-in so that they could work properly and by making the data available for future analysis.

The Plugins Package is responsible by the visualization of that information and the Tools Package for its analysis.

3.6.1 Layout and System Flow

Before getting in touch with the different packages, it is better to give a general idea of the SoundLog system layout and flow. SoundLog layout is composed by a toolbar (Figure 39) which controls the system flow and contains different system analysis functionalities; and by the different plugins (Figure 40).
When we start the application, we might choose between creating a new scenario and loading a scenario (Figure 39a). When creating a scenario we have to choose its name, type (class, lecture or meeting) and if we use the previously identified subjects information (Figure 41).
When a voice is identified by the Framework, we can label it in the application. Both the label and the timbre are persisted and if we choose to activate the last choice, the previously identified/labelled subjects will be available. So, if a user that was labelled before is identified, it will appear its correct name.

Afterwards, we can start to capture information, by clicking on the Play button (Figure 39b). From this moment, and on every second, we get information about sound (in dB), identified subjects and recognized words, from the Framework. We collect the data using the Capture Package and then the data is spread to the different plugins, that process and display it.

After capturing the wanted information we end the scenario. Whenever we intend, we can load the scenario and analyse it. For that, we persist (both captured information and state of the application). The full name of the scenarios depends on the type of scenario, the name given when it was created and the time it was created. Thus, scenarios of the same type with the same name, will have to be made at different times, having different full names. To facilitate the loading of a scenario, the list with all existing scenarios can be sorted by type and also by date (Figure 42).
3.6.2 Capture Package

The Capture Package main functions are to get the information captured by the framework and in the presentations, to provide information to the Framework and to save the captured information for future analysis. So, we develop classes to perform these tasks (Figure 43).

![Capture Package Diagram](image)

Fig. 43: SoundLog Interface - Capture Package

To create an independent system we needed to create an interface that only depended on Python modules. If the interface only depends on these modules, we can create an executable which contains the necessary Python modules inside (and then we don’t need to install Python). With this in mind, we had to put aside the idea of using IronPython\(^{10}\) (an open-source implementation of the Python programming language which is tightly integrated with the .NET Framework). Iron Python would allow us to use the C# dll Framework without any complication. As we only wanted to use Python modules, we had to resort to the ctypes module. The problem of using this module is that it only supports calls to a C++ language dll (and not our C# language dll). We had to change our Framework so it became a Com InterOp dll. We also needed to develop a C++ dll (the one who is called by the Python) that consists in a Wrapper dll. This dll is called by the SoundLogDLL class and, depending on the method to call, it will call its correspondent in the c# dll (Figure 44).

![Python Communication Diagram](image)

Fig. 44: Python Communication with Framework (C# DLL)

To access the PowerPoint Add-in information was simply to access the files that are in the directory created for the presentation (the directory with the hash name). Every time that we are in a PowerPoint presentation full mode and change the slide: the directory name, the

---

\(^{10}\) IronPython Obtained from http://ironpython.net/ in December 2009
presentation name and the slide to which the transition was made, are written into a .txt file by the PowerPoint add-in.

When we start capturing information, this file is analysed every second by the ChangeSlidesAnalysis class, in order to see if occurred any transition. So, the PowerPoint add-in only writes in the .txt file when we are capturing information.

We built the DataCollection class to capture the information from the framework (using SoundLogDLL) and presentations (using ChangeSlidesAnalysis). This class collects the information and passes it to the plugins. All the information is passed in the same way, through Dictionaries python element where each key represents a distinct second of the scenario. Each plugin will then deal with that information in different ways, thus showing all the information provided.

As previously mentioned, we decided to use the select recognition method in Speech Recognition. To do that, we had to provide a grammar to be compiled. This grammar is built in the SpeechRecognition class. The hard part of this class wasn’t to write the .xml file, but choose what words need to be recognized.

To do this, we initially tried a simple method that soon proved to be sufficient. When a presentation is visualized during the capturing of information by SoundLog application, all the textual content of it is saved in a .txt file. We’ve done this to be able to select which words of the presentation needed to be recognized. We start by removing all the accents and then we make an insensitive case comparison between the words to get their frequency. We remove the most common words (English and Portuguese) and also words with length inferior of three characters. We then consider the top twenty words to be the ones to recognize.

Finally, the last but not the least important role of this package consists in save all the information captured during the scenarios. To do this, we implemented a system that persist all the data into .txt files. We use the Serialization class to save, in every second, all the information needed. The data is saved on files that can be loaded in the future. The serialization class is responsible by the serialization of the data and also by the unserialization of the data when a scenario is loaded.

The Capture Package also accomplishes capture requirements because it provides the SoundLog application, at every second, with all the collected information. Furthermore, it also provides the means to the collection of data to work properly and the means to analyse the information in the future (record everything requirement). By helping capturing information and doing the speech recognition grammar, this package deals with all types of requirement that we described in section 3.1.
3.6.3 Plugins Package

With the information being captured we needed to visualize the processed information. For that, we developed the Plugin class (Figure 45). All plugins derive from this class. The Plugins Package deals with the plugins and temporal notion system requirements of visualization type. It also provides the means for the information to be analysed.

We initially started by developing the Noise, Speech and Subjects plugins whose aims are to, respectively, provide information about sound level, recognized words and identified subjects (voices).

Later on we developed the Timeline plugin that provides temporal information and also the Slides plugin that demonstrates information about the slides of the PowerPoint presentations that are visualized during each scenario.

Fig. 45: SoundLog Interface - Plugins Package
The main class plugin provides a set of functionalities that are available for the different plugins. As the total information existing in a scenario can be huge (remember that a class usually takes 90 minutes), it was developed *pan and zoom operations*. We can execute them through the toolbar buttons (Figure 39c) or by direct manipulation of the plugins (drag and drop to pan operations and right or left mouse double click to zoom in or zoom out). Without these pan and zoom commands, it would be impossible to clearly analyse the captured information.

The zoom operation has four types of granularity. The system starts with the lowest one (one second), but we can then zoom out through the ten seconds, thirty seconds or one minute granularities. Both zoom and pan commands are only possible to make through direct manipulation on a part of the plugin (Figure 46a).

We might swap the plugins order dragging them in the blue area (Figure 46b) and use the Slides functionality in the green area (Figure 46c)

![Fig. 46: Plugin – a) Pan and Zoon (Red); b) Swap (Blue); c) Sliders (Green)](image)

The main plugin class also provides another functionality that is used mainly to analyse slides information: the *Sliders*. In the noise, subjects and words plugins, the sliders area (Figure 46c – green area) is composed with sliders limiters. By the manipulation of these sliders (we'll explain in later on how we do it), we restrict the slides that appear on the slides Plugin. This is the quickest way to discover different kinds of slides. For example, problematic slides are the noisiest ones; or discussed slides are the ones with most intervenient.

There is also displayed in all the plugins, the information about *slides transitions* (changing of a slide during the PowerPoint presentation), with a dotted grey vertical line (in Figure 47 there are five slide transitions).
In the subjects and words plugins, we can highlight a subject or a word by clicking on it. In Figure 47, we highlight the word `framework`. All the scenario instants that have that word recognized will be overlapped with translucent yellow rectangles (in all plugins) to ease its identification. The colour of the word (or subject) will also change into green, red or blue if it is, respectively, the most recognized the least recognized or none of the previous. By clicking on each word or subject we will highlight only the one clicked, if we click again we remove that highlight. If we want to make compositions of words or/and subjects, we can use the Highlight Tool, a utility provided by Tools Package that we’ll explain in the correspondent section.

Other visual functionalities like the plugin selection (change the colour of the plugin when we have the mouse over it and show additional information of it), the plugin name or the plugin size are also manipulated by this main class.

With some insight into plugins and their functionalities (pan and zoom operations; plugins order and size; and sliders and highlight functionalities) it is time to talk about each specific plugin. They are only concerned with what to display, how to possibly manipulate that information and how to control the sliders.
Concerning the Noise plugin, this is intended to demonstrate, at every moment, the volume in decibels (Figure 48).

When the plugin is selected (with the mouse over it), a red vertical bar is shown. It corresponds to a decibel scale that is used in the plugin. As the volume of the human voice is typically between 40 dB and 60 dB, we scaled this bar so it would have a greater range for this interval. Thus, values up to 40 dB are at the bottom of it and above 60 dB are at the top. This is possible to notice because we have small horizontal lines in every 10 dB (meaning that the one at the middle corresponds to 50 dB). This bar allows the user to scale the noise level throughout the scenario.

Regarding the zoom command, for every instant, it will show the highest noise level detected. In the case of the minute granularity, it is displayed the biggest noise level captured in the one second granularity (and the same goes to the ten and thirty seconds’ granularity). In the one second granularity, it is shown the sound that the Framework provides (and it is also the biggest detected since the last capture).

In the Sliders functionality, this plugin allows the user to set low and high levels of noise level. In the sliders we can manipulate two yellow lines that represent the interval on which all sound level average of a slide must be to appear in the Slides plugin. These lines cannot cross each other because we define a minimal range between them. They are continuous where we can drag them and dotted otherwise.

Subjects Plugin

In the Subjects plugin, we visualize the identified subjects in each instant. In Figure 49 we see the instants where the user André was identified.
When a subject is identified for the first time, a coloured horizontal line will appear in that instant. Behind that line is displayed the name of the subject whose label can be defined in the Subjects Tool. From that moment on, in every instant where the subject is identified, the horizontal line will be continuous, otherwise it will be dotted.

Depending on the height of the plugin, a different set of subjects may appear. If there are more subjects identified that we potentially can demonstrate, arrows will appear in the left of the plugin to see the other subjects.

In this plugin, we identify if a subject is identified through the Framework. When the zoom is different from one second, we make logic OR of the corresponding range in the smaller zoom. So, if Subject 1 is identified in the fiftieth second, in the smaller zoom granularity it will only appear in that second, but in the higher zoom granularity the line will appear continuous for that entire minute.

Concerning the Highlight functionality, we can select identified subjects and all the instants where that subject was identified will be highlighted. Through direct manipulation we can only make unique selections. If we want to make compositions of subjects we can access the Highlight Tool.

For the Sliders functionality, we use the counter at the right of the plugin. This counter is limited between zero and the maximum number of subjects identified during the scenario, in one slide. There are only visible the slides that have an equal number or more persons identified, than the defined in the Sliders Subjects counter.

Words Plugin

In the Words Plugin, we visualize the recognized words at each instant (Figure 50).

Like we previously explained, the recognized words consist in a set of words that are passed to the speech recognition. These words are provided by the presentation and/or by documents inserted in Documents Tool.
Because there may be a lot of words recognized in each instant with granularity bigger than one second, we defined that only the three most important words are displayed (when the instant has more than three recognized words). The importance rate is related with the words frequency in the provided files or presentations.

The colour of each recognized word will be white or black depending on the PowerPoint presentation number. For each presentation is defined a distinct sequential identifier. When this identifier is even, the word colour is white and black otherwise.

All these colours differ when we highlight one. Selecting a word will change its colour to green, red or blue if the word is in, respectively, the most recognized ones, the least recognized ones or none of the previous. In Figure 51, we used the Highlight Tool to select the fala and framework words. Fala is one of the most recognized words and Framework is in the middle set.

![Fig. 51: Plugin (Selected) and with Words fala and framework Highlighted](image)

The Sliders functionality is also present in the Words plugin. It works exactly like in the Subjects plugin but instead of subjects, deals with words. The counter is now limited by the maximum number of words recognized in one slide.

### Slides Plugin

The Slides plugin contains detailed information about what happened in the scenario, providing more contextual knowledge. As previously noticed, it consists in a plugin that interacts with other plugins through the Sliders functionality.

Whenever occur a slide transition in the PowerPoint presentation, a miniature image of the slide to which the transition went is visualized (Figure 52). Also the grey dotted vertical transition line is defined in all plugins.
Fig. 52: Slides Plugin (Selected)

According to the granularity, it may exist a lot of transitions in each instant. When we don’t have enough space to display all of them, we group them in the last miniature (Figure 53).

Fig. 53: Plugin (Selected) with more slides that can be visible per instant

If we click one of these miniatures, a middle size image of the slide, or set of slides, appears (Figure 54). The visualization of several slides is dynamic. The size of each middle image starts with the default one. If needed, the slides shrink in order to be able to see all of them.

Fig. 54: Group of Middle Size Slides Image

With the middle size slides image, we can access in the slides information by clicking on it. It’s then created a new frame with the slide image selected and with information about it: presentation name; number and appearance of slide; max noise level detected; identified subjects; and recognized words (Figure 55).
The use of the sliders is attached with the information obtained for each slide. This functionality is one of the most important of the SoundLog system. We can easily detect the most discussed, most relevant or most noisy slides. When we know which slides brought complications to a class, we can change them in order to make a better presentation. For example, if a slide is the noisiest one, that could mean that students weren’t paying attention. Or if a slide has a lot of identified subjects (most discussed) it could mean that students don’t understand it and have doubts about that part of the class. On the other hand it could also mean that students like that part of the class. Through the different plugins we can contextualize the situation and after that, provide the students with a better presentation.
**Timeline Plugin**

The **Timeline Plugin** is responsible by the temporal notion of the system (Figure 56). It provides information about time through two different timelines: one that indicates the actual time that we see displayed in the middle of the plugin (actual timeline); and another that provides information about where we stand in the presentation displayed in the bottom of the plugin (total timeline).

![Fig. 56: Timeline Plugin (Selected)](image)

The actual timeline contains time information in the last instant of the scenario and in every x instants. In the total timeline, a green rectangle is filled and placed over it representing what we are seeing, according to all the time of the presentation. Besides this, we can click over the total time timeline and we pan to the area that was clicked.

The total timeline also provides information about highlight instants. When we highlight something, all the instants that have that something are referenced in the timeline with a green vertical line (Figure 57).

![Fig. 57: Timeline Plugin (Selected and one word Highlighted)](image)

The timeline plugin provides a lot of temporal information. It provides the user with the temporal notion needed in this kind of systems.

**Plugins Summary**

All the described plugins display the information captured by the Framework. The Plugins Package is clearly most related with the visualization requirements of the system but it has some of the analysis requirements linked to it. The pan and zooms operations, who are crucial to analyse the system, are present in all the plugins. The plugins are also related between them. If we pan or zoom in one plugin, all of them pan or zoom. Also, the Timeline plugin is related to all the others and the Sliders plugin is mostly related with the noise, subjects and words plugins.
The plugin combination is then present. Some functionalities were also defined in them but they are deeper in the Tools Package.

According to the type of scenario, different plugins are visualized. For a class or lecture, all plugins are visualized except the Subjects plugin. We decided this way because usually it’s always the same person who talks in this type of scenarios. Already for the meeting scenario, we choose to hide Noise and Slides plugins because the information provided by them is irrelevant in this type of scenario. We may choose an initial different set of plugins for each type of scenario by changing a config file. Through it we can choose the plugins order, and which to hide.

3.6.4 Tools Package

This package is responsible by the analysis requirements. The different tools that were previously introduced are all part of this package. This package helps controlling the flux of the program with the New and Load Presentation Tools. It provides the means to analyse the system with the Highlight and Report Tools (functionality). The Documents and Subjects Tools allow to contextualize the information. The Plugins Tools provide supports the system plugins layout and finally the Help Tool provides tips about the entire system. Simply explained, this package assists the user in several aspects besides the analysis requirements.

Both New and Load Presentations Tools were explained in the Layout and System Flow section.
After the creation of a new scenario, and before we start the information capture, we can use the Documents Tool to add more information to the Speech Recognition. Before this tool, we only recognize the most frequent words taken from presentations. This was a problem especially in meetings because it’s not often to have presentations on them. To improve the speech recognition part of the system, we now can add documents that are parsed in order to build a grammar for Speech Recognition.

We can insert PDF files (.pdf), Word files (.doc and .docx), Excel files (.xls and .xlsx), PowerPoint files (.ppt, .pptx, .pps and .ppsx) and Plain Text files (.txt and .rtf). The text from these files is extracted into a .txt file and then we built a grammar (using the Speech Recognition class from Capture Package) that we send to the Speech Recognition application to parse.

With these files we aid the Speech Recognition and by doing that, we aid our system. It is a very important Tool because without it Speech Recognition wouldn’t be available in a scenario that doesn’t have presentations.

Concerning the functionalities that support the information analysis, we built the Highlight Tool and the Report Tool. The highlight is available in the words and subjects plugins through direct manipulation but there only a unique selection can be made. If we want to make multiple selection of words and/or subjects to highlight, we only can do it using the corresponding tool.

Highlight Tool

Through the Highlight Tool we can visualize all the identified subjects and all the recognized words and afterwards we can select the ones we want to highlight. There are also available options that allow us to select sets containing the most and least identified subjects and recognized words. As previously explained, the highlight is then propagated throughout all the plugins and the highlight instants are visualized in the timeline plugin.
The other tool built to analyse the system was the *Report Tool*. After the ending of a scenario, either when we stop the capture of information or when we load a scenario, we can access this tool that provides a summary of what happened during the scenario and has two visualization methods: textual report (Figure 59) and visual report (Figure 60).

The report provides information about: the maximum and minimum noise level detected; the top three identified subjects and the top three recognized words (both per number of times); and finally information about the slides: most and least discussed, relevant, noisy and longest.

We can access to a more attractive report built in html. By using both the highlight and report tools, we can quickly find information that leads to the improvement of the different scenarios.
Plugins Tool

Through a configuration file, we define the plugins order and the plugins that should appear or not for each type of scenario. Although that configuration is properly adequate to most of the scenarios that we find, sometimes users need to use different plugins from the ones that the configuration file dictates. For that reason, we built the Plugins Tool. We can define the plugins order and which to show or hide.

Subjects Tool

In order to give more context to the Subjects plugin, we needed to find a way to label the identified subjects. That is the main purpose of the Subjects Tool. The identified subject is provided by the Framework and we can label it. As related before, all this subjects’ information is persisted and if intended, we can use subjects identified in previous scenarios. So, if subject labelled John is identified in a previous scenario and we decide to use those subjects in the one we’re, if the subject corresponding John is identified, that information is readily available and we don’t need to label him again. This is good because the system normally used by the same person and we don’t need to be labelling him all the times.

Help Tool

After the end of the first prototype we made a heuristic evaluation. With that in mind we decided to build a Help Tool (Figure 61). Information about the system flow, plugins, functionalities, etc. is provided in such a way to give users the means to properly use the system.

![Fig. 61: SoundLog Help Tool](image-url)
3.7 Implementation Summary

In this chapter we explain SoundLog system implementation in order to accomplish its requirements. All the implementation described, and mainly the one corresponding to the SoundLog interface, was tested. So, all the information that was described in this chapter regards the final functional prototype and not one of its previous releases.
Evaluation Method

Here, we want to check if the SoundLog objectives were achieved and if the scenarios constructed perform what is intended.

The SoundLog system was developed in two iterations: first functional prototype and final prototype. When we ended the development of the first functional prototype, we made a heuristic evaluation of it and afterwards we developed the final functional prototype. When we ended the final prototype, we tested it in two types of scenarios (classes and meetings) in order to check if the SoundLog system really works.

4.1 Heuristic Evaluation

The goal of the heuristic evaluation was to determine errors in the prototype that went unnoticed. Thus, the heuristic evaluation summed up to the execution of four tasks by users skilled in the ten Jakob Nielsen heuristics\(^\text{11}\). The tasks were chosen in order to sweep the maximum of the prototype features and a guide (see annex) was created for this purpose. At its end, users answered a questionary to see if they liked the system but mainly to get more suggestions from them.

Through the data supplied we built table 3 containing the number of reported heuristic errors.

<table>
<thead>
<tr>
<th>Number of Errors</th>
<th>H2 -1</th>
<th>H2 -2</th>
<th>H2 -3</th>
<th>H2 -4</th>
<th>H2 -5</th>
<th>H2 -6</th>
<th>H2 -7</th>
<th>H2 -8</th>
<th>H2 -9</th>
<th>H2 -10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3: SoundLog Usability Heuristic Number of Errors Reported

\(^{11}\) The 10 Usability Heuristics Obtained from http://www.useit.com/papers/heuristic/heuristic_list.html in August 2010
The first tasks consisted in loading a scenario and identify the noisiest second. In order to make users work with pan and zoom operations, the scenario was saved with the biggest granularity (one minute). Users had to zoom in to the lowest granularity and pan to find the pretended instant.

Only one of the users completed the task this way. One of the reasons was that when we were at the maximum granularity, users only saw the initial time. This happened because in the first functional prototype, time was delivered every x seconds. In the case of the maximum granularity, those x seconds had still not passed and we only saw the initial time. Users pointed the heuristic H2-1 to this situation. In order to solve this problem, we decided to always put the time in the last instant of each scenario.

So, one of the users answered incorrectly four seconds (due to the granularity) and the other three answered correctly. Still, only one used pan and zoom operations, the others used the Report Tool. In these last two cases, users complained because there wasn’t a perfect metaphor for the icons displayed and therefore the heuristic H2-2 was pointed. They also pointed the fact that the icons didn’t possess any label and once again the heuristic H2-1 was selected.

In the second task, users had to identify the most recognized word, confirm it with the result of the Report Tool and Highlight it. Two users ended the task this way and detected the same problem of the icons label (heuristic H2-1). One of the users clicked the word one time to highlight it and then clicked again and nothing happened. He expected that the highlight would be removed but it didn’t, and for that reason he pointed the H2-4 heuristic.

In the third task the goal was to label the users voice. For that, users had to create a new scenario and speak till its voice is identified. One of the users pointed the fact that there wasn’t any type of scenario chosen by default indicating a H2-4 heuristic error. The same heuristic was indicated when the Apply button of the Subjects Tool didn’t work in the usual way (changes only were made only when the user clicked the Ok button). Another pointed heuristic was the H2-3 because we can’t access to the Subjects Tool by direct manipulation of the correspondent plugin.

In the fourth and last task, the goal was to capture information from a presentation. Users started capturing information with SoundLog and afterwards they had to read the contents of the presentation. At the end of the presentation, they used the Sliders functionality to achieve information about slides.
This task was the most complex one and users pointed heuristic errors in the Sliders functionality. Two users pointed that the Help Tool should be better in this case (heuristic H2-10). One pointed that the Sliders Functionality wasn't intuitive (heuristic H2-2) and that the same was different from plugin to plugin (heuristic H2-7). Finally, one of the users closed the speech recognition application and SoundLog system crashed. The H2-5 was pointed for that reason.

This heuristic evaluation proven to be more useful that we initially expected. Through it, we detected different errors:

- Inexistent label in the toolbar icons;
- Display of the final time instant of each scenario;
- Click on one word highlight it, then click again should remove the highlight;
- A type of scenario must be selected by default;
- Correction of the Apply button in the Subjects Tool;
- Hide the speech recognition application in order to not crash the application;
- And improve the Help Tool.

All the errors existed in the first functional prototype and were fixed till the conclusion of the final functional prototype.

4.2 Prototype Evaluation

All the work done so far, helped us to identify the major priorities of the system. In order to conclude if the system really works and if it can provide correct information we first define the issues that must be evaluated. So, we need to see if the system can be used in a non intrusive way, if we can log all the information captured and if that information is well captured: sound level, voice identification, speech recognition and slide management.

4.2.1 Investigation Questions

With these tests we want to answer to the following questions, which became the goal of our application:

- P1: The contextual and sound information extracted is enough to analyse the scenarios?
- P2: Can we log the entire scenario information to proper analyse it later on?
- P3: The information is correctly captured and processed?
- P4: Does SoundLog help to improve the scenarios?
4.2.2 Participants Profile

Due to the goals that we want to achieve in the different scenarios, we decided to make the prototype evaluation in two distinct phases. For that, we first tested it on public presentations by using the system in one live class. Then, we tested the system on two real meetings.

The participants are all males with age between 25 and 49 years old. They have a Degree or Master in Information Systems and Computer Engineering and experience in the situations that were tests (Table 4).

<table>
<thead>
<tr>
<th>Sex</th>
<th>Age</th>
<th>Test</th>
<th>Education</th>
<th>Experience in the Tested Situation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>34</td>
<td>Class</td>
<td>Master in IT</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>25</td>
<td>Meeting 1</td>
<td>Degree in IT</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>26</td>
<td>Meeting 1</td>
<td>Master in IT</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>25</td>
<td>Meeting 1</td>
<td>Degree in IT</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>23</td>
<td>Meeting 2</td>
<td>Degree in IT</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>49</td>
<td>Meeting 2</td>
<td>Master in IT</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4: Participant Profile

4.2.3 Location

So, we had a total of six participants, one in the class test, three in the first meeting test (both tests took place at Instituto Superior Técnico – IST) and more two at the meeting tests in BES ESData (Figure 62).

We concluded that it wasn’t necessary test the system on lectures because it has the same goals of a class (Table 5).
4.2.4 Procedure

The tests started with the recording of all the contextual and sound data provided in the different scenarios. For that, the participants had to run the SoundLog application, create a scenario and start the information capture. In the end of the scenario (class or meeting) they stop the application. Later on, the participants realize a number of tasks (see annex) and answered to a satisfaction questionary. Although we have made few tests, we have measured all the time spent in each task.

We also recorded the audio data from the scenarios, to cross-analyse it with the data that SoundLog provides. In all tests we needed to see if the system could be used in a non-intrusive way and if the data collected was enough to analyse the scenario and to improve it afterwards.

Despite that, and because we had two distinct types of tests, the information that we needed to test in each was different. In the class scenarios, our goals were focused in the noise, speech and slides plugin. We needed to see if the sound level, words recognized and slides were correctly calculated and visualized in the system. As to the meeting scenarios, we also tested the system speech plugin but our main goal was to test the subjects plugin. We wanted to see if the different voices were correctly identified.

So, in table 6 is provided the different attribute that we have tested in each type of scenario.
4.2.5 Results

In this section, we talk about the results obtained in the two phases of the prototype test. We'll dispose them according to the attribute that we wanted to test.

Noise

The noise plugin provides the sound level in every instant of a scenario. The sound level is calculated by the Framework in every 200ms and when the Python interface asks for it, it is provided the biggest calculated since the last time we asked for it. The sound volume helps to identify complicated parts in all the scenarios.

Through the analysis of the recorded data we could clearly identify that when the noise level of the class or meeting was higher (when participants talked), the noise level went up and when they didn’t stayed down.

In order to analyse this data with more efficiency, we decided to make another test (an independent one). In this test we decided to use the SoundLog system to capture information. In every two seconds we snapped our fingers in order to see if the system calculated the noise from the snaps. This test was made in a controlled environment where there was no other noise evolved and occurred during twenty seconds. The results from the noise plugin are elucidative (Figure 63).

Fig. 63: SoundLog Noise Plugin Testing (Snaps every Two seconds)
The finger snaps were correctly calculated in every second and with a dB level average of about 30dB, which means that has a good accuracy.

Both figures 64 and 65 also prove that sound level is properly calculated. In the two tests, the sound level calculated was between the 40 and 60 decibels of the human voice.

With the independent test that we took, and allied to the sound level captured in the different scenarios, we conclude that the noise was calculated correctly both in time and in level.

**Speech**

Taking in consideration that we needed to test the speech recognition where we have documents to build a grammar to aid the speech recognition application, we purposely used the class scenario to test this system attribute.

Using the PowerPoint presentations, SoundLog built the grammar and then provided it to the Speech Recognition application. Using the most frequent words of the presentations and removing the ones with less than three characters, we can identify the most significant words of the class.

During the entire class the speech recognition only identified the word *não*. This was a poor result but, unfortunately, one that we already expected. And there are several reasons for that to happen.
The Speech Recognition, besides being a technology that still needs a lot of improvement, in our case, is associated with the English language. So, recognize a Portuguese word becomes quite hard.

So, in order to properly test our system speech recognition, we decided to do another independent test. This is different from the noise level independent test, because that was made to provide more useful information, in order to properly analyse the sound level capture, and this one is made to provide all the information.

To do the test we created a PowerPoint presentation with ten slides. Each slide as only one of the following words:

- Sound, capture, framework, real, time, interface, noise, speech, subjects and slides.

Then, we've read all the slides but decreased the duration on each slide transition by one second, starting with five seconds and ending with one second per slide.

Fig. 66: Words Recognized with in different Slide Transition Intervals (5, 3 and 1 seconds)

For each interval we tested the system speech recognition five times and the recognized words are displayed in Figure 66. The ratios for each interval are displayed in the following graphic (Figure 67).
As told before, we decided to use the grammar mode to improve the words recognition. Nowadays, the speech recognition engines, at least the one that we can access free of charges, are still very inefficient when they try to recognize everything that is said. Another way to improve the recognition is to do a lot of training by reading for the computer’s speech recognition application.

Although we couldn’t test the words recognition in a real situation, with our test we determinate that the probability of an important word to be recognized when spoken after another important one, is bigger than eighty percent when the range between them is higher than three seconds.

Subjects

The identification of different subjects consists on one of the most important attributes in a meeting. In public presentations the oral speaker is the most intervenient, but in meetings there are many people and by simply calculating the oral taxes of them, we can improve the meeting.

We tested this attribute during the meetings tests. The first meeting had three participants and the second meeting had two participants.

The first meeting occurred at Instituto Superior Técnico, it had three participants and lasted around sixteen minutes. During that time there were identified three subjects. The second meeting lasted twenty minutes and took place at BES ESData building in Carnaxide. This meeting had two participants but there were identified three.
In Figure 68 we can see the number of time that each subject was identified.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Identified 567 times</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Subject 0</td>
</tr>
<tr>
<td></td>
<td>Identified 547 times</td>
</tr>
<tr>
<td></td>
<td>- Subject 1</td>
</tr>
<tr>
<td></td>
<td>Identified 8 times</td>
</tr>
<tr>
<td></td>
<td>- Subject 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Identified 532 times</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Subject 0</td>
</tr>
<tr>
<td></td>
<td>Identified 573 times</td>
</tr>
<tr>
<td></td>
<td>- Subject 1</td>
</tr>
<tr>
<td></td>
<td>Identified 1 times</td>
</tr>
<tr>
<td></td>
<td>- Subject 2</td>
</tr>
</tbody>
</table>

**Fig. 68: Subjects Identified in Meeting Tests 1 and 2**

In order to avoid lots of miss recognitions, we built an algorithm that only identifies one subject when its timbre is detected during several seconds on a row. This helps with the correct identification of one subject. In other words, when a subject is identified, it is correctly identified. Although, for a subject to be identified for the first time it takes some seconds of talking.

In the first meeting we had three participants but most of the talk was made by only two. After their identification in the very first seconds of the meeting, the third participant, the one who talked less time, it was only identified at the end of the meeting, despite talking brief periods of time during all of it.

In the second meeting, there was miss recognition in one instant defining Subject 2 appearance. Besides this instant, the others went quite well and the subjects were correctly identified. Both subjects were identified in the first ten seconds and since then correctly identified during the meeting.

To build a system that doesn’t needed any type of training or device attached to the different participants; we had to decide between good or lots of identifications.

Analysing the sound data recorded from the two meetings we concluded that participants were correctly identified over eighty percent of the time (when a participant was identified, it corresponded to the participant who spoke) but they were only identified sixty percent of the times (Figure 69).
In this plugin our main goal is to reflect the PowerPoint presentation transitions by displaying the image of the slide to which the transition was made. We also need to associate the information captured during each slide to the slide itself.

It’s in this plugin that enters the PowerPoint add-in. When a slide transition occurs, it writes on a file that is being processed by the Python Capture Package at every second. The information to put in the slide correspond to the one captured.

All the slide transitions were correctly parsed and recorded. The information presented in each slide also corresponded with the information displayed for that interval.

The temporal notion is provided by the timeline plugin. We decided to make a test to see if the time calculated correspond with the time that passed. For that, we started the SoundLog application one second after starting the Sound Recorder Windows Tool (Figure 70).
In a test that took near twenty seven minutes, we lost four seconds for the sound recorder tool. Despite that, this means that we can lose a maximum of seven seconds in one hour. Something like this it’s definitely not critical in our system.

Analysis

After the capture of information, the participants executed different tasks. In order to find the information needed to complete each task, the participants needed to analyse the data displayed in the scenario.

During the tasks, the participants mostly used the pan and zoom operations, and also the sliders and report functionalities.

Intrusion

In the meeting tests, after an initial phase where participants toyed with the application to see if it really works, they executed the meeting as usual.

In the class test, the participant started the information capture at the beginning of the class and ended it at its end. During the entire class the participant used several PowerPoint presentations and SoundLog didn’t interfere.

The system was used as intended, capturing the information in the different situations. The possibility to minimize the application while it’s running and the fact that all tests occurred in real situations, where there’s no time to waste, turned this attribute in the easiest one to achieve. The participants’ attention was never diverted.
4.2.6 Discussion

In the beginning of this section we defined some investigation questions that consisted in the different contributions and goals of the system. Now, we are going to answer them.

- **P1**: The contextual and sound information extracted is enough to analyse the scenarios?

In a public presentation (class or lecture), we want to discover which parts were more controversial, more hard to explain and where delays occurred. With the contextual information provided by the slides and by the speech recognition, we can detect parts that have taken long times to explain. Allying to the sound level calculation, we can determinate the most controversial parts and define the exact time through the temporal notion that exists. We can determinate if there were many questions through the voice identification. The voice identification is also used in the meetings to determinate the oral rates of each participant and the speech recognition to discriminate between different topics of the meeting.

- **P2**: Can we log the entire scenario information to proper analyse it later on?

All tests occurred in real situations. The different participants only could use the system later on in order to make the tasks assigned to them. So, to load the scenario later on, we needed to persist it. From time to time, and at the end of each scenario, all the needed information is saved and can be loaded afterwards. Besides characteristics like the name and condition (plugin and subjects management) of the scenario, the information persisted consists in the captured one. After loading a scenario, the information is passed to the different plugins and they restore the system as it ended. Without this feature, all the work would be in vain.

- **P3**: The information is correctly captured and processed?

As previously mentioned, the information capture is working very well. We calculate correct values for the sound level and voice identification. Despite the tests have been made in Portuguese, we concluded, through the independent speech recognition test, that a three second interval between important words is the biggest range needed to recognize both of them. And the information from slides and temporal notion is nearly flawless. Overall, SoundLog is a system that properly captures the sound and contextual information during the different scenarios.
• P4: Does SoundLog helps to improve the scenarios?

At the end of all tasks we presented the participant with a satisfactory questionary. Table 7 presents the obtained results.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Captured Information</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3.67</td>
</tr>
<tr>
<td>Visualization (Plugins)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3.67</td>
</tr>
<tr>
<td>Functionalities</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Tools</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3.5</td>
</tr>
<tr>
<td>Overall</td>
<td>4,25</td>
<td>3.75</td>
<td>3.75</td>
<td>3.25</td>
<td>3.5</td>
<td>3.75</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Table 7: Satisfaction Questionary Results per Participant – from 5 (best) to 1 (worst)

Associating the different functions of the prototype we got an overall result of 3,8 (in five).

4.3 Evaluation Summary

We are satisfied with the results obtained in the information capture, in the analysis tools and in the participants’ comments. However, we believe that there could be improvements, especially in terms of identification of different subjects. The increase of identified persons, together with proper identification, would improve both the general results obtained and, consequently, would facilitate the improvement of the scenarios.
Conclusion

In this chapter we present and discuss the final conclusions and contributions of our work, reflecting on the new problems that arose and that may be addressed next.

5.1 Master Thesis Summary

In this master thesis, we presented a new way to capture contextual and sound information in order to gain insight from different situations (class, lectures or meetings) and with the purpose to improve those situations.

Through analysis and comparison of different studies made in the second chapter, we conclude that none of the works presented does exactly what is intended. In the third chapter we present our system implementation. We start by describing its requirements and afterwards the scenarios on which we can use our application. We discuss about what information needs to be captured, and how to visualize and analyse it. After that we describe the overall architecture of the system and we speak specifically of each major part. This consists on a framework and a PowerPoint add-in for sound and contextual information capture and an interface for visualization an analysis of that information.

In the next chapter, we describe the different evaluations that our prototypes went through. We present brief results from the heuristic evaluation that derived from the first functional prototype. We then focus on the final prototype evaluation explaining how it took place, where it took place and making an analysis of the results obtained from it. We made class and meeting tests in order to see if the system could really be used to improve the scenarios. We concluded that the information captured and visualized in the different plugins was more than enough to achieve this goal.
5.2 Final Conclusions

Despite having speech recognition, our motivation passed by detecting what relevant information could we extract from different scenarios using the sound itself. Through the analysis of different works, we discovered that the most relevant sound features that we needed to achieve, in order to gain insights from the different scenarios, were the ones that allow us to calculate the sound level, identify different subjects (identify who said) and recognize different words (recognize what was said). Using the FFT algorithm and some extra calculations, we got the sound volume and frequency. This low sound features helped us to detect complicated parts of the scenarios because those parts have a higher sound volume. To identify different subjects (voices) we decided to calculate the timbre using the MFFC. We then applied the VQ algorithm to reduce the timbre set to just one value. From the two meeting tests (with 2 and 3 people) we got 80% success accuracy in the 60% times that we got subject identifications. We consider these results the weakness of our system and an improvement on these algorithms can be made. However, we still can calculate orality taxes and detect controversial parts from the scenarios using the obtained results. Regarding the Speech Recognition, we decided to only recognize a set of words that we provide. These words are the most frequent from the PowerPoint presentations visualized during each scenario and from different files provided by the users of the system. The recognition is made for the English language and, if there’s the intention of use it in Portuguese scenarios, an improvement in the Speech Recognition engine must be made. This functionality provides contextual information. It’s important to know what we are talking about. All these information is captured by a Framework that consists in a c# dll.

To get more contextual information, we decided to make a PowerPoint add-in that provides more contextual information. We get the images from the slides, the slides transitions and also all the textual information (used by the speech recognition). All this data is saved in a directory whose name consists in the hashfile of all the presentation. By doing this, we save all the information whenever we have different presentations content, name or location.

We then created a Python interface (out prototype) that used the information captured by both the Framework and the PowerPoint Add-in. To call the Framework methods, we had to make a Wrapper C++ dll. We’ve created this dll because python only calls c++ dlls. In order to call a c# dll we had to use a different implementation of Python (like IronPython) and by doing it we couldn’t build an independent executable file. We can visualize all the information through different plugins: noise, words, subjects, slides and timeline. We decided to build a different plugin for each type of information that we capture and can be analysed to obtain results. To ease the analysis of all the information we also built a different set of functionalities and tools: pan, zoom, highlight, sliders and report. We can do this analysis whenever we intend because the scenarios state and information captured is saved.
5.3 Future Work

Although we got satisfactory results from the evaluation of the prototype, we also found that there is space to improve both speech recognition and subjects identification. In the first case, we might find a way to recognize the Portuguese language and get the truly most important words from the data presented. In the second case, the identification of different subjects works properly because identifies correctly the ones who spoke. However, an improvement in the algorithm might occur in order to make a bigger number of identifications.

With this document we concluded that we can improve different scenarios. Another feature that might be useful is a way to compare different scenarios. So, if a professor uses the SoundLog application in different classes about the same theme, he might want to compare them to see if one was better than the other (after using the system).
References


[13] Heimonen, Ovaska, T., Turunen, S., Hakulinen, Rajaniemi, J., Rå iihä, J. & Kari-Jouko (July 2010). *Visualization of Multi-Sensory Meeting Information to Support Awareness (PrimaVista)*, In Information Visualisation (IV), 2010 14th International Conference, At London, United Kingdom, ISSN 1550-6037


7

Annex

7.1 SoundLog Evaluation

7.1.1 The Ten Usability Heuristics – Jakob Nielsen, 1994

H2-1 – Tornar o estado do sistema visível

Visibility of system status

H2-2 – Falar a linguagem do utilizador

Match between system and the real world

H2-3 – Utilizador controla e exerce livre arbítrio

User control and freedom

H2-4 – Consistência e aderência a normas

Consistency and standards

H2-5 – Evitar erros

Error prevention

H2-6 – Reconhecimento em vez de lembrança

Recognition rather than recall

H2-7 – Flexibilidade e eficiência

Flexibility and efficiency of use

H2-8 – Desenho de ecrã estético e minimalista

Aesthetic and minimalist design

H2-9 – Ajudar o utilizador a reconhecer, diagnosticar e recuperar erros

Help users recognize, diagnose, and recover from errors

H2-10 – Dar ajuda e documentação

Help and documentation
7.1.2 Heuristic Evaluation Guide

SoundLog Guide

Antes de mais, quero agradecer pela sua disponibilidade para a realização destes testes. Tradicionalmente, o som é usado pelo seu conteúdo, procurando-se usar técnicas como de reconhecimento de fala para perceber o que está a ser dito. Mas o próprio som, esquecendo a sua semântica, é capaz de fornecer informação. Por exemplo, no decorrer de uma aula, se o nível de ruído aumenta isso pode indicar que algum slide gerou comentários, ou que os alunos perderam o interesse. Isto pode ser usado para criar automaticamente um log do interesse dos alunos nas várias partes da matéria tendo em vista uma melhoria da aula nas partes em que esta é mais necessária. Numa reunião, pode indicar quando foi referido algum aspecto polémico, etc. Com este propósito em mente, desenvolvemos o sistema SoundLog. Este sistema captura informação em tempo real para diferentes cenários (apresentações públicas e reuniões) e permite uma análise posterior dessa informação.

Com este teste pretendemos então avaliar a interface do sistema SoundLog. Pedimos assim a sua máxima atenção durante a execução do mesmo. Depois de concluídas as tarefas será fornecido um pequeno questionário com observações onde deverá colocar todos os aspectos que não concorda, ou que gostou menos.

SoundLog

Heuristic Evaluation Tasks

1ª TAREFA: CARREGAR CENÁRIO E IDENTIFICAR O INSTANTE MAIS BARULHENTO
CARREGAR O CENÁRIO LECTURE XPTO e detectar o instante (segundo) mais barulhento.

2ª TAREFA: CARREGAR UM CENÁRIO E USAR AS FUNÇONALIDADES HIGHLIGHT E REPORT
Encontrar a palavra mais usada durante o cenário CLASS IST, fazer HIGHLIGHT da mesma e comparar com a informação dada no REPORT.

3ª TAREFA: CRIAR UMA REUNIÃO E IDENTIFICAR E RENOMEAR A SUA VOZ
Criar o cenário do tipo MEETING com o nome OLARI e falar até a sua voz ser identificada. Por fim mudar o nome da voz identificada para o seu.

4ª TAREFA: CRIAR UMA APRESENTAÇÃO, CORRER POWERPOINT E IDENTIFICAR SLIDE COM USO DE SLIDERS
Criar o cenário do tipo CLASS com o nome SLIDE e iniciar a captura de informação. Abrir o powepoint SOUNDLOG.PPT e ler os conteúdos dos slides. Parar a captura de informação e identificar o slide com maior número de palavras reconhecidas usando SLIDERS.
Heuristic Evaluation Questionary

Com este questionário pretende-se compreender se os utilizadores gostaram de usar o sistema SoundLog e se possuem alguma indicação em como melhorar o mesmo. Este questionário é anónimo e os dados serão tratados com confidencialidade. Muito obrigado pela disponibilidade para responder a este questionário.

1. Concorda com os diferentes tipos de cenários propostos?
   a. Sim
   b. Não. Porquê: _________________________________________

2. Considera que a informação capturada e demonstrada através dos diferentes plugins é suficiente?
   a. Sim
   b. Não. Porquê: _________________________________________

3. Considera suficiente as opções de resize, pan, zoom e swap existentes para os plugins?
   a. Sim
   b. Não. Que falta:

4. A identificação de diferentes vozes está persistida entre os diferentes cenários? Considera esta abordagem correcta?
   a. Sim
   b. Não

5. O reconhecimento de palavras deve ser efectuado quando não existe qualquer documento a analisar ou apresentação a decorrer, possuindo assim um pobre grau de reconhecimento?
   a. Sim
   b. Não

6. O uso da funcionalidade Sliders facilitou a identificação do Slide pretendido na tarefa 4?
   a. Sim
   b. Não

7. Actualmente, a informação adquirida nos PowerPoints pode ser angariada através do add-in de PowerPoint antes do cenário suceder, ou aquando a visualização do PowerPoint. Considera esta abordagem correcta?
   a. Sim
   b. Não. Porquê: _________________________________________

8. Durante a execução das tarefas usou a Ajuda?
   a. Sim
   b. Não

9. Durante a execução das tarefas usou a Ajuda?
   a. Sim
   b. Não
10. Se a alínea (a) da pergunta anterior fizer parte da sua resposta, responda a esta pergunta. Considerou a Ajuda obtida relevante?
   a. Sim
   b. Não
   c. Podia ser melhor? Indique como: __________________________

11. O mecanismo de HighLight deve permitir fazer combinações ao clicar nas palavras e nos sujeitos, ou deve permanecer a unicidade neste ponto?
   a. Sim, deve permitir
   b. Não, deve ser só efectuado na ferramenta de HighLight.

12. Considera que deve ser usado o Report textual ou deve passar-se logo para o Report mais visual desenvolvido em HTML?
   a. Manter ambos
   b. Passar logo para o HTML
   c. Outro? Indique qual: __________________________

Sugestões:
7.1.4 Prototype Evaluation

Class/Lecture Evaluation Tasks

Nota: apenas use a funcionalidade Report na última tarefa.

- Tarefa 1: Identifique o instante mais barulhento da aula.
- Tarefa 2: Identifique o tópico mais vezes abordado.
- Tarefa 3: Determine o slide mais barulhento (Noise) e o slide mais discutido (Words) através da funcionalidade Sliders.
- Tarefa 4: Verifique através do uso da funcionalidade Report, Textual e Visual (versão html), se o que determinou anteriormente se verificou.

Meeting Evaluation Tasks

Nota: apenas use a funcionalidade Report na última tarefa.

- Tarefa 1: Identifique a pessoa mais interveniente.
- Tarefa 2: Identifique o tópico mais vezes abordado.
- Tarefa 3: Verifique através do uso da funcionalidade Report, Textual e Visual (versão html), se o que determinou anteriormente se verificou.
Evaluation Satisfaction Questionary

Com este questionário pretende-se compreender se os utilizadores gostaram de usar a aplicação SoundLog, pedindo-se a estes que a classifiquem segundo alguns critérios. Os utilizadores podem deixar os comentários que desejarem. Este questionário é anónimo e os dados serão tratados com confidencialidade. Muito obrigado pela disponibilidade para responder a este questionário.

Ao preencher o questionário tenha em conta que o grau de satisfação:

1 = Muito Insatisfeito;
2 = Insatisfeito;
3 = Nem Satisfeito, Nem Insatisfeito;
4 = Satisfeito
5 = Muito Satisfeito.

Scenarios and Application Flow

Qual é a sua satisfação acerca dos Tipos de Cenários escolhidos?
1    2    3    4    5

Qual é a sua satisfação acerca da Criação de Cenários e Captura de Informação?
1    2    3    4    5

Qual é a sua satisfação acerca do Carregamento de Cenários e Análise da Informação?
1    2    3    4    5

Captura de Informação (Plugins)

Qual é a sua satisfação acerca do Volume de Som capturado?
1    2    3    4    5

Qual é a sua satisfação acerca da Identificação de diferentes Vozes?
1    2    3    4    5

Qual é a sua satisfação acerca das Palavras Reconhecidas?
1    2    3    4    5

Qual é a sua satisfação acerca da Informação capturada das apresentações PowerPoint?
1    2    3    4    5
**Funcionalidades**

Qual é a sua satisfação acerca da funcionalidade Sliders existente nos plugins?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

Qual é a sua satisfação acerca da funcionalidade Highlight?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

Qual é a sua satisfação acerca da funcionalidade Report (Textual e Visual)?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

**Tools**

Qual é a sua satisfação acerca da Ferramenta Subjects (Renomeamento)?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

Qual é a sua satisfação acerca da Ferramenta Plugins (Gestão do Layout)?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

Qual é a sua satisfação acerca da Ferramenta Help (Ajuda)?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

**Geral**

Qual é a sua satisfação acerca da Informação Capturada?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

Qual é a sua satisfação acerca dos Plugins definidos?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

Qual é a sua satisfação acerca das Funcionalidades existentes?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

Qual é a sua satisfação acerca das Ferramentas existentes?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]

Qual é a sua satisfação acerca do Sistema?

1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [ ]