Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# MASTER THESIS

**TITLE: Adaptation of Voice Server to Automotive Environment**

**MASTER DEGREE: Master in Science in Telecommunication Engineering & Management**

**AUTHOR: David Salinas Vila**

**DIRECTOR: David Conejero Olesti**

**SUPERVISOR: Roc Messeguer Pallarés**

**DATE: October 27th 2009**

**Títol:** Adaptation of Voice Server to Automotive Environment

**Autor:** David Salinas Vila

**Director:** David Conejero Olesti

**Supervisor:** Roc Messeguer Pallarés

**Data:** 27 d'octubre de 2009

## Resum

En aquest document es presenta la investigació en dos temes de l'adaptació de les tecnologies de la parla a l'entorn de l'automòbil, que són l'adaptació d'un servidor de veu i els algoritmes d'adaptació del locutor a l'entorn de l'automòbil. S'ha realitzat mitjançant un acord acadèmic amb Telefónica I+D.

Aquest projecte s'integra en dos projectes d'investigació anomenats "Movilidad y Automoción para Redes de Transporte Avanzados" (MARTA) i Caring Cars. Tenen com a objectiu estratègic consolidar les bases científiques i tecnològiques a la mobilitat del segle XXI per permetre al sector dels STI ("Sistemes de Transport Intel•ligent") espanyols respondre als reptes d'eficiència, sostenibilitat, etc. els quals la societat europea, i la espanyola en concret, han de confrontar en els pròxims anys. En aquests projectes Telefónica I+D (TID) s'encarrega de l'estudi, especificacions i implementació de les tecnologies de la parla a l'entorn de l'automòbil tenint en compte les condicions d'usabilitat dels vehicles.

TID disposava d'un servidor que contenia eines de parla, el meu treball en particular va ser el d'adaptar aquest servidor a l'entorn de l'automòbil. A més a més, l'addició de noves llibreries, afegir noves funcions al servidor i ampliar i desenvolupar la comunicació mitjançant fitxers XML, i d'aquesta manera que es puguin cridar a aquestes noves funcions. Les tecnologies usades en el desenvolupament del servidor han estat C/C++, Java, JNI, XML, shell script, AWK i Perl.

A l'entorn de l'automòbil, els reconeixedors de veu veuen degradat greument el seu rendiment per culpa de la variabilitat de l'entorn, així que les estratègies dels reconeixedors han de ser analitzades i millorades per aquest escenari. Com a part del projecte, he realitzat tests per avaluar el comportament dels algoritmes d'adaptació al locutor en l'entorn de l'automòbil, i així poder incrementar la taxa de reconeixements amb èxit.

**Title:** Adaptation of Voice Server to Automotive Environment

**Author:** David Salinas Vila

**Director:** David Conejero Olesti

**Supervisor:** Roc Messeguer Pallarés

**Date:** October, 27th 2009

## Overview

This document presents the research in two topics in the adaption of speech technologies to automotive environment, these are the adaptation of a voice server and speaker adaption algorithms to an automotive environment. It has been done with an academic agreement with Telefónica I+D.

This project is embedded within two investigation projects named “Movilidad y Automoción para Redes de Transporte Avanzados” (MARTA) and Caring Cars. They have as a fundamental strategic goal to consolidate the scientifically and technological basis to 21th century mobility to allow Spanish ITS (“Intelligent Transport Systems”) sector to answer the challenges of efficiency, sustainability, etc . which European society and especially Spanish society has to confront in the next years. In these projects Telefónica I+D (TID) is in charge of the study, specification and implementation of speech technology in automotive environment considering vehicle usability conditions.

TID had a server that contained speech tools, and my particular job was to adapt it to automotive environment. Moreover, add new libraries that annex new functions to this server and extend and develop the communication through XML files to use these new functions. Technologies used in the development of the server are C/C++, Java, JNI, XML, shell script, AWK and Perl.

In automotive environment, speech recognizers severely degrade their performance because of environment variability, so the strategies of recognizers have to be analyzed and improved in this new environment. As part of the project, I realized tests to evaluate the performance of speaker adaptation algorithms in an automotive environment to increase the rate of successful recognitions.

# INDEX

# INTRODUCTION

In recent years the increment of the number of cars per capita has increased the concern of the authorities and society in general about car accidents, which as have been seen in some studies, are mostly caused by driver distraction. Together with the great increase of electronic devices that the driver has to act has led to emerge a special interest in the development of more usable and secures interfaces in the car, as is the use of a voice interface between man-machine. This interest has led Telefónica I+D (TID) to investigate this type of projects given its extensive experience in speech recognition and generation of speech synthesis. These investigations have focused on adapting TID tools to a hostile environment such as the automotive, and the study of improvements against noise.

My job was to develop and adapt a voice server, based on a server used for other functions, to an automotive environment. The server developed in this project will be used in two public projects MARTA and Caring Cars. Another part of the project is to continue with the evaluation of tests of speaker adaptation algorithms, which serve to improve recognition in a noisy environment like in a vehicle.

The report is structured in five chapters, described in next paragraphs:

Chapter 1 - In this chapter there is an extended explanation about what does the project involve and what is my job, motivations and goals of it, as well as and state of art about technology related to work developed.

Chapter 2 - In this chapter it is explained to the reader the basic knowledge about speech technologies and software used to correctly follow the report. It will help to understand some integration solutions chosen as well as some empirical results of the tests. It tries that any reader not familiarized with that technology could follow the report without problems, and for that reason it avoids some technical aspects.

Chapter 3 - In this chapter it is explained in detail the voice server and their components. The explanation will be based in the operation of it within the voice system in Caring Cars and MARTA projects. To facilitate the comprehension and provide an overall view of what has been implemented, it will also be explained in detail the communication between client and server. Later, there will be a description of the main features of ASR (Automatic Speech Recognition), TTS (Text To Speech) of Telefónica I+D and Agnitio verification library used in this project.

Chapter 4 - In this chapter it is explained my job developing speaking tools in Telefónica I+D, it is done whilst developing the server program. My task has been centered in testing different strategies about the use of different algorithm for the speaker adaptation, exactly the adaptation of independent speaker acoustic models to a particular speaker to increase the rate of successful recognitions by ASR. This task is the continuation of some tests started previously by TID colleagues.

Chapter 5 - In this last chapter there are exposed the conclusions that can be obtained from this project, taking into account all the development and tests done. Moreover, there is included the future work that can be done in this topic. It is important to stress that my project is included as a part in a bigger project that is still developed by other colleagues, so it is presumably that in a near future these improvements can be applied.

# CHAPTER 1. PROJECT OUTLINE

## 1.1. Motivations

Over the past four decades the number of electronic devices in cars and its sophistication has increased exponentially. It has emerged new technologies that can be incorporated in the car (mobile phone, PDA's, MP3, onboard computer, GPS....). All of these factors that overload our vision with a lot of information, coupled with the fact that today people spend a lot of time traveling in the car, make conduction more unsafe, due to distractions caused by the interaction with the current human-machine interfaces used, most of them visual or tactile (button, touchscreen, control panels,....).

Because of these facts, a great concern about interoperability between humans and machines has been incited in car companies and in the government responsibles of road traffic safety. The introduction of electronics and its new possibilities makes that we should think in their control on cabin in ergonomics, useful and essentially usability way, and that they always follow criterions of safety and efficiency.

Considering for example the simple fact of touch or watch GPS's screen while someone is driving in a highway, it makes that the driver has to put simultaneously his attention on the GPS, the road and the rest of information that car's sensors give to him (speed, fuel level,...) doing the task unsafe and uncomfortable. Knowing that, it is easy to think in a type of human-machine interface able to improve the safety and comfort of the car like voice interface, or a multimodal interfaces, that allow user to have access to all the information somehow safer and comfortable, and will fulfil the requirement of interoperability. Not merely thinking in a way that user says a command when pushes a push to talk bottom, but thinking in a voicebased dialogue. Some researches [10] show that human error is the source of over 90% of all road accidents. Many accidents can be prevented if the driver is made aware of his or her physical condition, and encouraged to focus his or her attention or stop the vehicle. Research has also shown that solitary drivers are more prone to accidents than drivers accompanied by an adult passenger. An interactive voicebased dialogue supported by sensor based information can play a role in reducing road accidents by mimicking the role of an adult passenger.

According to the growing interest in this type of interface, Telefónica I+D has decided to investigate in this kind of projects, due to the vast experience of more than fifteen years in the development of applications with voice recognition and artificial voice synthesis. The objective is to adapt the technology developed by Telefónica I+D inside a vehicle to increase its safety.

## 1.2. Goals

The main objectives of this project are to obtain adaptive recognition tools to automotive environment and improve recognition on special car environment: noise caused because of wind, another car, car itself, rain and continuously changing over time. Moreover, it is desired to obtain well performance of the system in a simple user case.

My project is going to be realized in Telefónica I+D enterprise, collaborating with the current working groups. To better understand the aim of my work, in next paragraph it is explained the present development that TID is doing in speech recognition applied to a vehicle.

The main problem that TID has to deal with is the unfriendly environment of vehicles. As consequence of this, it should be used a small vocabulary owing to the elevated error percentage caused to high noise level. To fight against noise, it is being improved the recognition robustness, following several strategies from low level acoustic processing (echo cancellation, additive noise reduction, microphone arrays), to speech recognition with activity detection, noise modelling, and robust speech parameterization. Another study focus has been speech recognition, it means not just to take into account one word like happens nowadays, but to use the speech and make it more complex, so the dialog can be improved. Following, there are some specific researches developed:

- Study the different noise sources and distortion in a vehicular environment.
- Research, development and evaluation of the latest methodologies and algorithms to detect the extremes to distinguish between voice and silence delimiting the start and end of words, within the voice robust coding system "AURORA[1]".
- Research, development and evaluation of speech recognition systems in non-stationary variable length noise environments.
- Research, development and evaluation of acoustic adaptation systems to the speaker and environment using MAP and MLLR algorithms.
- Adaptation of a speech synthesis system to be used in limited capacity hardware installed in a CAR-PC vehicle.
- Study of techniques to context representation, and access to this information during spoken interaction: interaction enhancement.
- Study techniques to incorporate context information in the adapting process of TID recognizer.
- Preliminary analysis of context information necessary for prioritizing tasks and adequate control of spoken interaction in the vehicle.

[1] Acoustic models trained with parameterized files according to AURORA standard [6]

My task in Telefónica I+D can been divided in two different topics. The first one is to develop and adapt a voice server to offer a speech language system and the second one is a speaker adaptation. When I began the project TID had a server half developed applied to voice functions, an objective of the project was to develop a new one based on it, and adapt it to be suitable to be installed inside a car. The programming language of the server is linked with the development of it taking as a model a previous one made in C. Moreover, I have added new functionalities. A feature of the server is that it offers functions at any service, independently of the programming language used and furthermore it is scalable.

Speaker adaptation means to adapt a speech recognizer to a specific speaker and environment. My task, particularly, has been to test different strategies to speaker adaptation, and study and evaluate the best strategy to achieve a high percentage in success recognitions for a specific speaker.

The server developed in the project will be employed in two public projects where TID is working on. Although the specifications are slightly different in one project than in the other, they do not influence on the server. The two public projects where the server will be used are PROFIT Caring Cars, that is an European project, and CENIT "Movilidad y Automoción para Redes de Transporte Avanzadas" (MARTA), that is a Spanish project. These two great projects are dedicated to increase car safety by enabling wellness applications in an automotive environment.

In both projects there are important enterprises working in, as Philips Electronics Nederland BV, NXP IC Lab, TOFAS Turk OtomobilFabrikasi A.S., GRUNDIG Elektronik A.S., among others in Caring Cars project and Ficosa International S.A., ETRA investigacion y desarrollo S.A.,TECNALIA, CEDETEL, Agnitio S.L., SEAT in MARTA project. The task to be developed by TID in both projects is similar. In addition to develop the server, TID will work conjointly with Universidad Politecnica de Madrid (UPM) to develop a complete speaker language system. TID specifics tasks in these projects are, although little differences, tackles the development a module on Human Machine Interface (HMI), a Voice management module for seamless interaction between the user and the services (either running on this platform or from any other device in the surroundings). This module will be built using the best practices within the automotive industry and will allow multimodal and safe interfacing with the driver. The HMI modes will encompass a Graphical User Interface (GUI), voice command technologies and text-to-speech functions, which are designed to act as single uniform assistant to the user and turn interaction into a more efficient and less driver's attention consuming task.

During the development of this project, it has been finished the software specifications on MARTA and Caring Cars projects, so specifications of Telefónica I+D, voice server software were changed due to the change of the initial operative system that had to implement the voice server.

Fig. 1.1 and Fig. 1.2 show a basic schema of the development where TID is in charge. We can observe that in the schemas that there are few differences between Caring Cars and MARTA, that actually is the difference between HMI specifications. One of the major difference is that in Caring Cars the main applications and services are inside OSGI platform (explained in chapter 3), while in MARTA they are outside HMI, it implies that communication between applications and services with HMI manager are carried in a different way in each project. But these differences almost not influence the voice system that in both receive orders through a XML.

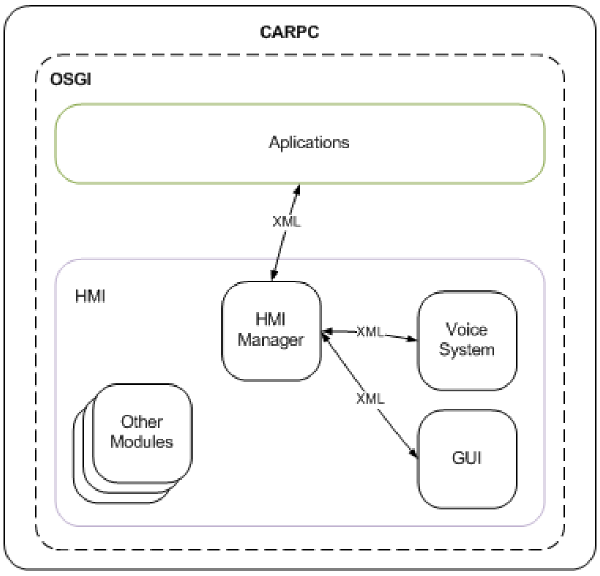

**Fig. 1.1** Basic schema of the architecture of Caring Cars

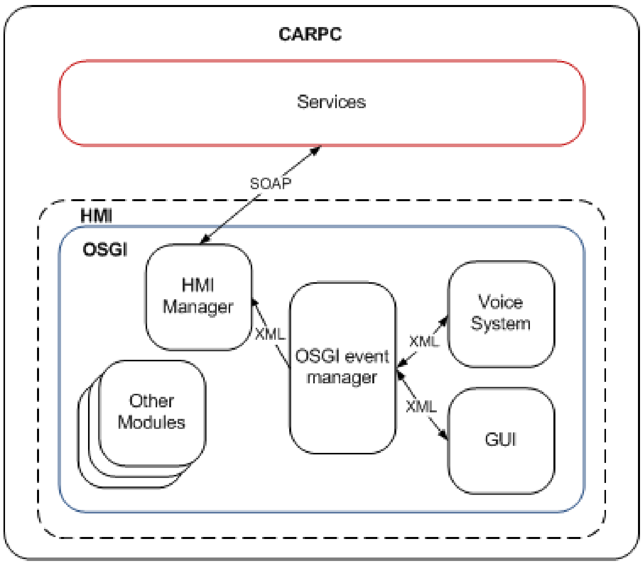

**Fig. 1.2** Basic schema of the architecture of MARTA

## 1.3. State of the Art

In the context of a multimodal interface in a vehicular environment including voice systems, at this moment we have a highly evolved example in U.S.A, the Ford Sync, in-car communications and entertainment system developed by Ford and Microsoft. The system is currently offered on 12 different Ford, Lincoln and Mercury vehicles in North America, starting with the 2008 model year.

Ford partnered with Microsoft to create Sync as a way for vehicle owners to stay connected with their cell phones and digital music players. Instead of needing separate connections for these devices, Sync works as an all-in-one solution that can be updated as new phones and players are released. Ford Sync looks to provide a simpler, one-stop-shop for connection and use of cell phones and MP3 players. To use these services, it should be hit the voice command button in a Sync-equipped car and request services. The system employs the user Bluetooth paired phone to connect with a server where the user can request, by voice, directions, traffic conditions, a local business search, and other services.

**Fig. 1.3** With Ford Sync, consumers can operate a mobile phone or digital media player in three ways: using voice commands, the vehicle's steering wheel, or its radio controls.

But from a viewpoint of user interface, these systems are based only on instructions with commands and control capabilities to a very limited dialogue with the system, for example the mechanism for interaction was the touch of a push to talk button and subsequent launch of a single command, such as climate control *temperature up*. The problem with this type of interface is that the driver needs to know all the commands available, which generally make up a list of phrases that are large stipulated in the user manual, and are difficult to remember. Even commands often are not natural, following an artificial language to optimize the accuracy of the recognizer, which made it even harder to memorize.

This kind of problem has tackled in (Pieraccini and Goodman, 2003) where they implemented a prototype of a multimodal interface for some functions. Developing a conversational interface as opposed to command and control systems, and introduced a graphical interface along with a haptic interface (thanks to a touch screen). The graphical interface help voice interface providing the user with additional information about the state of dialogue, suggestions on what to say, and feedback about the commands launched by the user (for example, to confirm the command recognized by the system).

An example of a possible architecture for this type of interfaces can be seen in the project TALK Fig. 1.4. This project has developed a system enabling multimodal interact with an MP3 player that supports the usual features. The system has two modes of interaction by the user: one monomodal, controlling functions through voice whenever the driving situation makes this possible and reasonable, and one that also uses multimodal voice along with a haptic controller. Similarly the output of the system is multimodal, including spoken messages and relevant information on a display.

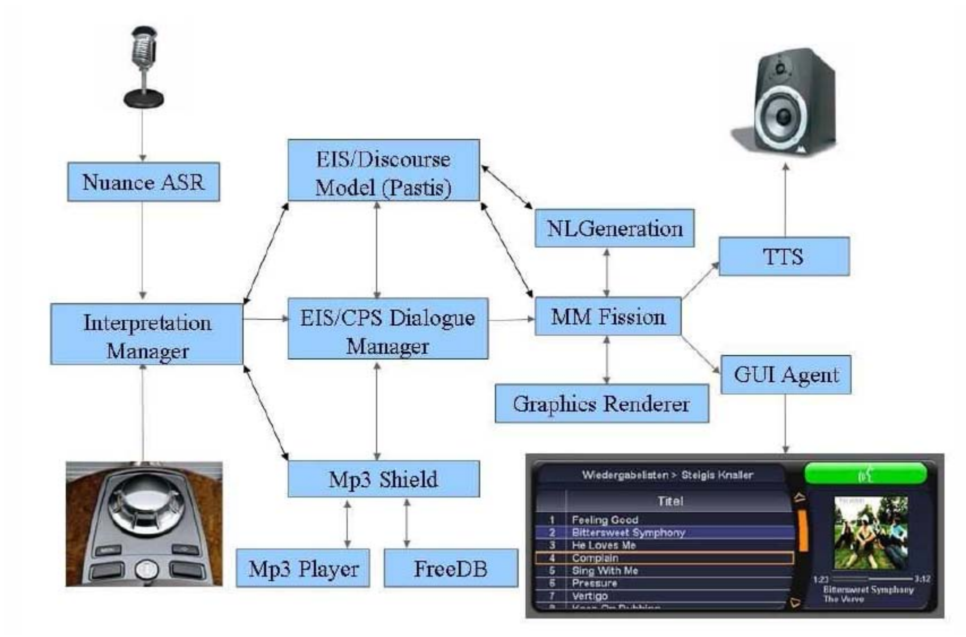

**Fig. 1.4** Board system architecture [Talk Project, Deliverable 5.3]

# CHAPTER 2. SPOKEN LANGUAGE SYSTEMS

That chapter will explain to the reader the basic knowledge about speech technologies and software used to correctly follow the report. It will help to understand some integration solutions chosen as well as some empirical results of the tests. It tries that any reader not familiarized with that technology could follow the report without problems, and for that reason it avoids some technical aspects.

Speech is the most natural means of communication among humans. It is also believed that spoken language processing (technologies related to speech recognition, text-to-speech, and spoken language understanding), will play a major role in establishing a universal interface between humans and machine. This spoken language processing is a diverse field that relies of knowledge at the levels of signal processing, acoustics, phonology, phonetics, syntax, semantics, pragmatics, and discourse. A spoken language system has at least one of the following three subsystems: a speech recognition system that converts speech into words, a text-to-speech system that conveys spoken information, and a spoken language understanding system that maps words into actions and that plans system-initiated actions.

Usually, the technology employed in one of these three subsystems is also applied inº the others. Manually created rules have been developed for spoken language systems with limited success. But, in recent decades, data-driven statistical approaches have achieved encouraging results, which are usually based on modelling the speech signal using well-defined statistical algorithms that can automatically extract knowledge from the data. The data-driven approach can be viewed fundamentally as a pattern recognition problem. In fact, speech recognition, text-to-speech conversion, and spoken language understanding can all be regarded as pattern recognition problems. The patterns are either recognized during the runtime operation of the system or identified during system construction to form the basis of runtime generative models such as prosodic templates needed for text to speech synthesis. While a statistical approach is used and advocate, there is no need to exclude the knowledge engineering approach from consideration. If there are a good set of rules in a given problem area, there is no need to use a statistical approach at all. The problem is that, nowadays, there is not enough knowledge to produce a complete set of high-quality rules. As scientific and theoretical generalizations are made from data collected to construct data-driven systems, better rules may be constructed. Therefore, the rule-based and statistical approaches are best viewed as complementary [1].

## 2.1. Automatic Speech Recognition

In this section, it is explained the basis to understand the steps that automatic speech recognition follows, since it receives the voice of the speaker until it recognizes the words.

A typical practical speech recognition system consists of basic components shown in the dotted box of Fig. 2.1. The signal processing handles the analogue acoustic waveform in order to obtain the representation of speech signal in the frequency domain, that is especially useful because the frequency structure of a unit of speech (word, syllable, triphoneme, phoneme...) is generally unique, and quantifier and coding this signal, speech coding [1], to obtain finally a series of data, cepstrum[2], that allow decoder work with them. *Acoustic models* include the representation of knowledge about acoustics, phonetics, microphone and environment variability, gender and dialect differences among speakers, etc. *Language models* refer to a system's knowledge of what constitutes a possible word, what words are likely to co-occur, and in what sequence.

The speech signal is processed in the signal processing module that extracts salient feature vectors for the decoder. The decoder uses both acoustic and language models to generate the word sequence that has the maximum posterior probability for the input feature vectors. All words embedded within a limited vocabulary size supplied by the search algorithm on Fig. 2.1.

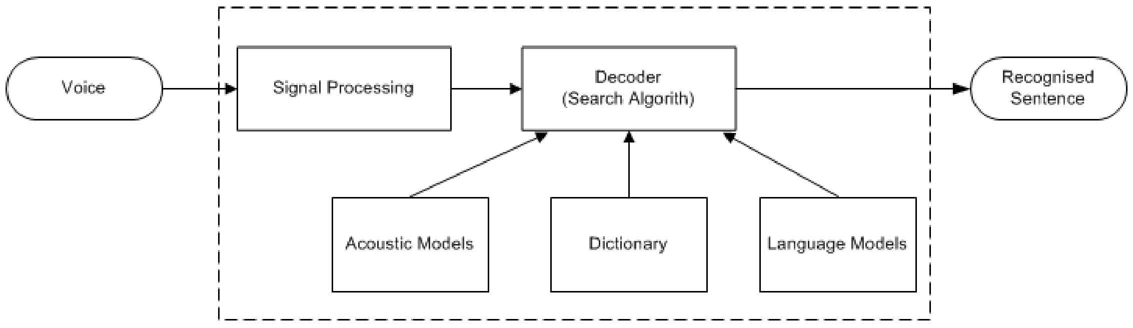

**Fig. 2.1** Basic system architecture of a speech recognition system

### 2.1.1. Acoustic Modelling

Acoustic Models is a statistical representation of the sounds of the speech that makes up each word. It is used by a speech recognition engine to recognize speech. If we want to have success we should have an accurate acoustic model. But it is not so easy because of the variations in context, the same phonetically word could have different meanings in different context, the words

[2] The *Mel-Frequency Cepstrum Coefficients* (MFCC) is a representation defined as the real cepstrum of a windowed short-time signal derived from the FFT of that signal.

write, right or Mr. Wright in English for instance. Variations in speaker, we do not speak the same words always equal, we speak different in a presentation or when we talk with a friend or when we discuss a subject. And variations in environment, the same word could be very different within a room in silence or in a room with a lot of noise, besides different accents when a word is pronounced from a man, a woman, people of different regions in a country, etc. All of these factors lead to the necessity to make an acoustic modelling, which plays a critical role in improving accuracy and is arguably the central part of any speech recognition system.

The practical challenge is how to build accurate acoustic models*,* and language models, that can truly reflect the spoken language to be recognized. We want that for the given acoustic observation $X = X_1X_2...X_n$ , we find out the corresponding word sequence $W= w_1w_2…w_n$ that has the maximum posterior probability.

One of the key issues in acoustic modelling has been the choice of a good unit of speech. In small vocabulary systems of a few tens of words, it is possible to build separate models for entire words, but this approach quickly becomes infeasible as the vocabulary size grows. For one thing, it is hard to obtain sufficient training data to build all individual word models. It is necessary to represent words in terms of sub-word units, and train acoustic models for the latter, in such a way that the pronunciation of new words can be defined in terms of the already trained sub-word units [2].

Although we may try to say each word as a concatenated sequence of independent phonemes, these phonemes are not produced independently, because our articulators cannot move instantaneously from one position to another. For example, if context-independent phonetic models are used, the same model for *t* must capture various events, such as flapping, unreleased stops, and realizations in */t s/* and */t r/.* Then, if */t s/* is the only context in which *t* occurs in the training, while */t r/* is the only context in the testing, the model used is highly inappropriate. While word models are not generalizable, phonetic models overgeneralize and, thus, lead to less accurate models.

A *triphone* model is a phonetic model that takes into consideration both the left and the right neighbouring phones. If two phones have the same identity but different left or right contexts, they are considered different triphones. This is the model used in most systems.

In the next step we should examine *triphone* in the context of the other *triphone* around them. It runs the contextual *triphone* plot through a complex statistical model and compares them to a large library of known words, phrases and sentences. Then determines what the user was probably saying and either outputs it as text. Today's speech recognition systems use powerful and complicated statistical modelling systems. These systems use probability and mathematical functions to determine the most likely outcome. The model that dominate the field today are the HMM, (Hidden Markov Model described in (Section 2.1.2). The main disadvantage of use that models are the great number of triphones (more than 20000), and each triphone is in turn, modelled by a 5-state HMM, it makes too many states. Since there isn't sufficient training data to build models for each state, they are clustered into equivalence classes called senones (see [3]).

### 2.1.2. Hidden Markov Models

To minimally understand the HMM, in this section it is included a short explanation about Markov Chain. A Markov Chain models a class of random processes that incorporates a minimum amount of memory without being completely memoryless. If we associate the value of a random variable, *X*, to a state, the Markov chain can be represented by a finite state process with transition between states specified by a probability function. Using this finite state representation, we will have *the Markov assumption*: the probability that the Markov chain will be in a particular state at a given time depends only on the state of the Markov chain at the previous time (for more informaion see [1]).

### 2.1.3. Language Models

Large vocabulary continuous speech recognition requires, besides an acoustic pattern matching**,** the use of a language model or grammar to select the most likely word sequence from the relatively large number of alternative word hypotheses produced during the search process. The absence of explicit word boundary markers in continuous speech causes several additional word hypotheses to be produced, in addition to the intended or correct ones. For example, the phrase *It's a nice day* can be equally well recognized as *It sun iced A.* or *It son ice day*. They are all acoustically indistinguishable, but the word boundaries have been drawn at a different set of locations in each case. Clearly, many more alternatives can be produced with varying degrees of likelihood, given the input speech. The language model is necessary to pick the most likely sequence of words from the available alternatives.

That information could be represented among several forms, the formal language model, where two things are fundamental: the grammar and the parsing algorithm. The *grammar* is a formal specification of the permissible structures for the language. The *parsing* technique is the method of analyzing the sentence to see if its structure is compliant with the grammar. With the advent of bodies of text (*corpora*) that have had their structures hand-annotated, it is now possible to generalize the formal grammar to include accurate probabilities. Furthermore, the probabilistic relationship among a sequence of words can be directly derived and modeled from the corpora with the so-called stochastic language models, such as *n*-gram, avoiding the need to create broad coverage formal grammars.

The formal language runs correctly when there are a little vocabulary and has a few different sentences, but their size grows exponentially when the vocabulary increase and it does not represent all the possibilities of natural language that in much times does not follow grammatical rules. In a stochastic language models, it allows theoretically all possible combination among words.

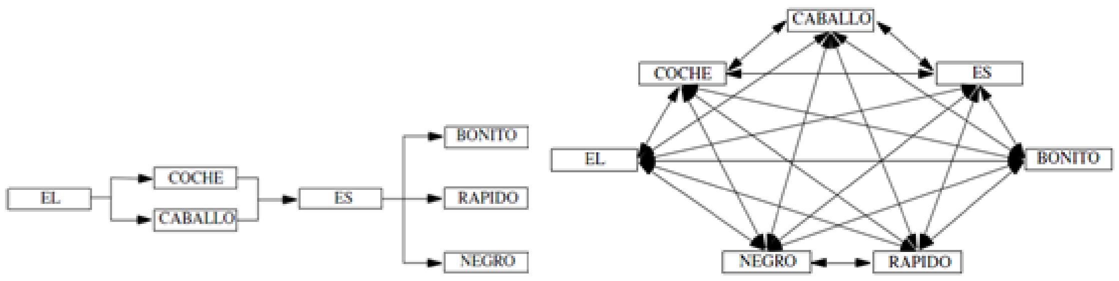

**Fig. 2.2** Example of possible links in a formal language (left) and stochastic language (right)

Simple tasks, in which one is only required to recognize a constrained set of phrases, can be used rule-based regular or context-free grammars which can be represented compactly. However, that is impossible with large vocabulary tasks. Instead, bigram and trigram grammars, consisting of word pairs and triples with given probabilities of occurrence, are most commonly used. One can also build such language models based on word classes, such as city names, months of the year, etc. However, creating such grammars is tedious as they require a fair amount of hand compilation of the classes. Ordinary word n-gram language models, on the other hand, can be created almost entirely automatically from a corpus of training text.

Clearly, it is infeasible to create a complete set of word bigrams for even medium vocabulary tasks. Thus, the set of bigram and trigram probabilities actually present in a given grammar is usually a small subset of the possible number. Even then, they usually number in the millions for large vocabulary tasks. The memory requirements for such language models range from several tens to hundreds of megabytes.

It is preferable to have a smaller vocabulary size, since this eliminates potential confusable candidates in speech recognition, leading to improved recognition accuracy. However, the limited vocabulary size imposes a severe constraint on the users and makes the system less flexible.

### 2.1.4. Search Algorithm

Continuous speech recognition (CSR) is both a pattern recognition and search problem. In the case of HMM-based systems, the former refers to the probability of a given HMM state emitting the observed speech at a given time. The latter refers to the search for the best word sequence given the complete speech input. The complexity of a search algorithm is highly correlated with the search space, which is determined by the constraints imposed by the language models.

Speech recognition --searching for the most likely sequence of words given the input speech-- gives rise to an exponential search space if all possible sequences of words are considered. The problem has generally been tackled in two ways: Viterbi decoding using beam search, or stack decoding, which is a variant of the A* algorithm. Some hybrid versions that combine Viterbi decoding with the A* algorithm also exist [2]. in this report only do a brief explain on Viterbi decoding due to it is nowadays the preferred method for a most of speech recognition and it is the used in speech recognition's project, if you want more information about others algorithm see [1][2].

Viterbi decoding is a dynamic programming algorithm that searches the state space for the most likely state sequence that accounts for the input speech. The state space is constructed by creating word HMM models from its constituent phone or triphone HMM models, and all word HMM models are searched in parallel. Since the state space is huge for even medium vocabulary applications, the beam search heuristic is usually applied to limit the search by pruning out the less likely states. The combination is often simply referred to as Viterbi beam search. Viterbi decoding is a time-synchronous search that processes the input speech one frame at a time, updating all the states for that frame before moving on to the next frame. Most systems employ a frame input rate of 100 frames/sec [2], (for more information about viterbi decoding see [2]).

## 2.2. Text-to-Speech Conversion

The term *text-to-speech*, often abbreviated as TTS, is easily understood. The task of a text-to-speech system can be viewed as speech recognition in reverse – a process of building a machinery system that can generate human-like speech from any text input to mimic human speakers. TTS is sometimes called *speech synthesis*, particularly in the engineering community.

The conversion of words in written form into speech is nontrivial. Even if we can store a huge dictionary for most common words; the TTS system still needs to deal with millions of names and acronyms. Moreover, in order to sound natural, the intonation of the sentences must be appropriately generated.

The basic components in a TTS system are shown in Fig. 2.3. The text analysis component normalizes the text to the appropriate form so that it becomes speakable. The phonetic analysis component converts the processed text into the corresponding phonetic sequence, which is followed by prosodic analysis to attach appropriate pitch and duration information to the phonetic sequence. Finally, the speech synthesis component takes the parameters from the fully tagged phonetic sequence to generate the corresponding speech waveform.

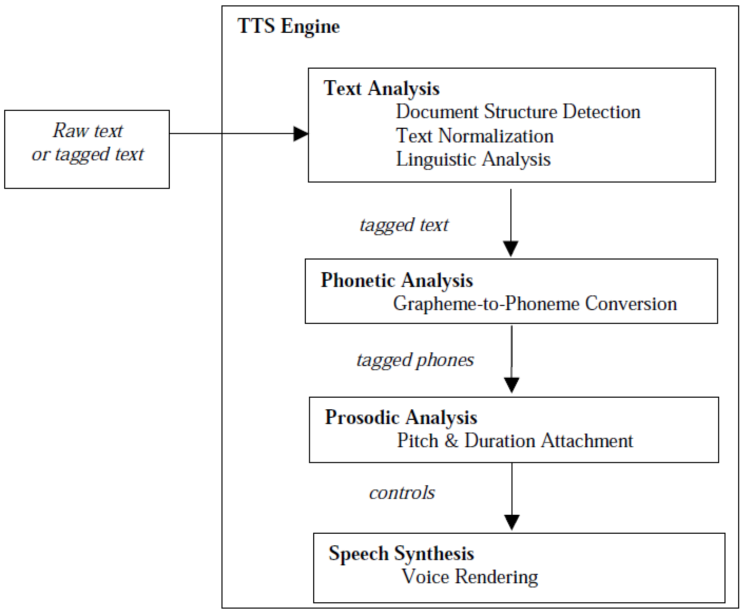

**Fig. 2.3** Basic system architecture of a TTS system (source [1])

### 2.2.1. Text and Phonetic Analisys

Text analysis and phonetic analysis have as objective convert words into speakable phonetic representation. To have a success on that, first it have a text analysis for indicating all knowledge about the text or message that is not specifically phonetic or prosodic in nature. Text analysis for TTS involves three related processes:

- *Document structure detection*—Document structure is important to provide a context for all later processes. In addition, some elements of document structure, such as sentence breaking and paragraph segmentation, may have direct implications for prosody.
- *Text normalization*—Text normalization is the conversion from the variety symbols, numbers, and other nonorthographic entities of text into a common orthographic transcription suitable for subsequent phonetic conversion.
- *Linguistic analysis*—Linguistic analysis recovers the syntactic constituency and semantic features of words, phrases, clauses, and sentences, which is important for both pronunciation and prosodic choices in the successive processes.

Besides this processes, some languages need another services to have a grapheme-to-phoneme conversion, it is trivial for languages where there is a simple relationship between orthography and phonology. Such a simple relationship can be well captured by a handful of rules. Languages such as Spanish and Finnish belong to this category and are referred to as *phonetic* *languages*. English, on the other hand, is remote from phonetic language because English words often have many distinct origins.

It is arguable that text input alone does not give system enough information to express and render the intention of the text producer. Thus, more and more TTS systems focus on providing an infrastructure of standard set of markups (tags), so that the text producer can better express their semantic intention with these markups in addition to plain text. These kinds of markups have different levels of granularity, ranging from simple speed settings specified in *words per minute* up to elaborate schemes for semantic representation of concepts that may bypass the ordinary text analysis module altogether. It does not remain to be said that location is an important issue in a TTS system, due to writing systems of language communities may differ substantially in arbitrary ways, necessitating serious effort in both specifying an internationalized architecture for text and phonetic analysis, and localizing that architecture for any particular language.

### 2.2.2. Prosody

Prosody is a complex weave of physical, phonetic effects that is being employed to express attitude, assumptions, and attention as a parallel channel in our daily speech communication. The semantic content of a spoken or written message is referred to as its *denotation*, while the emotional and attentional effects intended by the speaker or inferred by a listener are part of the message's *connotation*. From the listener's point of view, prosody consists of systematic perception and recovery of a speaker's intentions based on:

- *Pauses*: to indicate phrases and to avoid running out of air.
- *Pitch*: rate of vocal-fold cycling (fundamental frequency or F0) as a function of time.
- *Rate/relative duration*: phoneme durations, timing, and rhythm.
- *Loudness*: relative amplitude/volume.

An easy form to see the prosodic generation is Fig. 2.4 that shows schematically the elements of prosodic generation in TTS. The input of the prosody module in Fig. 2.4 is parsed text with a phoneme string, and the output specifies the duration of each phoneme and the pitch contour.

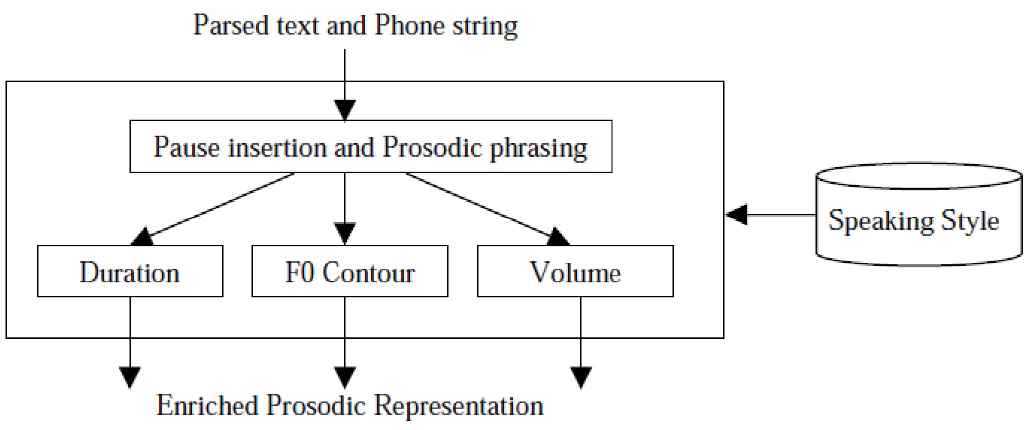
Parsed text and Phone string
Pause insertion and Prosodic phrasing
Duration
F0 Contour
Volume
Speaking Style
Enriched Prosodic Representation

**Fig. 2.4** Block diagram of a prosody generation system; different prosodic representations are obtained depending on the speaking style we use (source [1]).

The *speaking style* of the voice can impart an overall tone to a communication. And could add character, refers primarily to long-term, stable, extralinguistic properties of a speaker, such as membership in a group and individual personality, or an emotion. Also could add an accent, a different tone or other effects before it pass to speech synthesis module.

### 2.2.3. Speech Sysnthesis

*T*he speech synthesis module of a TTS system is the component that generates the waveform. The input of traditional speech synthesis systems is a phonetic transcription with its associated prosody. The input can also include the original text with tags, as this may help in producing higher-quality speech.

The most important attribute of a speech synthesis system is the quality of its output speech. It is often the case that a single system can sound beautiful on one sentence and terrible on the next. For that reason we need to consider the quality of the best sentences and the percentage of sentences for which such quality is achieved. This tradeoff is illustrated in Fig. 2.5, where we compare four different families of speech generation approaches:

- *Limited-domain waveform concatenation*. For a given limited domain, this approach can generate very high quality speech with only a small number of recorded segments. Such an approach, used in most interactive voice response systems, cannot synthesize arbitrary text.
- *Concatenative synthesis with no waveform modification*. Unlike the previous approach, these systems can synthesize speech from arbitrary text. They can achieve good quality on a large set of sentences, but the quality can be mediocre for many other sentences where poor concatenations take place.

- *Concatenative systems with waveform modification*. These systems have more flexibility in selecting the speech segments to concatenate because the waveforms can be modified to allow for a better prosody match. This means that the number of sentences with mediocre quality is lower than in the case where no prosody modification is allowed. On the other hand, replacing natural with synthetic prosody can hurt the overall quality. In addition, the prosody modification process also degrades the overall quality.
- *Rule-based systems*. Such systems tend to sound uniformly across different sentences, albeit with quality lower than the best quality obtained in the systems above.

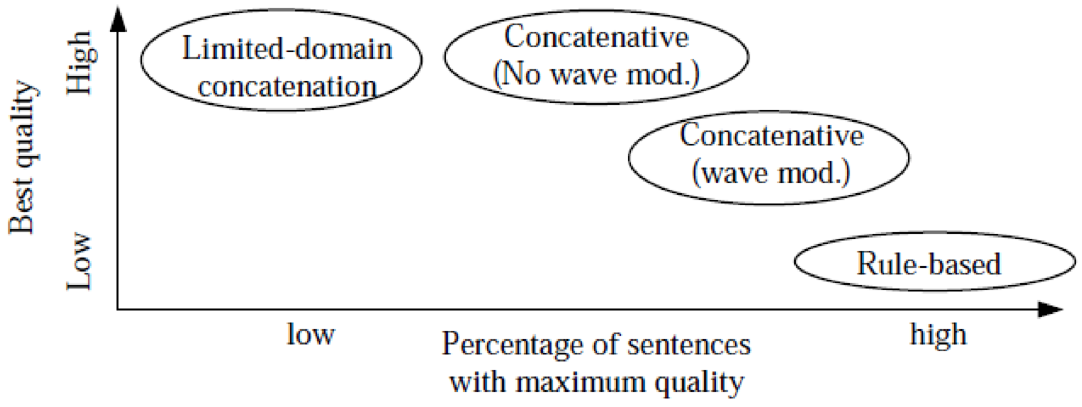

**Fig. 2.5** Quality and task-independence in speech synthesis approaches (source [1])

How is logical to think this quality will come determined for the equipments where TTS be implemented.

## 2.3. Spoken Language Understanding

In this section we make a little explanation about Dialog Manager the component that communicates with applications and the spoken language understanding modules such as discourse analysis, sentence interpretation, and message generation. These components are parts outside the project, nevertheless will be the components that our server has to supply its services, and for that reason it is interesting to pay a little attention on their. This component with component's server, speech recognition and text-to-speech, conform a spoken language understanding system.

While most components of the system may be partly or wholly generic, the dialog manager controls the flow of conversation tied to the action. The dialog manager is responsible for providing status needed for formulating responses, and maintaining the system's idea of the state of the discourse. The functions that deliver spoken language understanding system are useful if we want to give autonomy to the speaker when he wants to interact with the machine. Suppose a recognizer correctly transcribes a series of spoken words into the written form—the system still has no idea what to do, because there is often no direct mapping between a sequence of words (or the syntactic structure of the sentence) and the functions that the system provides.

On sentence interpretation module convert (translate) a user's query (sentence) into the semantic representation. In other words, one has to fill the semantic slots with information derived from the content (words) in the sentence, whereas on discourse analysis module make a process also called semantic evaluation, attempts to interpret each sentence with knowledge about the current dialog status or discourse. Otherwise, for most applications, it is highly unlikely that a user can access or retrieve the desired information with just a single query. The query might be incomplete, imprecise, and sometimed inconsistent with respect to the discourse history. Even if the query is unambiguous, the speech recognition and sentence interpretation modules in a SLU system may make mistakes. Thus the SLU system needs to provide an interactive mechanism to perform clarification, completion, confirmation, and negotiation dialogs with users. The dialog manager controls the interactive strategy and flow once the semantic meaning of the query is extracted and stored in the system's representation. It perform as a GUI application that contains an event handler. The event handler handles dialog events passed from the discourse analysis module and generates appropriate responses to engage users to solve the problems.

# CHAPTER 3. VOICE SYSTEM

In this chapter it is explained in detail the voice server and their components. The explanation will be based in the operation of it within the voice system in Caring Cars and MARTA projects. To facilitate the comprehension and provide an overall view of what has been implemented, it will also be explained in detail the communication between client and server. Later, there will be a description of the main features of ASR, TTS of Telefónica I+D and Agnitio verification library used in this project.

## 3.1. System General Schema of MARTA and Caring Cars

As it has been explained in CHAPTER 1, voice system is located within HMI, offering voice services. In the block diagram of Fig. 3.1 it can be note that the voice system consist of two main components, the dialogue modules, which are developed by UPM and TID Madrid, and the voice server and voice tools that are developed in TID Barcelona. The server offers all needed functions to the dialogue modules, consequently, to develop the server it is needed collaboration with colleagues in charge of them.

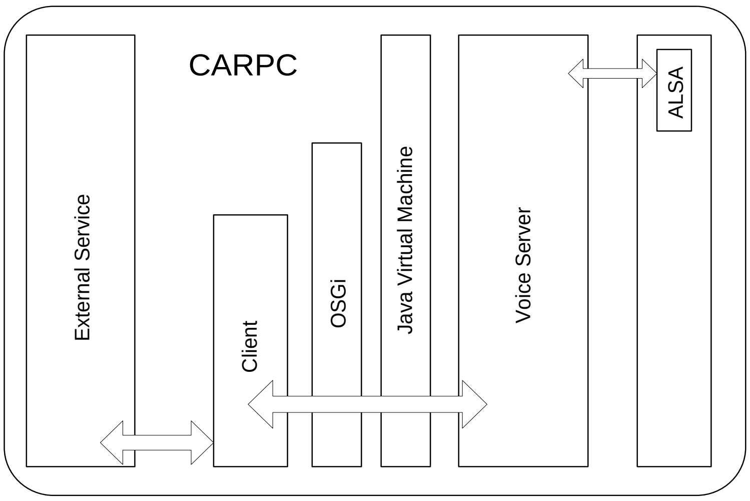

**Fig. 3.1** System general schema

In the left hand side of Fig. 3.1 can be found the external services, which supply voice functions to their applications, according to the specific need of each service. These services can request to the voice system the voice services customized to adequate to their needs. These requests, controlled by HMI manager, are received in the voice system by the client which is in charge of control and offer the different functions provided by the voice system.  The voice system manages the services requests and sends the commands to the voice server to be resolved. In the voice server are located TID tools, which offer voice services, as well as other available functions to the services. Once the client function to execute is received, the server performs the required actions, if it is necessary it accedes to the vehicle hardware through ALSA libraries of the operating system.

In this simple scheme, it can be observed that the server does not have to manage any of the requests made by higher level services, which are controlled by the client, and simply makes the request response function of the commands sent by the client. These functions are offered by voice tools and voice server to the dialogue module through an XML.

The server is designed with the premise that it must operate independently of the software platform used on the client, consequently the manner in which data and commands to use server features are sent, as well as data and responses of the server is through an XML file format (see Appendix B). This file will provide a way to send with a simple structure the commands and data required by the server, as well as a way to return the response data in a structured way. In the following sections it is explained deeply each of the previous blocks.

## 3.2. Client

The client, as it has been commented previously, is responsible of managing services requests, as well as maintain a dialogue to make the voice system richer, and not only to offer instructions with command and control capabilities. An example of this last one could be a mechanism for interaction a push to talk button and subsequent launch of a single command, such as radio control volume up. The client of this project handles requests of services and performs the actions necessary to fulfill these requests, using the functions offered by the server, and configuring its services.

This client needed to achieve the software specifications of Caring Cars and MARTA projects that are designed as an open, modular, reusable architecture focused on the vehicular environment. This software architecture allows the rapid development and integration of services with guaranteed resilience and reliability requirements. Focusing on these requirements, it was chosen a software platform based on middleware by the Open Services Gateway initiative (OSGi).

### 3.2.1. Java OSGI

Java OSGI is based on a Java-based framework for the development, deployment and monitoring of services and provides the common functionalities to the logic of the service applications. The OSGi framework brings to the project a dynamic component model, so that applications (known as bundles for deployment) are developed independently but easily integrated into the common platform. Applications are communicating with one another in a very flexible manner (either by invocation or through categorized events) and are updated dynamically with the tools provided by OSGi platform (see [11]).

## 3.3. Client-Server Communication

In this section it is explained a vital part, the client-server communication. As it has been commented previously, commands and data are send trough an XML file, this file is sent to the server by socket, encapsulated in a transport frame with a format previously used by TID. We have used some TID libraries developed for this kind of communication, programmed in C/C++ language. To achieve the delivery on time of the first use case and to optimize the server development time, it is chosen to integrate and use of these libraries on the Java OSGI client as is now in C / C + +, with Java Native Interface framework (JNI), leaving as a future work the development of the libraries in Java. Once the XML file is embedded in TID format, it is sent to the server. It extracts the file using the same libraries.

Taking into account that voice functions require an elevated execution time, and that at the same time the server needs to be available to execute other functions or to stop the ones that are being executed, it was decided that the server executes them with a thread. Once a function execution is initiated, it returns a code indicating if the function starts correctly or not (see Appendix A). This code is also sent in TID format, while the server initiates a thread that executes the requested function. This execution keeps running and once it finishes it returns information indicating the operation end and if needed errors during the operations or with the data to return. This response is sent directly in a flat XML file Fig. 3.2.

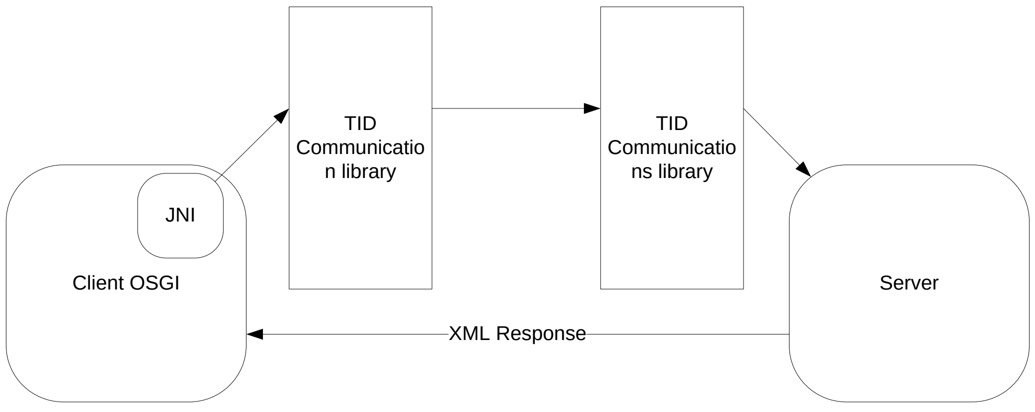

**Fig. 3.2** Communication between client and server

### 3.3.1. JNI

The Java Native Interface (JNI) is a programming framework that allows Java code running in a Java Virtual Machine (JVM) to call and to be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages, such as C, C++ (see [12]).

In our case, this allows the client to access to TID communication library methods. It is done with some simple interfaces to JNI added to TID library communication, which execute the function calls in a simple way. The structure to perform a call to any method of these libraries is J'name' of the function to execute, without the need to include any attribute in functions that are not needed, and if the function require data passing, it is done including the corresponding XML file to the function that executes the method.

Jstart();

Jrecono("xml");

Once the method is executed, it is used C + + libraries, Fig. 3.3, to send the execution order and data in TID format to the server and wait for the response code for this, which is returned after the execution of JNI method. In this way the client is able to know whether the transaction is executed properly.

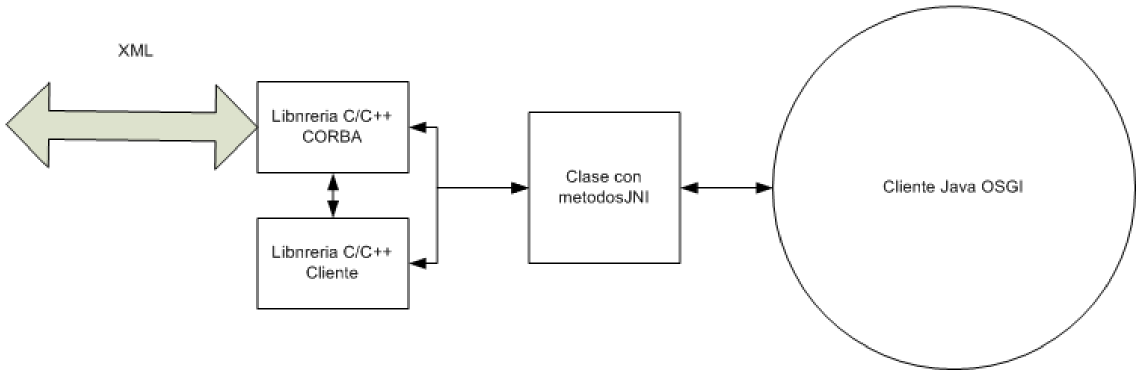

**Fig. 3.3** Client OSGI diagram

### 3.3.2. TID Communication Library

There exist two problems when sending XML files, which do not have a fixed size, in any application where there is a client-server communication. Whoever receives the file needs to know how big is it in advance, to reserve enough memory to read it, and to verify that the data is the expected to receive and that it is correct. In this section we proceed to explain schematically how TID libraries deal with these problems.

TID Communication Library offers an own communication protocol to send data. The frame to be sent contains two items that are two buffers in which the data is flattened (concatenating the sent data byte by byte). We designate the buffers as buffer A and buffer B. Initially, the buffers are empty and each time a data is flattened and added, buffer size is increased and then the data is copied there, Fig. 3.4.

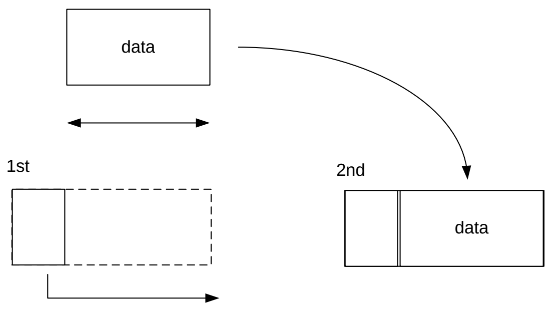

**Fig. 3.4** Data flattened into buffer

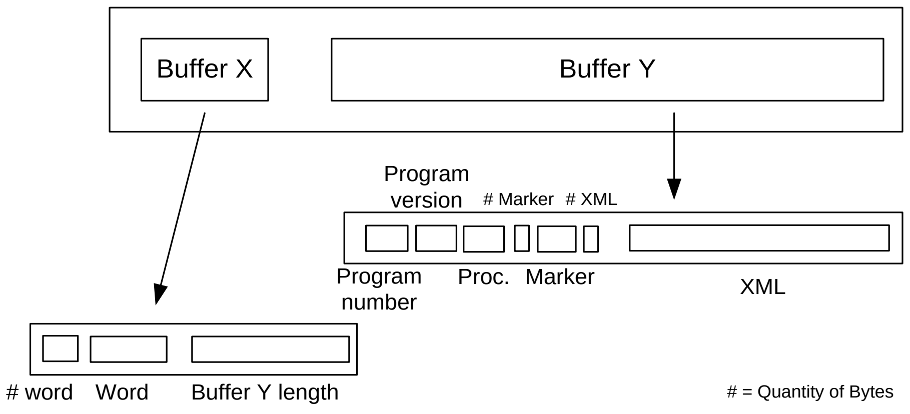

**Fig. 3.5** Frame with two buffers

In buffer B, the first thing added is a header with information of to the communication and application, Fig. 3.5. The header is formed by 3 integers with a fixed length (4 Bytes each) and a word:

- Number of program and version: it is used to make the library compatible with several programs, in this project it is always the same value.
- Procedure number: in this project it is the function that the server has to execute.
- Marker: it is a word that indicates the end of the header and beginning of data. As this word is flattened, and it is of variable length, first it is added an integer that indicates the length of the word followed by the word itself.

After the header, there is added to buffer B the data. In this project it is the XML file, that also has a variable length, and like the marker, it is preceded by an integer that indicates its length.

Once the buffer B is built, it can be formed buffer A, which has a known size fixed when the design is build. It contains a word, added like the marker, and an integer with the size of buffer B. The word is used to check that it is buffer A and that is correctly received. The integer indicates buffer B size, including the header and data.

When both buffers are formed, they are copied into a frame, and it is sent. This frame contains first buffer A, with a fixed length, and after buffer B which has variable length.

When a frame is received, first of all it is read a known fixed length of Bytes, which is buffer A. From this buffer it can be obtained buffer B length. When buffer A is obtained, data is unflattened. Each time some Bytes are unflattened, it is moved a pointer that shows the next Bytes to be read in the buffer, Fig. 3.6, in this way it can always be known the amount of Bytes that are need to read.

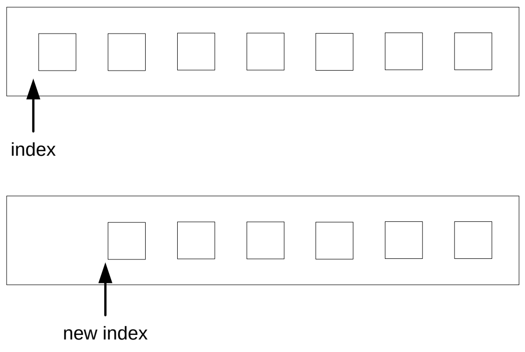

**Fig. 3.6** Data unflattened

The first thing that needs to be read in buffer A is a word. By definition, a word has a variable length, so when this system is used it is accorded a word and the first information read is an integer containing the word length. After, it is read the word and checked that it indicates that it is buffer A. If it is correct, it is read the next value that is an integer (4 Bytes) that indicates buffer B length. Thus it can be read the remaining frame, buffer B.

Once the second buffer is obtained, it is read the three integers: program, version and procedure and it is also read the marker. It is checked that program, version and marker are correct, and after it is obtained the procedure number to execute. Finally, in a simple way it is obtained the XML file, which is next the data in buffer B. It contains an integer indicating the length of the XML file and after the file itself.

With this procedure it is obtained a communication protocol of variable length file, with which it is checked the correctly receptions thanks to a control key word.

### 3.3.3. Finally Server Response

The final response from the server has been performed in a simple way without using TID libraries because of the need of a first functional case of use in time. Due of it is lost the advantage that the receiver knows the size of the data and verifies the integrity of them. It is planned that in a future these responses are also server through TID library. The response is done by simply opening a socket with the client and sending the XML file in plane through the opened socket Fig. 3.2.

## 3.4. Voice Server

Once it is exactly known the data received by the server, it is going to be explained which use does the server with them to execute the required functions by the client.

Voice Server has been designed with the aim to offer the customer all functions of TID speech technology library and various libraries that will be added in the future as required by the services needs. In the present state of development, voice server incorporates several libraries Fig. 3.7. TID speech technology libraries offer voice recognition capabilities, management of vocabularies and text to speech conversion, it is also used the audio hardware access to these libraries to offer the ability to play a sound and finally a library KIVOX of Agnitio for speaker verification. The server architecture allows easily the incorporation of further libraries to increase the functioning possibilities of voice server.

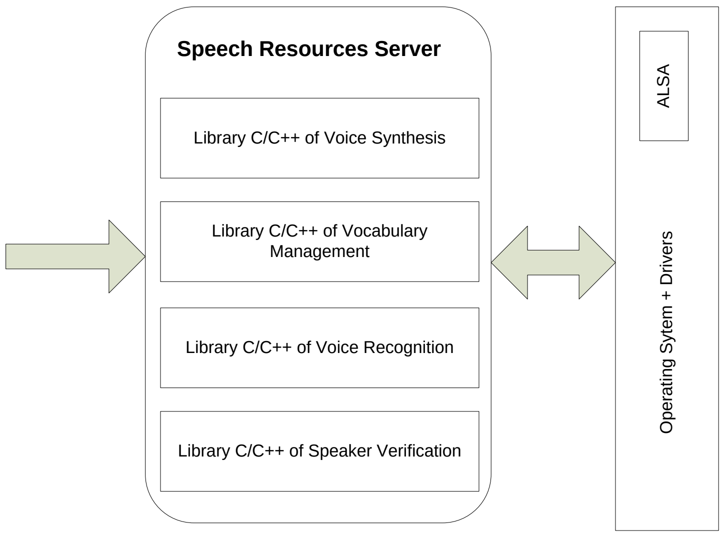
Speech Resources Server
Library C/C++ of Voice Synthesis
Library C/C++ of Vocabulary Management
Library C/C++ of Voice Recognition
Library C/C++ of Speaker Verification
ALSA
Operating Sytem + Drivers

**Fig. 3.7** Voice Server Libraries

Server functions can be joined in three big groups, the ones related with text to speech, the ones with voice recognition and the ones related with speaker verification, bearing in mind and extra capability of playing sound.

### 3.4.1. Text To Speech (TTS)

Within the three large groups, text to speech converter turns into artificial voice the text that is sent. This converter has different configuration options that are available to the client, they are:

- Choose a male or female speaker.
- Audio speaker frequency of 8/16 KHz
- Where to extract audio, in vehicle hardware or in a file. If it is selected in a file, it should be introduced where to save it.

Although TID converter is multispeaker and multilingual it is decided in both projects to include only one male and one female speaker of Spanish language. This is due to hardware limitations and that having a greater number of speakers would imply a substantial increase in required server memory.

The server, to carry out a conversion, will receive an XML file with the text that has to be converted, after the text is received the server and executes the necessary actions to play by the vehicle's audio the text sent. Once the conversion is finished, it would be sent an XML file indicating that it has completed successfully or indicating whether it has ended because of an error (see Appendix B).

This reproduction can be stopped at any time by sending the specific command. It is also studied to offer the possibility of pause and resume later the conversion, and although it has been added these features to the server, they have not been implemented since they have not been requested by any service in any of the two projects.

### 3.4.2. Automatic Speech Recognition (ASR)

Like the TTS, automatic speech recognizer also offers several configuration possibilities:

- Vocabulary in use
- Maximum length of sentence/command to recognize
- Maximum initial silence
- Maximum silence between words
- Maxim duration of voice pulse

The vocabulary used is directly related to the application or service which needs the recognition, therefore agreed with colleagues in TID Madrid that they provide this vocabulary. It has also to take account that these vocabularies must be of a limited number of words for a simple service in the automotive environment, which should be much more limited than in a normal application, both for technical reasons in speech recognizer and for usability reasons according to the driver. These vocabularies are a series of files with a specific format that must be created previously by a TID vocabulary management library, which will be accessible to customers so he can generate them directly. The other settings are times required to adjust the recognizer, so adjust its operation to the vocabulary used. The adjustment of these times will be in charge of the customer, who is the responsible to manage voice tasks.

The recognition begins when prompted and then it starts to record audio and processing. After completing the recognition an XML is returned with the result of it, including also the five most likely candidates, which will be the words or phrases that the recognizer considers more likely to be correct. This type of response has been agreed jointly with colleagues in of TID Madrid as the client is the responsible of managing the dialogue, so he has more data than the recognizer to discern which candidate is correct. Like the converter, if there is any error in the recognition it will also return a XML indicating it.

### 3.4.3. KIVOX Verifier

AGNITIO verification libraries provide their functions to validate the driver or vehicle user and thus for example be able to load adapted acoustic models for speaker and the vehicle's environment and increase the correct recognition rates. To use verification, first it is necessary to realize a training of the various speakers that will be verified. Once these trainings are done, each time it is need to verify a speaker it will be done a comparison of it with the corresponding model (file that is used by KIVOX library to verify), and will be returned if the verification is affirmative or negative, and an error code if there is any anomaly in the process.

To perform the two different actions it is only be necessary to send an identifier of the speaker who performs the training or verification. There is no need to send any more information as this will be responsible for activating the audio recording for training or verification acceding vehicle hardware, the server will store and manage the trained models.

### 3.4.4. Sound Reproduction

Sound reproduction feature was incorporated to meet MARTA requirements, and simply consist of playing a sound (beep) stored in the computer taking profit of vehicle hardware access by speech technology libraries. There is no option to choose which sound to be played although it does provide a little configuration of how it is played to meet the requirements of the project:

- Number of repetitions of the sound
- Time of silence between each repetition

Once seen the different functions blocks provided by the server, we will see a list of all the functions provided by the server, these will be sent together with data when needed with XML in the format described in Apenndix B. When it is not necessary to send any data it is sent a frame to the server, in TID format, containing the operation code to execute leaving the data field empty:

- Check speaker: We verify the current speaker, comparing with previously trained models
- Start Server: Initializes the server
- Stop Server: Terminate the server
- Configuring TTS (Text To Speech): Sets the server TID, language, locator, etc
- Start TTS: It sends a text to convert it in voice
- Stop TTS: Stops text to speech conversion
- Setting of ASR (Automatic Speech recognition): Sets recognizer, times, etc
- Start ASR: Starts a recognition
- Stop ASR: Stops in progress recognition
- Play Sound: Provides the ability to play a sound
- Train Kivox speaker model: We train a model to perform a subsequent verification
- Verify speaker: verify the current speaker, comparing with previously trained models

## 3.4.5. Voice Server Software Architecture

This section is based in server architecture, it also comments actions that each module realizes within the server. In Fig. 3.8 can be seen a block diagram with the complete architecture of the server.

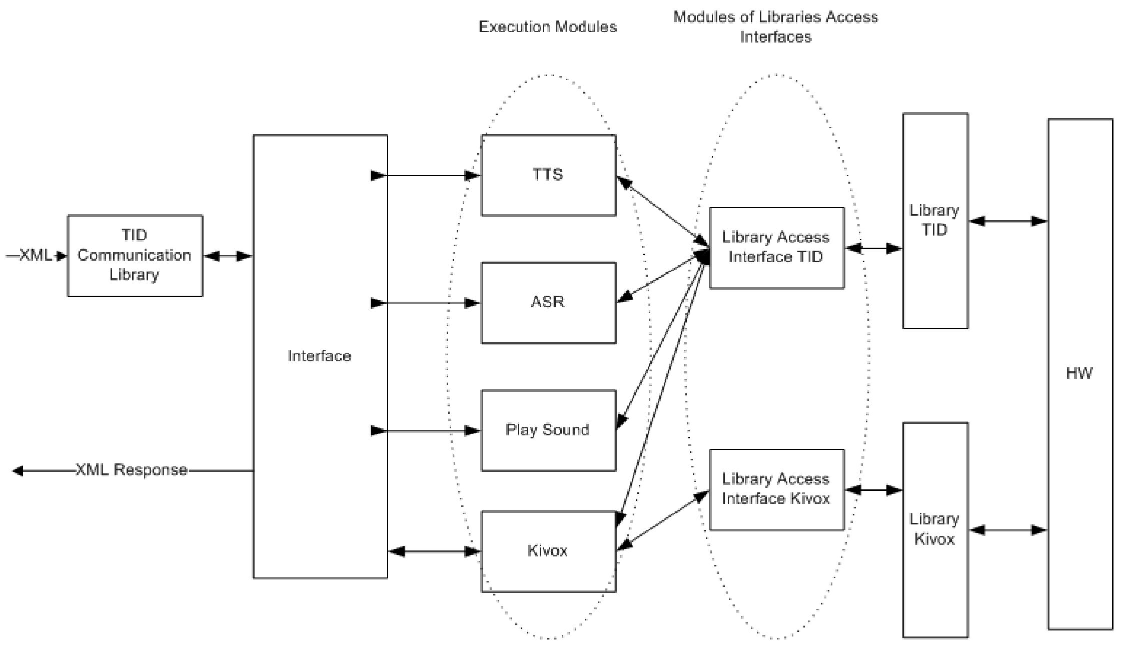
Execution Modules
Modules of Libraries Access Interfaces
XML
TID Communication Library
XML Response
Interface
TTS
ASR
Play Sound
Kivox
Library Access Interface TID
Library Access Interface Kivox
Library TID
Library Kivox
HW

**Fig. 3.8** Diagram of Voice Server Software Architecture

Module of TID Communication Library is always listening in the port pointed in the server configuration file Fig. 3.8. This module contains TID communication libraries, which as has been explained in previous sections once a frame is received it is unflattened and data and execution code are extracted.

### *3.4.5.1. Interface*

This code and data are received by the interface, that is where they are processed. The interface is responsible for sending the execution to execution modules of the required function according to the code received. Also, in case that the frame contains an XML file it is unflattened through TID communication libraries and passed it to the required module. Once sent the execution to requested module it is created a frame with TID format and waits for the response of the execution modules with the operation result. After this response is received, it is dispatch the response through TID communication libraries and is for the execution of new functions.

### *3.4.5.2. Execution Modules*

These modules, that are called by the interface, are the ones that initiate the execution of the requested operations. There exist four modules, one for each block of the functions provided by the server, TTS, ASR, play sound and speaker verification.

These modules initiate the execution of functions through libraries access interfaces, followed by the sending of information about if the execution was successful or not at the interface. It is made a different treatment depending on the time required for implementing each function. In functions that require a high execution time (Start TTS, Start ASR, Play Sound, Train speaker model and Verify speaker) before starting the execution of the operation itself through libraries access interfaces and wait the operation result, it checks if access to this service is available and sends the information to the interface so that it do not remain locked during the duration of the operation and can handle requests received by the server. If the service is available it starts a new thread that is actually the one that executes the operation and waits for its result. In this way all modules are available to perform new functions or be ordered to stop them.

To prevent access to the functions during the execution of a previous one, flag remain in each module indicating the current use of the functions. This becomes necessary because we are accessing the functions that run on the vehicle's audio resources, which can only perform an action at once.

These modules also get the XML file unflattened and are responsible of process the XML and extract the data from it to send to the libraries access interfaces.

### *3.4.5.3. Modules of Libraries Access Interfaces*

These interfaces give access to the various functions offered by each library to execution modules. Are the instruments through which execution modules communicate with libraries as interface mode. They maintain a libraries state, reactivating its services if necessary.

There are two interfaces to accede to libraries. The first one is an interface that accedes to speech tools library and contains all functions for TTS and ASR. It also provides access to audio objects through ALSA libraries in the OS (Operating System), giving access to audio play or speech recording for later use by KIVOX libraries to perform functions of models training and speaker verification.

The second interface accede KIVOX libraries, these libraries do not have access to vehicle’s audio hardware so, as discussed in the paragraph above, the audio sent to these libraries is to be obtained by TID libraries. KIVOX libraries before performing an operation check that the audio is invalid return an error message if not.

#### *3.4.5.4. Conclusion*

This disposition in the server architecture allows us to incorporate in a simple way new functions and libraries in the future, without having to modify the existing structure. If it is desired to add a new library, it must be added a new interface to libraries access module that allows communicating with it and performing all functions required of it. This interface will have an execution module that will be called when needed some function of the new library and will have to process the data to perform it.

## 3.5. Characteristics of Libraries Used

### 3.5.1. Voice System Telefónica I+D

#### *3.5.1.1. Speech recognition*

The TID recognition system has the following general characteristics:

- It is speaker independent
- It is independent of the microphone and the telephone line
- Based on hidden Markov semicontinuous models
- It is a continuous speech recognizer in Spanish
- Handles a vocabulary of about 2000 words
- Uses statistical models of language
- Use context-dependent units to model vocabulary words

Speech recognizer performance is based on combining information from acoustic models and statistical models of language, which are described below. In Fig. 3.9 it can be seen the flow diagram of the recognizer.

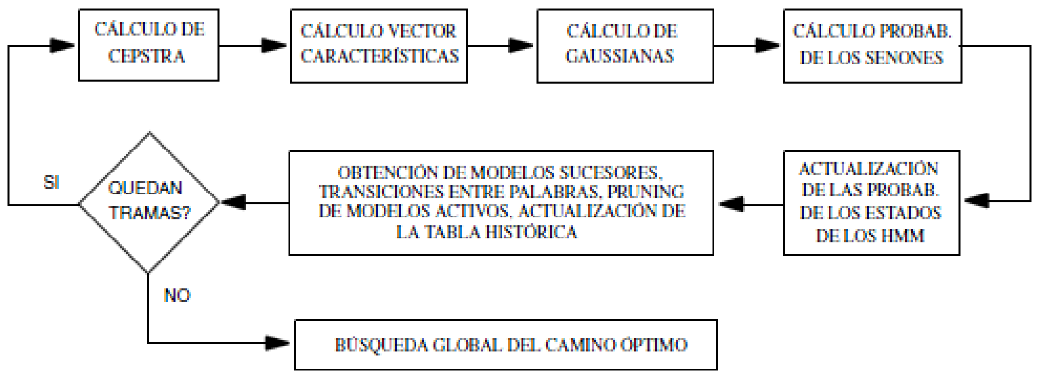
CÁLCULO DE CEPSTRA
CÁLCULO VECTOR CARACTERÍSTICAS
CÁLCULO DE GAUSSIANAS
CÁLCULO PROBAB. DE LOS SENONES
ACTUALIZACIÓN DE LAS PROBAB. DE LOS ESTADOS DE LOS HMM
OBTENCIÓN DE MODELOS SUCESORES, TRANSICIONES ENTRE PALABRAS, PRUNING DE MODELOS ACTIVOS, ACTUALIZACIÓN DE LA TABLA HISTÓRICA
QUEDAN TRAMAS?
SI
NO
BÚSQUEDA GLOBAL DEL CAMINO ÓPTIMO

**Fig. 3.9** Recogniser block diagram

Acoustic modeling is performed using context-dependent units (triphonemes). Each triphonemes is represented by a hidden Markov model of five states that allows three sorts of transitions from each state:

- Transitions to departure state
- Transitions to next state
- Transitions jumping next state

Stochastic Grammars

Stochastic grammars allow, in principle, all possible combinations between words, assigning a probability to each.

It has to be noticed that the recognizer used in all experiments is multipass type, with a first pass in which makes a purely acoustic recognition by hidden Markov models, followed by a second in which is applied the knowledge gathered in the language model used.

The language model (LM) used in all tests is a statistical back-off category, based on trigrams consisting of groups of three words in which a central word is accompanied by the immediately before and after. This LM is an open type in which any word can be followed by any other.

The number of unigrams (single words) specified in the LM is 378 (the 375 words that comprise the vocabulary plus 2 labels of start and end of sentence, and 1 unknown word tag), the number of bigrams (pairs of word) is 1827 and the number of trigrams is 2791.

### *3.5.1.2. Text to Speech*

TID System is a multilingual multispeaker TTS, based on unit concatenation, which employs a technique of corpus selection. This system uses dynamic programming techniques for both the selection of acoustic units to the selection of intonation units. So far, the languages included into the TTS of TID are Iberian Spanish, Catalan, Galician, Euskera, Iberian Portuguese, Peruvian Spanish, Mexican Spanish, neutral Latin American Spanish and Brazilian Portuguese.

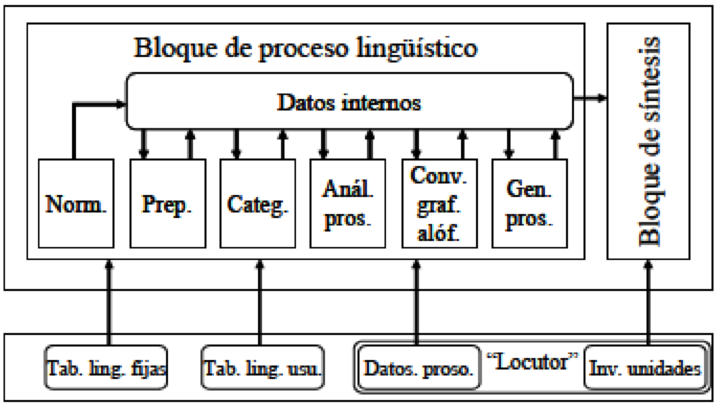

**Fig. 3.10** Structure of TID TTS system

The acoustic units handled are diphonemes which generally contain the speech signal interval between the stable part of a sound and the stable part of the next sound. The components of the resulting synthetic speaker are:

- Data needed for the procedure for determining prosodic boundaries used by the prosodic analysis module
- Parameters of sounds durations model
- Inventory of accent groups in the construction of contours
- Diphonemes inventory

The inventory of accentual groups contains 4940 items. Of these, most correspond to groups extracted from sentences in declarative mode (4707). Of the remainder, 212 belong to interrogative mode sentences, and only 21 to exclamatory mode phrases. The diphonemes inventory contains a total of 38004 units, which contain 420 different diphonemes identities (assuming that the identity is given by the label of the initial and final sounds of diphonemes). The variants of each identity vary in number, from the 780 of the most common unit [D-e] to the case of identities with a single variant (there are 22 identities with just one variant).

### 3.5.2. Agnitio Verifier

Kivox library from Agnitio is not like most speaker verification technology used in enterprise speech applications today that are text-dependent. Essentially, this is a word for word comparison of what was said during verification to the exact text used during enrolment (phrase, password, etc.) The flexibility of these systems is very low, as a new enrolment is required when, for instance, your password is changed or the user wants to use a different phrase. Agnitio's Free speech solutions instead asks the user to enrol anywhere between 30 seconds and 2 minutes of free speech, one time. This enrolment can be guided or completely unstructured (just tell the user to start talking or engage him or her in a conversation). Verification options are almost limitless, ranging from asking the user to answer a few questions, to repeat a completely random phrase or provide a shared secret in a Multi-Factor Authentication process. As long as 3 seconds of audio is captured and passed to KIVOX Verifier, a decision can be made regarding the person´s identity claim.

# CHAPTER 4. Speaker Adaptation

In this chapter it is explained my job developing speaking tools in TID, it is doing whilst developing the server program. My task has been centered in testing different strategies about the use of different algorithm for the speaker adaptation, exactly the adaptation of independent speaker acoustic models to a particular speaker to increase the rate of successful recognitions by ASR. This task is the continuation of some tests started from TID colleagues.

To introduce about speaker adaptation topic, this section explains the problems that make necessary this adaptation, and techniques and algorithms used in tests. Also includes a brief summary of the previous test realized, finally present the results and studies them to obtain which strategies to follow to improve the correct recognition rate in a car environment.

## 4.1. Speaker Variability

Most state of the art speech recognition systems being Speaker Independent (SI) can handle a variety of speakers both male and female with low average error rates. However some speakers result in very poor performance compared to others. This occurs because a number of different speakers are used in training the original speech model. This leads to speech models that may not accurately model speech units for a particular speaker but are good on average. A comparison of SI systems to Speaker Dependent (SD) systems shows that the SD system may have 2-3 times lower error rates compared to a SI system.

Moreover, mismatches between training and testing environment severely degrade performance. Two major sources of mismatches are speaker and environment variability. Speaker variation is typically caused by different speaking styles and other physiological differences between speakers, such as vocal tract lengths, etc. Environment variability includes channel distortion, such as that which affects telephone speech, additive noise, and reverberation which results when the microphone is far away from the speaker. Since a large amount of data is needed to train a good speech recognizer, it may not always be possible to obtain this amount for every speaker. The problem then is to obtain a good SD model with very little data. The solution to this is Speaker Adaptation in a SI system.

## 4.2. Acoustic compensation

The main objective of the work is to develop acoustic adaptation technology that allows that recognizers used in car application have the ability to improve their recognition rates.

The techniques currently used to compensate the effects of speaker variability can be divided into three main groups [4]: adaptation techniques, normalization techniques and adaptive training techniques. Of these three types of techniques, in the project have been selected the adaptation techniques, because they are the ones that get further reductions in error rates [4].

Adaptive training techniques consist in jointly estimating parameters of hidden Markov models and parameters of an affine transformation of the vectors of means. This process results in a set of models that are less dependent on the specific characteristics of each user's voice. During the recognition phase, applies a transformation to independent models to make them into dependent of the new speaker.

### 4.2.1. Acoustic Adaptation

There are several modes of use of the acoustic adaptation platform, which differ primarily in how to get the transcripts necessary to carry out adaptation, and when the recording of locutions is done. Thus, it is necessary to differentiate between an adaptation:

- Supervised and unsupervised: if you have the verbatim transcriptions of the speaker it is supervised, however, if there is no transcription of the locutions, if it is not used the output of the recognizer using speaker independent models, adaptation is considered unsupervised.
- Invisible and Visible to user: If the locutions that are used for adaptation are read by the speaker it is understood that this adaptation is visible to the user. In the case that is invisible to the user, in successive interactions, the system will learn the characteristics of the speech of users without the intervention of these.
- Offline and Online: This is related if the adaptation is performed in real time, while the driver interacts with the system, online, or offline if performed when the vehicle is stopped and the user interacts with it.

## 4.3. Adaptation Algorithms

### 4.3.1. MLLR Algorithm (Maximum Likelihood Linear Regression)

In this section it is explained briefly the method MLLR of speaker adaptation applied to continuous hidden Markov models. The method is based on the use of linear[3] regression techniques and part of a speaker-independent system. This system is used to generate statistics on the differences between the

[3] mathematical method that models the relationship between a dependent variable Y, independent variables Xi and a random term ε

features modeled by the system and features of the new speaker. Finally, statistics are used to update the model parameters, resulting in a system adapted to the speaker.

This method allows a global transformation of the model parameters even with few adaptation data. The adapted models are so much better the larger the database of adaptation, as it is possible to estimate a larger number of transformations. The basic assumption of this method is that the main difference between two speakers lies in the position of phonemes in acoustic space. Under this assumption it is sought transformation matrices of the vectors of Gaussian averages so as to maximize the probability that the adapted models generate adaptation.

### 4.3.2. MAP Algorithm (Maximum a Posteriori)

This section outlines the method MAP, this method is as Bayesian adaptation, since it involves the use of a priori knowledge about the probability distribution of the parameters to adapt. So if you are able to predict the general behaviour of these parameters before you rely on any kind of adaptation data can be used efficiently to adapt information to obtain a MAP estimator acceptable.

It has to be taken into account that if a priori information which is taking in consideration has not shown any tendency in the distribution of the parameters, the MAP estimator will coincide with that obtained using a maximum likelihood method such as MLLR. In the application environment adaptation to the speaker, this a priori information is usually contained in the speaker independent models.

An obvious drawback of MAP adaptation method compared with MLLR is that it needs more adaptation data to be truly effective, since the MAP estimator is defined at the component level, so those components for which no data are available adaptation will remain unchanged.

But on the other hand the advantage is that MAP offers a superior performance to MLLR when you have large amounts of adaptation data, precisely because of this detailed update of each component is not available in MLLR, which implements common transformations to Gaussian groups.

The two techniques can be serialized, i.e. one can apply MLLR after MAP. Doing so, it is possible to take advantage of the different properties of the two techniques.

## 4.4. Speaker Adaptation Tests

The tests that I do are based and also continue a job realized previously, with little differences about how we want to test. To comprehend more the differences, the reader has to know that in requirements approved in MARTA and Caring Cars at the moment of this report, the adaptation will be offline, when the car is turn off, in order to avoid overloads in PCCAR.

The test done previously centered not to proof the performance of different algorithms, which have already been proved with several studies that demonstrate their performance, but to evaluate the performance of the algorithms MAP and MLLR in car environment conditions where this techniques has never been proofed, the study of intelligent selection of adaptation corpus, that consist in choosing the sentences to use to adapt that follow a series of rules, as well as the possibly to use speaker adaptation as a environment adaptation obtaining a new independent acoustic models.

With these previous studies it was tested the operation and adjusted some parameters to optimize the performance of the speaker adaptation algorithms to car environment, as well as the number of transformation to do related with the number of data that we use to adapt in MLLR and an internal parameter in MAP algorithm.

Regarding the selection of the corpus of intelligent adaptation, different tests were conducted, as the study of potential profits by incorporating estimates of confidence thresholds, confidence percentages obtained by the TID ASR, to improve supervised and unsupervised adaptation, or the application of reduction techniques on transcripts unsupervised for improving them, as extraction of monosyllables of 1 and 2 letters, among others. During the realization of these tests was observed that the combination of MLLR+MAP algorithm is the obtained usually better results, thus improving the use of two methods separately. About the intelligent selection of the adaptation corpus, the results have showed that no improvement was obtained in this selection. These initial tests also showed the need for a minimum number of sentences for improve recognition rates in the adaptation. Regarding the possibility of using the speaker adaptation as an environment adaptation, adapting the independent models of recognizer to a series of speakers, all in the same environment, and getting some new independent models by means of each speaker adapted models, with the obtained results we can observe that it is possible to make an environment adaptation through the speaker adaptation algorithms.

The following tests, which are what I have done, try to check if it is possible to achieve improvements by several adaptations with little data and can thus consider the possibility of do speaker adaptation online, which would enable that the recognition system will be adapted to the changing daily environment of the car, sunny, rain, wind, etc, because as has been shown in the previous tests, speaker adaptation techniques are also capable of adapting to a new environment, as well as to perform adaptation without the need to wait for a large amount of data, often difficult in the automotive environment, where the vocal interactions with the ASR and the user will normally be minimal.

### 4.4.1. Tests Environment/Background

In this chapter it is specified the aspects of recognition basis from which have developed the tests, regarding the adaptation database and the evaluation used and the characteristics of the models, vocabulary and language model that made recognition system which will be applied to speaker adaptation.

#### *4.4.1.1. Database*

As in previous tests, in this phase there is not available a database recorded in car environment, so it is used a telephone database, like the reference database with which were implemented speaker-independent models and speaker recorded in a microphonic environment, to check the behaviour of the adaptation to speaker in a different environment to the reference.

To perform the experiments it has been used a subset of the reference database TRESVOL, developed in TID in 1999 with a vocabulary of words and numbers, consisting of 2 speakers (1 male and 1 female) with all their corresponding pronunciations (120 words each, 60 sentences and 60 numbers), recorded in a telephone environment. The speaker recorded in a microphonic environment has been carried out with the recording of the same 120 sentences.

The main characteristics of the three speakers are:

- Speaker Fanny has a voice that deviates from the average, so that speaker adaptation techniques are quite necessary to use recognition systems.
- Speaker Luis, who achieves results with the speaker independent models with word hit rates acceptable for most services.
- Finally, speaker David, that reaches a fairly low success rates. This may be due mainly because the speaker has been recorded in a microphonic environment.

Sentences have an average duration of 4-5 seconds, and are formed on average between 5 and 6 words.

#### *4.4.1.2. Vocabulary*

Usually, in speaker adaptation tests it is intended to simulate a task management for telephone services, with natural language interaction. But to carry out these tests vocabulary has been reduced from the original 4079 words to 375 words. The reason is that it is intended to simulate tests with what could be the vocabulary in a simple service in the automotive environment, which should be much more limited, both for technical reasons the speech recognizer, and for reasons of driver usability.

The success rates of recognition in the automotive environment are too low to pose a voice service with natural language interaction with a large vocabulary. So was limited to a service of telephone services management, including the possibility of giving phone numbers.

Moreover, about the usability of the driver, voice interaction services must be well defined and delimited, and that the driver knows what the purpose of the service is, and therefore the automatic voice interaction system is an aid, not service generating loss of concentration on driving.

#### *4.4.1.3. Recognition System*

The recognizer on which experiments are carried out is the ASR from Telefónica described in CHAPTER 3.

It is necessary to stress that for the training of this recognizer it has been used a different database to that the used in the experiments of adaptation presented, but also in a telephone environment. Notice also that recognizer used in all experiments is multipass type, with a first pass in which makes a purely acoustic recognition by hidden Markov models, followed by a second in implementing the knowledge gathered in the language model used.

### 4.4.2. Evaluation

#### *4.4.2.1. Tests Description*

The tests carried out to verify the improvements obtained in the hit rate to adapt with little data have consist in follow several different strategies combining the two algorithms and the amount of phrases to adapt. These different strategies are designed to achieve a minimal improvement in hit rate in the first iteration, which lead to obtain a more accurate recognition of sentences, which are the transcripts that algorithms adaptation use and get on the next iteration a better hit rate with little data due to have an accurate transcripts.

The developed tests process has been conducted in two phases, in the first phase it has have been tried different combinations with different numbers of phrases, over several iterations to adapt the models to each iteration, and have been tested with speaker David in supervised mode to prove what strategies are feasible. Turning in a second stage to perform the tests with the combinations that have worked better, unsupervised in the speaker David and the other two speakers in both supervised and unsupervised. The realization of the test in the supervised case enables us to see if there are significant differences between the two cases, taking into account the results we are interested for the project are the unsupervised case, the more susceptible of being used for a system in a car environment.

It has also realize adaptation with different numbers of sentences without any iteration to have some reference values and be able to observe if when making the adaptations in several iterations with little data we get any improvement or otherwise the hit rates obtained are worse than if we make the adaptation directly. Those adjustments have been made in two cases, using the algorithm MLLR and MLLR+MAP combination.

The different combinations chosen are to take the sentences in the receiving order and when reached a certain number of sentences an adaptation is carried out with one of the two algorithms. Then, the process is repeated adapting again models previously adapted with one of the two algorithms or a combination of both of them.

Following, it is the explanation of the combinations used in the first phase. Combinations employed can be distinguished in two cases Fig. 4.1.and Fig. 4.2 Fig. 4.3.

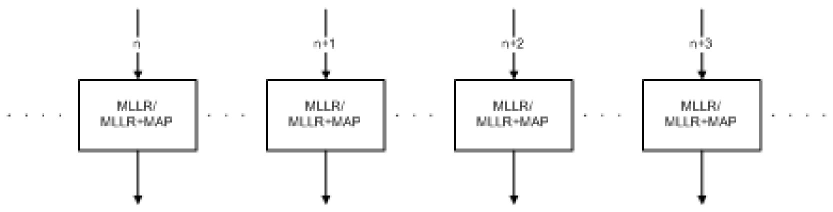

**Fig. 4.1** Combination without iterations

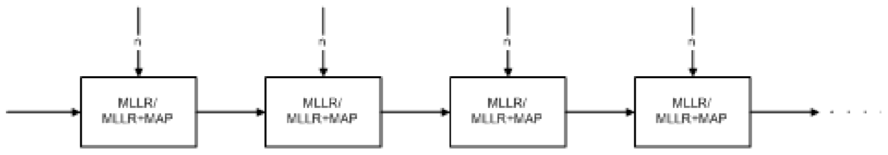

**Fig. 4.2** Simple combination with iterations

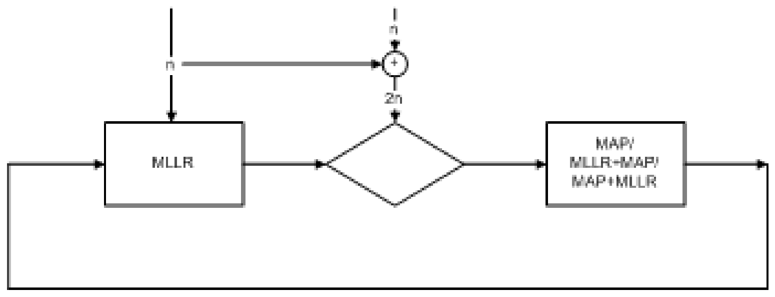

**Fig. 4.3** Combination with iterations

In the first case Fig. 4.1, there exist an adaptation each time of the original independent models with MLLR or MLLR+MAP every time we get a sentence, this will be useful to observe the results of adapting a number of phrases from the originals and in this way compare with the results of the iterations.

In the second case we expect to have n sentences, independent models are adapted with MLLR or MLLR+MAP, wait again n phrases and adapt models with MLLR+MAP. This combination was performed with n=3 sentences and carrying out all iterations with MLLR algorithm and all iterations with the combination of MLLR+MAP algorithms. Finally, the tested combination is wait n phrases, perform MLLR adaptation, wait again n sentences and use as training corpus the current n phrases and the past n sentences, and adapt the models with different combinations MAP or MLLR+MAP or MAP+MLLR. The different combinations tested with this kind of iteration are:

- 3 sentences adaptation MLLR, 6 (3+3) sentences adaptation MAP
- 3 sentences adaptation MLLR, 6 (3+3) sentences adaptation MLLR+MAP
- 3 sentences adaptation MLLR, 6 (3+3) sentences adaptation MAP+MLLR
- 4 sentences adaptation MLLR, 8 (4+4) sentences adaptation MAP
- 5 sentences adaptation MLLR, 10 (5+5) sentences adaptation MAP

Once obtained the results of the first phase, it is decided to select two combinations made of the last case Fig. 4.3, specifically combinations of 4 MLLR 4+4 MAP and 5 MLLR 5+5 MAP, discarding all others because they did not show any trend of hit rate improvement regarding the original independent models.

As discussed in section 4.4.1.1 the database of each speaker has 120 words. Corpus has been divided in two: 40 phrases for the training corpus and 80 phrases for the test corpus. The division and management of sentences has been carried out in a random way, so it can be emulated a somewhat more realistic online sentences entry to the recognizer, which we can not predict the order of entry of the sentences, or the kind of sentences.

The reason it is necessary to perform this separation is due to that if it is used the same phrases to train and evaluate it would bring falsified results, because the hit rates obtained in the phrases that were used to train would be maximum. Since adaptive algorithms have made the adjustment of independent models for these sentences, and therefore for these sentences would get a very high success rate. Separating training and test corpus it can be avoid such problems.

For each test, it is shown the results of recognition of test corpus with the speaker independent models and with models adapted for each of the groups of adaptation presented in the previous section. Thus, one can observe the existence of the improvement on the results achieved with the speaker independent models.

### *4.4.2.2. Results*

In this section there are presented the results of the tests realized. As discussed above, these tests are designed to evaluate the possibility of online adaptation with a low amount of data and apart from adapting the speaker independent acoustic models, to achieve improved success rates in the automotive environment done at that moment. It is based in tests done previously, they showed that speaker adaptation algorithms also perform an adaptation to the environment, an essential part when this may be changing continuously in an environment like the car.

The most interesting results are the obtained in the unsupervised adaptation, since for the MARTA project environment and Caring Cars, speaker adapting services and environment does not have transcriptions of locutions of the driver, so it will need to take full advantage of the unsupervised adaptation. We left the results of supervised adaptation as reference values for evaluating the performance of the different strategies followed.

The results are presented in the same order specified in section 4.4.2.1, first testing with the speaker David in supervised mode, which helps us to evaluate and select optimal strategies to perform the tests with the other speakers. Below are the results of tests with the chosen combinations for the remaining speakers in the supervised adaptation as in unsupervised.

Finally, there is important to point out some considerations about the graphs where the results are:

- The axis of the coordinate X is the number of adaptation sentences used. As can be seen, starts in 0, which are the results achieved with speaker-independent models. Adaptations have been made from 1 to 40 sentences, except in two cases 3 sentences adaptation MLLR 6 (3+3) sentences and adaptation MLLR+MAP and 3 sentences adaptation

MLLR 6 (3+3) sentences adaptation MAP+MLLR in which tests have been done with only 20 sentences from the same training corpus.

- The Y coordinate axis is the %Corr, it means the success rate of speech for the test group defined. Should be borne in mind that scale and extreme values are not the same in all charts, provided to facilitate the reading.

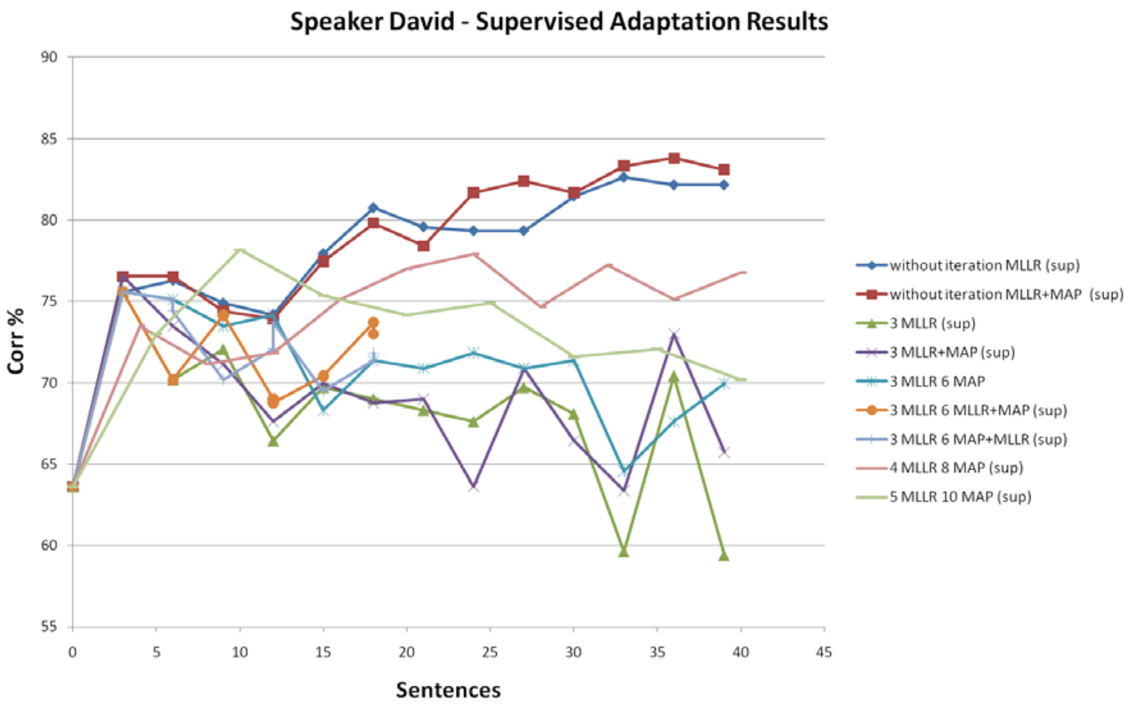

**Fig. 4.4** Results speaker David supervised

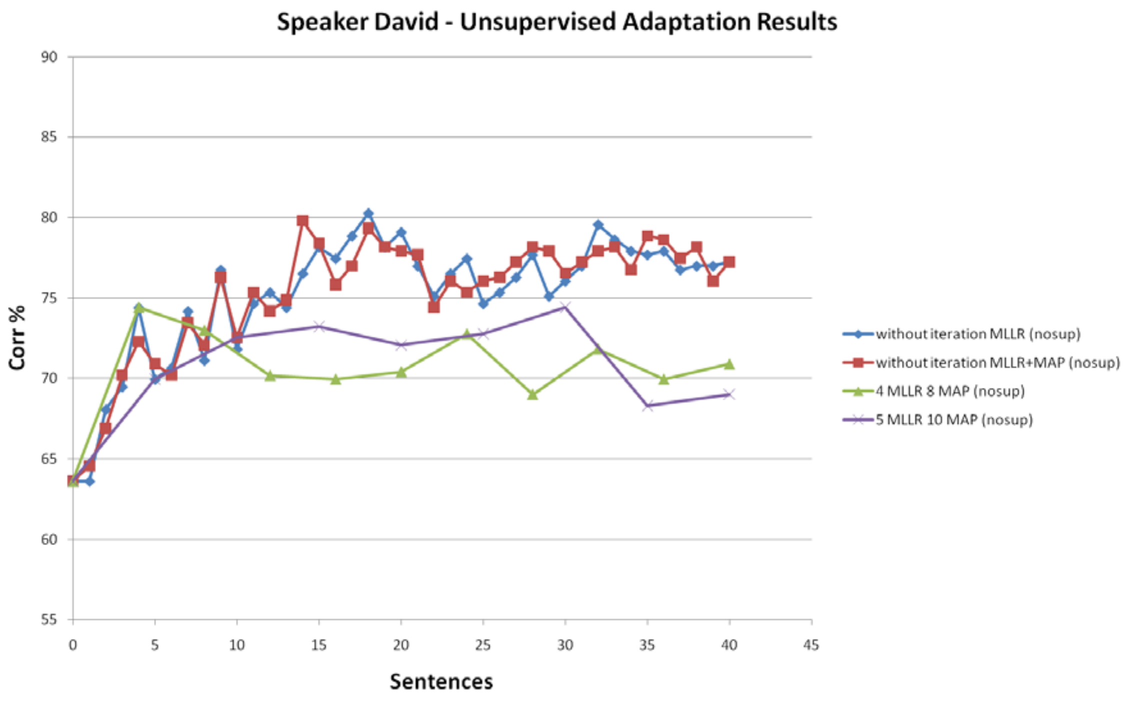

**Fig. 4.5** Results speaker David unsupervised

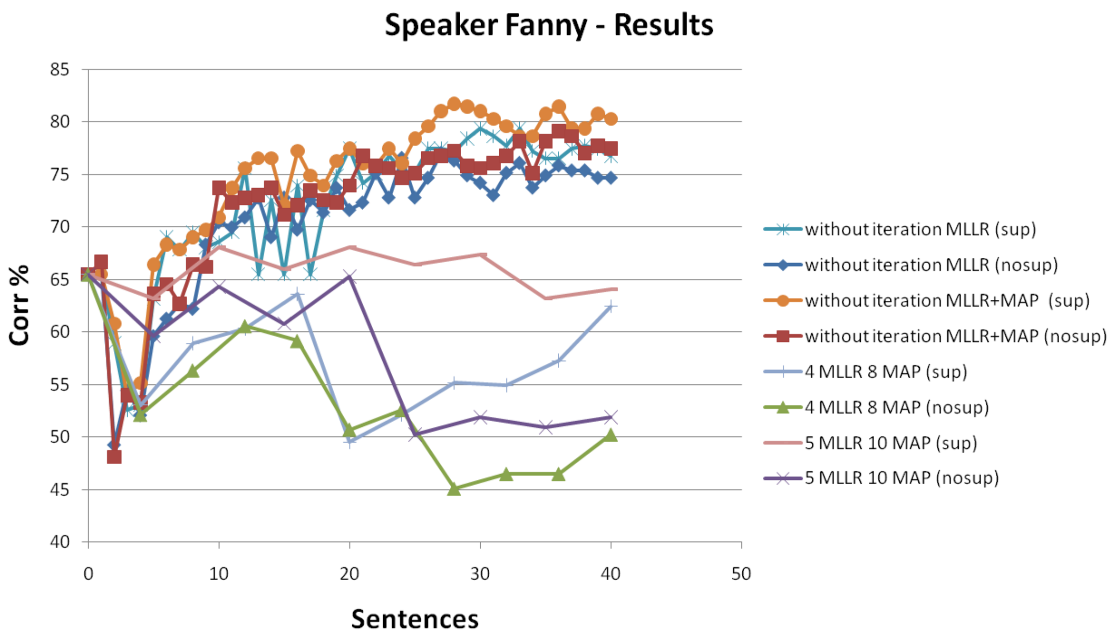

**Fig. 4.6** Results speaker Fanny

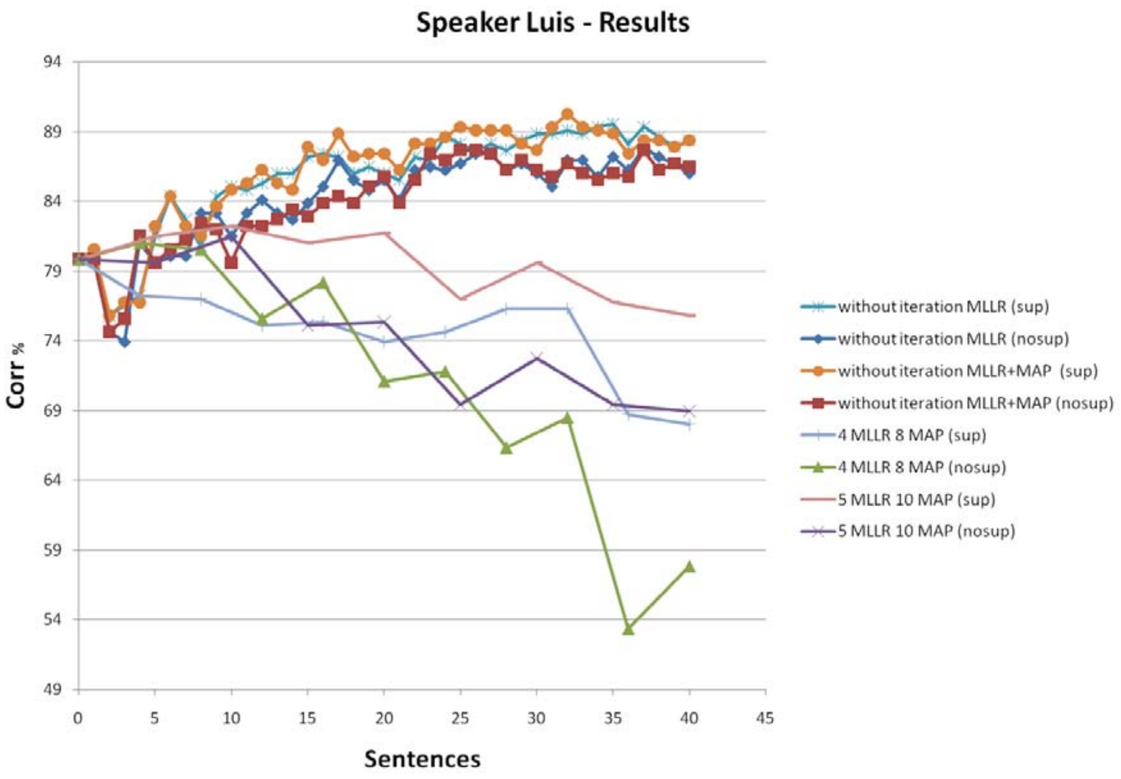

**Fig. 4.7** Results speaker Luis

In the results of the first phase, Fig. 4.4, it can be observed that some of the strategies chosen not only do not improve success rates, but after some iterations the results are worse than the original speaker independent models and for this reason it is chosen to dismisses these strategies, since do not entail any improvement. Others show a very irregular tendency at any time without demonstrate ever a trend of improvement in the rate, so they are discarded too. While the combinations 4 MLLR 8 MAP and 5 MLLR 10 MAP show some valid results for testing with other speakers and adapting models and study the feasibility of these combinations.

The following results show unsupervised speaker David (Fig. 4.5) and speaker Fanny (Fig. 4.6) and Luis (Fig. 4.7) in supervised and unsupervised, . Is can be noticed that these obtained results are lower in most cases even to the percentages of success obtained by the original models, except the speaker David unsupervised, in which there is a slight improvement, although always in all cases the improvement results obtained from a corpus of 10 sentences are much lower than if we made the adjustment directly.

### *4.4.2.3. Analysis of Results*

Once obtained the results of all the speakers, we realize that the possibility of the speaker adaptation with few data, less than 7 sentences, is unworkable as it is not getting any improvement at making adaptations. Contrary to what is sought, transcripts are worsen and thus harms the following iterations to perform adaptation. In principle, we were not seeking that with 40 sentences to get an improvement of %Corr higher from one made directly to the original independent models, but rather to obtain after certain iterations improvements in rates similar to those obtained with the end number of sentences (40), always keeping in all iterations an improvement over the original models, thus improving the recognition and very little data to consider making an adjustment online.

It can also be noted that the results clearly vary according to the phrases used to make the adaptation, which seems to be highly related to the results obtained by applying the adaptation with few sentences. Tests done before the first phase were made with different random training corpus, which was observed more variety in the results, which makes predicting that the 'quality' of the phrases makes adaptation to improve or even deteriorate results. This 'quality', not related to the confidence estimator obtained with the recognizer, as was seen in previous tests, and various trials carried out in the current tests, in which the phrases that were used for adaptation were chosen confidence estimator obtained by the recognizer, but results were worse than with randomly chosen phrases. This leaves for further study the possibility to choice sentences according to this ‘quality’ and thus enables the use of adaptation with few data, being right now the best option to wait for a large number of phrases so that adaptation has a corpus varied enough.

# CHAPTER 5. Conclusion and Future Lines

In this project it has been developed a server able to provide a complete voice system to an application with any programming language. In the current state of the project, the code is ready to offer the simplest use case, that is to serve a single dialogue of interaction with the user, in order to begin testing the two projects, Caring Cars and MARTA, in a global manner. Since real tests in cars can not be made until the projects are in a more advanced state of development, there is not possible to present any result. At this early stage problems have appeared when making the integration of different parts of the diverse companies involved in the project, which has highlighted that communication with other developers to solve these issues is a must. These difficulties have still more importance in the part where there is decided the main guidelines which must follow the projects and the definition of each part so that the pieces finally fit together without much difficulty.

Once finished the first development phase of the server and solved the first problems of integration, the development will be continued with the evolution of the server by modifying its responses to be made with the TID communication libraries and to avoid using C code in the Java client through JNI, providing some new Java libraries to use the TID communication format. Another possible improvement of the server is the incorporation of new functions in case they are requested by services, and ultimately to a bug correction phase until the final delivery.

About speech adaptation, in the implementation phase of the tests we have observed that according to the obtained results, the possibility of online adaptation, with very little data given the small amount of dialogue and vocabulary that can be obtained on application focused to use in an automobile, seems unfeasible according to the results of the tests. Although the obtained results do not have to withdraw because the database used is not sufficiently large to draw accurate conclusions, but to do small tests to guide the tests to be performed when a bigger database in the environment car is available. It is also noticeable that the results are extremely variable depending on the chosen sentences, which makes us assume that perhaps the combination of two different databases, a numerical one and one of sentences is not the most suitable for adaptation. For example, it could be a case of having the training corpus mainly with numerical sentences, and to make the evaluation tests with a corpus mixed with numbers and sentences, so the result will say that phrases recognition has gotten worse while it has actually increased the correct recognition rate for numbers, giving us misleading results as the adjustment would have been really successful. We have adapted numerical sentences, we recognize better numbers, and instead the evaluation results would be bad because of the low success rate with sentences.

These assumptions lead to new approaches to make a specific adaption for each vocabulary and also lead to realize more tests to verify these ideas with a larger database. Besides, it will be necessary to repeat the initial tests with this larger database to extract results and analysis more representatives about speaker adaptation algorithms performance in a car environment.

# BIBLIOGRAPHY

[1] Huang X., Acero A. and Hon H.W., *Spoken Language Processing*, Prentice Hall, 2001

[2] Mosur K. Ravishankar, *Efficient Algorithms for Speech Recognition*. Ph.D. thesis, Tech Report No. CMU-CS-96-143, School of Computer Science Computer Science Division Carnegie Mellon University Pittsburgh, May 1996

[3] Hwang, Mei-Yuh. *Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition*. Ph.D. thesis, Tech Report No. CMU-CS-93-230, Computer Science Department, Carnegie Mellon University, Dec. 1993

[4] D. Tapias, “Speaker Compensation in Automatic Speech Recognition”, Robustness in Language and Speech Technology, capítulo 3, pp. 47-100, Ed. Kluwer Academic Publishers, 2001, ISBN: 0-7923-6790-1

[5] M. Á. Rodríguez, J. G. Escalada y A. Armenta. Descripción Del Sistema I De Telefónica I+D Presentado A La Evaluación Albayzin’08 Para Ctv. División de Tecnología del Habla Telefónica Investigación y Desarrollo, 2008

[6] Speech processing, transmission and quality aspects (STQ); distributed speech recognition; front-end feature extraction algorithm; compression algorithms,” Tech. Rep. Standard ES 201 108, European Telecommunications Standards Institute (ETSI), April 11 2000

[7] Prabhu Raghavan. Speaker and Environment Adaptation In Continuous Speech Recognition. The State University of New Jersey, June 1998

[8] James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, Amanda Stent. Towards Conversational Human-Computer Interaction. Dept. of Computer Science University of Rochester, 2001

[9] Chin-Hui Lee. Spoken Language Systems - Technical Challenges for Speech and Natural Language Processing. Bell Labs' Dialogue Systems Research Department

[10] Caring Cars. Aplicaciones de Salud y Bienestar en el Automóvil. URL: http://www.tid.es/netvehicles/caringcars/portal/home.htm

[11] OSGI Alliance. URL: http://www.osgi.org/Main/HomePage

[12] Java Native Interface. URL: http://java.sun.com/j2se/1.4.2/docs/guide/jni/

[13] C++ Reference. URL: http://www.cplusplus.com/reference/clibrary/

[14] CORBA-Object Management Group (OMG). URL: http://www.omg.org/gettingstarted/corbafaq.htm

[15] awk-Linux Command. URL: http://linux.about.com/library/cmd/blcmdl1_awk.htm

[16] debian Wiki. URL: http://wiki.debian.org/

[17] Advanced Linux Sound Architecture (ALSA) project homepage. URL: http://www.alsa-project.org/main/index.php/Main_Page

[18] RedHat. URL: http://www.redhat.com/

# ACRONYMS

| | |
|---|---|
| ASR | Automatic Speech Recognition |
| CENIT | Consorcios Estratégicos Nacionales en Investigación Técnica |
| CSR | Continuous Speech Recognition |
| GUI | Graphical User Interface |
| HMI | Human Machine Interface |
| HMM | Hidden Markov Model |
| JNI | Java Native Interface |
| JVM | Java Virtual Machine |
| LM | Language Model |
| MAP | Maximum a Posteriori |
| MARTA | Movilidad y Automoción para Redes de Transporte Avanzadas |
| MLLR | Maximum Likelihood Linear Regression |
| OS | Operating System |
| OSGi | Open Services Gateway initiative |
| PROFIT | Programa de Fomento de la Investigación Tecnológica |
| SD | Speaker Dependent |
| SI | Speaker Independent |
| TID | Telefónica I+D |
| TTS | Text To Speech |
| UPM | Universidad Politécnica de Madrid |
| XML | Extensible Markup Language |

# APPENDIXES

# APPENDIX A. VOICE SERVER ERROR CODES

## A.1. General Errors

-3003 // Gestor eventos ya creado
-3004 // Gestor eventos no creado
-3005 // Al crear el dispositivo de eventos
-3006 // Dispositivo TTS ya abierto
-3007 // Dispositivo TTS no abierto
-3008 // Al abrir el dispositivo TTS
-3009 // Al obtener el dispositivo de audio
-3010 // No existe obj. de audio
-3011 // No existe obj. comunicador de eventos
-3013 // Libreria no inicializada
-3014 // Libreria ya inicializada
-3015 // Conversion TTS en curso
-3016 // El objeto del conversor ya esta creado
-3017 // El objeto del conversor no esta creado
-3018 // El objeto del conversor ya se habia liberado
-3019 // No se ha pasado ningun parametro
-3020 // No se ha indicado el fichero de salida de muestras
-3021 // Los parametros son incorrectos y se establecen los que hay por defecto
-3022 // Formato del parametro texto incorrecto
-3023 // Error en la conexion al socket de respuesta
-3024 // Error al cerrar el socket de respuesta
-3025 // Error al enviar los datos por el socket de respuesta
-3026 // El objeto del reconocedor ya esta creado
-3027 // El objeto del reconocedor ya se habia liberado
-3028 // El objeto del reconocedor no esta creado
-3029 // Reconocimiento ASR en curso
-3030 // Dispositivo ASR ya abierto
-3031 // Al abrir el dispositivo ASR
-3032 // Dispositivo ASR no abierto
-3033 // Error en los objetos de audio
-3034 // Error fatal en un dispositivo
-3035 // Error al establecer los parámetros
-4000 //KVERIFIER_NO_ERROR: Operation completed successfully. No errors or warnings detected.
-4001 //KVERIFIER_CANNOT_FREE_MEMORY (1): Resources cannot be freed. This should happen if a null pointer or memory size zero is passed.
-4002 //KVERIFIER_ENVVAR_NOT_FOUND (2): KIVOX_VERIFIER environment variable has not been detected. This sould not happen as the installer registers the variable.
-4003 //KVERIFIER_INITIALIZE_UNHANDLED_ERROR (3): An unexpected controlled error has occurred. Please contact KIVOX support.
-4004 //KVERIFIER_CANNOT_OPEN_CONFIG_FILE (4): kivox_verifier.ini file can not be opened. It uses to be a path error. If the installer defines the

environment variable and ‘data’ folder path is not changed this error should not happen.

-4005 //KVERIFIER_CONFIG_FILE_MISSING_TAG (5): There is a missing tag in the configuration file or it has been incorrectly quantified. There is a formatting error in fivox_verifier.ini

-4006 //KVERIFIER_ERROR_CONFIG_SNR_WARN_BELOW_SNR_ERROR (6): SNR warning threshold cannot be lower than error threshold.

-4007 //KVERIFIER_ERROR_CONFIG_NETSP_WARN_BELOW_NETSP_ERROR (7) Net speech warning threshold cannot be lower than error threshold.

-4008 //KVERIFIER_CANNOT_LOAD_UBM_FILE (8): UBM file cannot be loaded. If neither environment variable nor ‘data’ path have been modified, please contact KIVOX support.

-4009 //KVERIFIER_CANNOT_LOAD_NAP_MATRIX (9): NAP matrix cannot be loaded. If neither environment variable nor ‘data’ path have been modified, please contact KIVOX support.

-4010 //KVERIFIER_CANNOT_INITIALIZE_CORE_UBM_NAP (10): NAP matrix has been loaded but cannot be initialized. If neither environment variable nor ‘data’ path have been modified, please contact KIVOX support.

-4011 //KVERIFIER_INVALID_DATA_INPUT_PARAMETERS(11): Incorrect parameters have been passed to the API, for example, an audio input without data.

-4012 //KVERIFIER_AUDIO_NO_WAV_FORMAT (12): Audio input passed is not WAV formatted or is corrupt.

-4013 //KVERIFIER_AUDIO_WAV_NOT_PCM16 (13): Passed WAV is not PCM16

-4014 //KVERIFIER_AUDIO_WAV_ NOT_MONO (14): Passed audio is not mono.

-4015 //KVERIFIER_AUDIO_INVALID_SAMPLE_RATE (15): WAV sampling is different from 8KHz

-4016 //KVERIFIER_ERROR_SNR_OR_NETSPEECH_CALCULATION(16): An error has occurred while calculating an audio input SNR or net speech. Please contact KIVOX support.

-4017 //KVERIFIER_SNR_BELOW_ERROR_THRESHOLD(17): Audio input SNR is below error threshold established in kivox_verifier.ini

-4018 //KVERIFIER_NET_SPEECH_BELOW_ERROR_THRESHOLD(18): Audio input net speech is below error threshold established in kivox_verifier.ini

-4019 //KVERIFIER_SNR_BELOW_WARNING_THRESHOLD (19): Audio input SNR is below warning threshold established in kivox_verifier.ini

-4020 //KVERIFIER_NET_SPEECH_BELOW_WARNING_THRESHOLD(20): Audio input net speech is below warning threshold established in kivox_verifier.ini

-4021 //KVERIFIER_ERROR_CREATING_FPG_STREAM (21): Error creating FPG internally. Please contact KIVOX support.

-4022 //KVERIFIER_ERROR_CREATING_FSI_STREAM(22): Error creating FPG internally. Please contact KIVOX support.

-4023 //KVERIFIER_ERROR_CREATING_MIV_MODEL(23): Error creating audio model (.MIV) internally. Please contact KIVOX support.
-4024 //KVERIFIER_ERROR_APPLYING_CHANNEL_FACTORS (24): Error applying channel factors. Please contact KIVOX support.
-4025 //KVERIFIER_ERROR_COHORT_DIRECTORY_NOT_FOUND (25): Reference population models folder cannot be found. If neither environment variable nor ‘data/cohort’ path have been modified, please contact KIVOX support.
-4026 //KVERIFIER_ERROR_CREATING_OBSERVATIONS_MATRIX (26): Error creating observations matrix. Internal error. Please contact KIVOX support.
-4027 //KVERIFIER_ERROR_OPENING_FPG_FROM_COHORT (27): Error opening reference population FPG file. During enrollment, this error may indicate problems opening the file. Howerver, during identification process this error indicates that a reference population model that was part of the optimal population of a system user has been deleted.
-4028 //KVERIFIER_ERROR_OPENING_FSI_FROM_COHORT (28): Error opening reference population FSI file. During enrollment, this error may indicate problems opening the file. Howerver, during identification process this error indicates that a reference population model that was part of the optimal population of a system user has been deleted.
-4029 //KVERIFIER_ERROR_OPENING_MIV_FROM_COHORT (29): Error opening reference population MIV file. During enrollment, this error may indicate problems opening the file. Howerver, during identification process this error indicates that a reference population model that was part of the optimal population of a system user has been deleted.
-4030 //KVERIFIER_ERROR_CALCULATING_SINGLE_DISTANCE (30): Error calculating distance between models. Please contact KIVOX support.
-4031 //KVERIFIER_ERROR_NOT_ENOUGH_POPULATION_MODELS(31): There are not enough models in reference population folder (data/cohort).
-4032 //KVERIFIER_ERROR_CREATING_VERIFIER_MODEL (32): Error creating Verifier KVV model. Please contact KIVOX support.
-4033 //KVERIFIER_MODEL_INVALID_OR_CORRUPTED (33): KVV model supplied is not valid or is corrupted. Please contact KIVOX support.
-4034 //KVERIFIER_ERROR_SIMPLE_ SCORE (34): Error calculating basic ‘score’. Please call KIVOX support.
-4035 //KVERIFIER_ERROR_CONVERTING_AUDIO_TO_PCM16 (35): Error converting audio to subtype PCM-16.
-4036 //KVERIFIER_AUDIO_NO_ALAW_NO_MULAW (36): Error converting audio. The entry subtype is not supported by the converter.
-4037 //KVERIFIER_ERROR_VERIFIER_LICENSE (37): License files not present or license files belonging to a different computer.
-4038 //KVERIFIER_INITIALIZED_ALREADY (38): Trying to initialize and already initialized instance of KIVOX Verifier.
-4039 //KVERIFIER_NOT_INITIALIZED (39): Error trying to use Verifier without initializing it.
-4040 //KVERIFIER_ERROR_LICENSE_EXPIRED (40): License expired.
-4041 //KVERIFIER_MAX_CONCURRENCY_IS_REACHED(41):Maximum number of licensed simultaneous Verifier’s instances loaded.

-4042 //KVERIFIER_WARNING_LOG_INITIALISATION_FAILED(42):The format of the logging configuration file is not correct and the initialization failed.
-4043 //KVERIFIER_WRONG_POPULATION (43): The current population does not match with the one this model was trained.
-4044 //KVERIFIER_WARNING_LICENSE_ABOUT_TO_EXPIRE (44): License is going to expire in less than 15 days.

## A.2. Errors Sent From ASR

When any Word is recognized, is returned and XML with follow structure:

```
<request><sda_event>END_ASR</sda_event>
   <candidate num="0">
      5
   </candidate>
</request>
```

El número devuelto es el que indica la causa.

1 - Evento. Terminado ASR por vencimiento de ventana de reconocimiento
2 - Evento. Terminado ASR por anticipación de usuario
4 - Evento. Terminado ASR por pulso de voz demasiado largo
5 - Evento. Terminado ASR porque el usuario no contesta

# APPENDIX B. XML FILE

The execution of available server functions and responses are done in an XML structure, which facilitates the compatibility with any software.

Functions Start Server, Stop Server, Start TTS, Stop TTS, Start ASR, Stop ASR, Train Kivox speaker model and Verify speaker are sent with empty XML field, while in the functions Configure TTS, Start TTS, Configure ASR and Play Sound are sent within an XML, also the responses from the server when completed an operation is also be performed via XML when the operation was successful, if some error occurs in the execution of function it is returned an error code.

The basic structure of XML is firstly a tag, a markup construct that begins with "<" and ends with ">". Tags come in two flavors: start-tags, for example <section>, end-tags, for example </ section>, which indicates the function to perform, and within this tag will be the necessary data for the function. Each data will be in a tag *param* containing the name and parameter value. There exist an exception to this structure, it is when it is sent the text to speech convert, which simply is between a tag indicating that it is a text.

<'function name'><param name="parameter name" value="parameter value"><param na...>....</ 'function name'>

Responses are between request tags, these indicate through server event tags the sort of event created by the response and following the return data properly tagged. XML sent to server to execute the functions and their responses are formed according the following structures:

**Configure TTS:**

XML structure:

<paramtts><param name="parameter name" value="parameter value"><param na...>....</paramtts>

Dealt parameters:

-To specify speaker *name="gender" value="male/female"*

-Language (it is always Spanish in the first version) *name="language" value="spanish"*

-Output audio frequency *name="frequency" value="8/16"*

-Recognized text output *name="output" value="file/audio"*, if file output *name="filename" value="name"*

**Start TTS:**

XML structure:

<text>"hello"</text>

Returns <request><sda_event>END_TTS</sda_event></request> if the execution is correct.

**Configure ASR:**

XML structure:

<paramasr><param name="parameter name" value="parameter value"><param na...>....</paramasr>

Dealt parameters:

-File that contains information about grammar and needed models to recognizer (also includes information of other parameters) *name="grammar" value="DIASSEMANA.VOC"*

-Windows size (maximum sentence length), depends of the grammar: Long sentences --> 25-30", single words --> 8"

*name="time" value="25000"*

-Length of initial silence *name="silence" value="2000"*

- Length of embedded silence (words separation within a sentence): Short sentences --> 1", Long sentences --> 2-3"

*name="embed" value="1000"*

-Pulse length (voice pieces divided by embedded silences) *name="pulse" value="5000"*

Bear in mind that all values are in milliseconds.

**ASR Response**

The answer of automatic speech recognition do not contains just the text recognized, but contain the first five candidates that recognizer calculates likelier to be valid and also send with each candidate the confidence, percentage obtained of the candidate is the correct.

```
 <request><sda_event>END_ASR</sda_event><candidate num="1">
         <result grammar="NOMBRE GRAM.">
                     <rule id="gram" confidence="XX">
                               <rule id="ReglaNivel1" confidence="XX">
                               …
                                          <rule id="ReglaNivelN" confidence="XX">
                                                     [LO QUE SE HA RECONOCIDO]
                                          </rule>
                               …
                               </rule>
                     </rule>
          </candidate>
          <candidate num="2">
                     <result grammar="NOMBRE GRAM.">
 …
          </candidate>
          …
</result></request>
```

**Play Sound**

XML structure:

<playbeep><param name="repeat" value="3"><param name="gap" value="500"></playbeep>

As it can be observed, in this function there are set just two parameters, the number of sound repetitions and the separation between them, gap, the value of the gap is in milliseconds.

It returns <request><sda_event>END_PLAYBEEP</sda_event></request> if the execution is correct.

**Training Response and KIVOX Verification**

It is sent an identifier to realize the training, which is related with the models of Agnitio library. After, when identification is required, it would be verified audio with corresponding model of this identification.

<kivoxtraining><param name="ID" value="identificador del locutor"></kivoxtraining>

<kivoxverify><param name=”ID” value=”identificador del locutor”></kivoxverify>

It returns <request><sda_event>END_KIVOX_TRAINING</sda_event></request if the execution is correct in training or

<request><sda_event>END_KIVOX_VERIFY</sda_event><result>”result kivox verify”</result></request>, result has to be USER_VALID or USER_NOT_VALID