RESEARCH AND DEVELOPMENT CENTRE

Real Time Speech Translator

Xavier Garcia Cabrera

25/06/2008

Author	Xavier Garcia Cabrera
Project director	Jan Rudinsky
Date	25/06/2008

Summary

1 Overview

This document is written to report on my work on the Real Time Voice Translator Project, the project I carried out as my final thesis project during the academic year 2007-2008. During this period I have been working in the Research and Development Center (RDC) for Mobile Applications, a department of the Czech Technical University (CTU) in Prague. In the RDC I was a member of the Automatic Call Center Project (ACC Project) team, and within it, I was assigned to carry out the Real Time Voice Translator Project.

The Automatic Call Center Project (ACC Project), now renamed to Voice2Web Project, is a project carried out by the Research and Development Center. The RDC is a department inside the Electro Technical Faculty of the CTU that carries out Research and Development projects regarding the Information Technologies (IT). Some of its partners are IBM, Vodafone and Ericson, who the RDC is doing projects for.

The ACC Project began on 2007 and its aim is to develop Voice Applications, within the IBM and RDC agreement, using IBM Voice Technologies and whatever open standards or open source software. IBM is an ACC Project partner and provides financing for it. It also provides hardware and software licenses to the ACC Project and gives us support. The members of the ACC Project are developing several Voice Applications at the same time, all them following the ACC Project purposes.

Although this document is focused on the Real Time Voice Translator Project, it will also explain in the introduction some aspects of the ACC Project. This is because the Real Time Voice Translator Project has a lot of points in common with it and it is worth, to understand it well, understand some points of the ACC Project as well.

2 Introduction: ACC Project

2.1 Platform description

To run Voice Applications and achieve the goals of our team, it is needed to have a platform to build on it the all the projects we are going to carry on. This platform is composed of hardware, software and network architecture that work together to achieve its proposal.

The following sections describe the different parts of our platform, and explain how they are configured to work together.

2.1.1 Software

To get all the features we need from our platform we use some software installed on our servers that provides the services we require. We have been adding the different software applications during the developing of the project as we have needed them, not all at the same time at the beginning of it. Therefore, the number of the software applications used may be increased in the future if we require some extra functionality the software we have now installed is not able to provide.

Each time we have found the necessity of adding some functionality, we have found out the applications that are able to provide us our requirements and we have chosen which fits better to our present and future necessities. One of the most important criteria we have used to choose the software applications to use is that it should be open source. The only exception we have done, is with the IBM WebSphere software, due IBM is supporting our project and provided us this for free.

In the Table 2-1 you can see the current software installed in our servers and the features it provides to our platform and the applications in which they are used.

Software	Server	Features and Uses
Fedora Linux	Adela, Was, Bolek	This is the OS used on all our servers. As all the Linux distributions, it provides all the basic functionalities for a server like: user accounts, ssh server, ftp server, network protocols…
Voice Enabler (VE)	Adela	It is a part of the IBM WebSphere Server framework. It provides the basic capabilities

		to enable Voice Applications. It understands the SIP and CCXML protocols and manages and processes the VXML file. It receives the SIP calls forwarded by Asterisk server, assigns them a VoiceXML Application, and executes and interprets the VXML files of that application.
Voice Server (VS)	Was	This is a part of the IBM WebSphere Server framework. It is installed in the Was server and implements the both Voice Engines: ASR (Automatic Speech Recognition) and TTS (Text To Speech). It receives the request from Voice Enabler through MRCP Protocol and returns the result of processing them by the Voice Engines. The engines require more amount of CPU power than other applications and this is the reason why it is installed in a separate machine.
IBM HTTP Server	Adela	It is supplied inside the IBM WebSphere Server framework. It is an HTTP server, and provides de basic features of this kind of servers. It is used to host web pages and the VXML files of static VoiceXML applications.
Apache-Tomcat	Adela	It is an HTTP Server enabled to run Java Servlets. It is used to run the Servlets we need to generate Dynamic VoiceXML Applications. It also host some web pages and some configuration files used by Servlets.
MySQL	Adela	MySQL is a very popular and open source database. It provides the same functionalities of the most powerful proprietary databases. We use it as a database in the development of VXML IDE Project.
Java Framework	Adela, Was	It provides de JVM (Java Virtual Machine) and extra resources needed to execute Java applications. It is required to be installed in all the machines that run IBM WebSphere framework. We also require it in Adela to run the Java Servlets.
Asterisk	Bolek	It is the most worldwide used open source SIP server. We use it to manage the incoming SIP calls and redirect them to our Voice Applications. It also make possible to receive SIP calls behind a NAT/Firewall.

Table 2-1. Software used in ACC Project Platform.

2.1.2 Hardware

We have three physical machines. These machines are the servers that run the software that provides the functionalities we need. Two of these servers, those that run IBM software, were provided by IBM. The other, where is the Asterisk server installed, belongs to RDC department. Our entire infrastructure is behind the Firewall/ NAT of the CVUT University, but the server that runs the Firewall/NAT server belongs to the university and is administrated by the University's administrator, so it will be considered out of our platform.

Each of the machines is identified by a name (Adela, Was and Bolek) and run different software applications. Following our principle of using always open source software in the whole project, we use Fedora Linux operation system on all these servers. In Table 2-2 you can see the software installed on each machine.

Name	Installed Software
Adela	Voice Enabler (IBM WebSphere) IBM HTTP Server (IBM WebSphere) Apache-Tomcat HTTP Server MySQL Database
Was	Voice Engine (IBM WebSphere)
Bolek	Asterisk SIP Server

Table 2-2. Server names and installed software.

In the following table you can see the hardware technical features of each of our servers:

	Adela	Bolek	Was
Model	DELL - PC ABACUS ARCH 8700N	DELL - PC ABACUS ARCH 8700N	SUPERMICRO SuperServer
CPU	Intel Pentium 4 3.6 GHz	Intel Pentium 4 3.6 GHz	Intel Xeon 5130 @ 2.00GHz
RAM	2 GB	2 GB	2 GB
HD	230 GB	230 GB	230 GB
Graphics card	GeForce 550	GeForce 550	-

Table 2-3. Servers' hardware features.

2.1.3 Architecture

Our platform is constituted for several servers which are interconnected within a network. We are not using a dedicated network only for our project. We are using the university network, shared with all other university servers. Our servers are Bolek, Was and Adela. They connect and receive connections from internet through the university Firewall/NAT server. That is the reason why we need to use asterisk to be able to manage SIP connections behind a NAT server. In Figure 2-1 you can see the ACC Project network architecture.

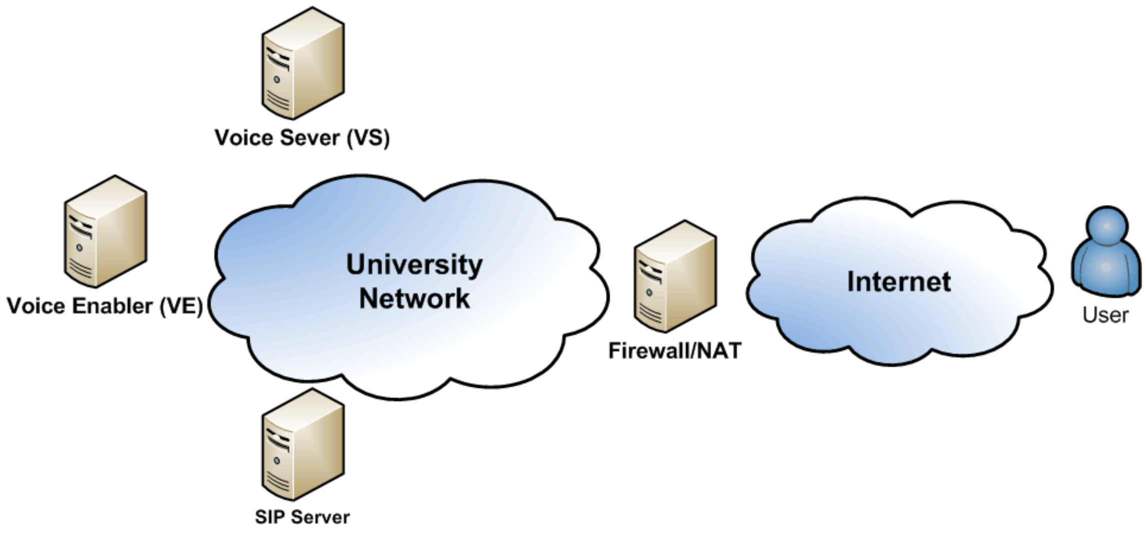


Figure 2-1. ACC Project network architecture.

When a user wants to use our services, he has to establish a connection with our SIP server (Asterisk server installed on Bolek) which is behind the university firewall. Once this connection is established, our SIP server redirects the call to the Voice Enabler (VE) where there are the VoiceXML applications. Voice Enabler takes the control of the call and executes the Voice Application that user is requesting. To be able to interact with the user, Voice Applications need to use a Voice Server (VS). When it is needed, VE establish an MRCP connection with the VS (Bolek server) to make the queries and receive the answers from the Voice Engine. In Figure 2-2 you can see the call diagram and the protocols used to establish the connections between servers.

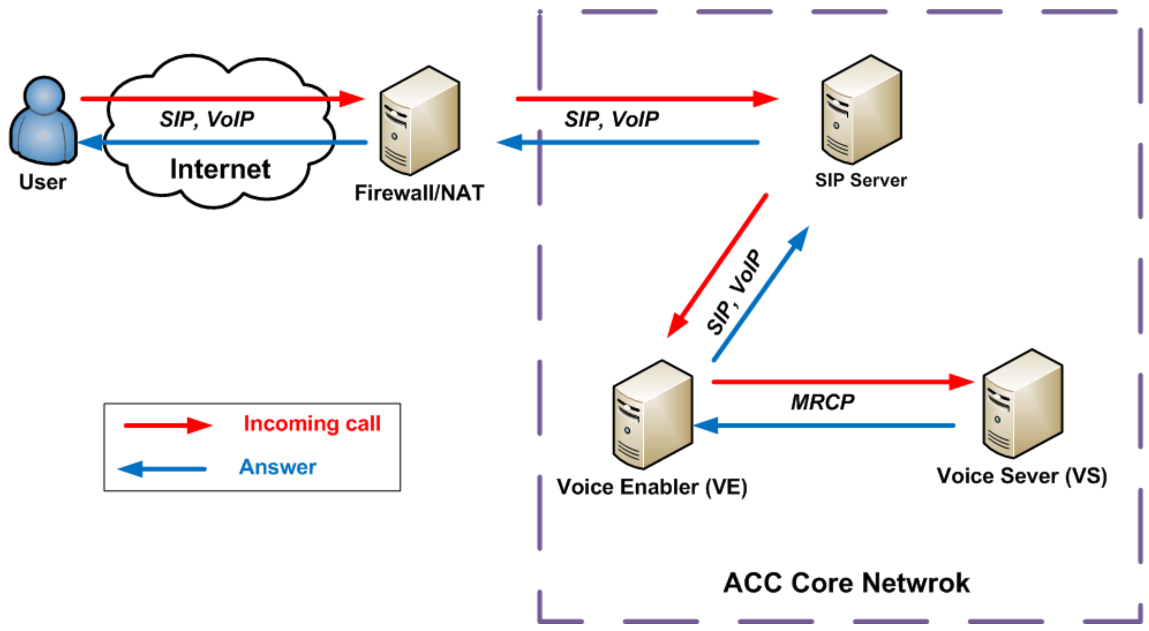


Figure 2-2. Call diagram.

2.2 ACC Team Projects

ACC Project is a big Project which purpose is developing all kind of Voice Applications and not only those focused on call center applications. Therefore, ACC Project is made up of other smaller projects carried out for its members that match the main project aims.

At the beginning, ACC Project team made a brainstorming and its members proposed several Voice Projects to realize. After that, the proposed projects were sorted by priority and the team began to work on those which were more important according on its criteria. After that, some other projects were coming up during the development of the Project and they were added to the list.

In this section, I will explain some of these projects. Some are currently on development and others are planned to be done in the future by the ACC Team.

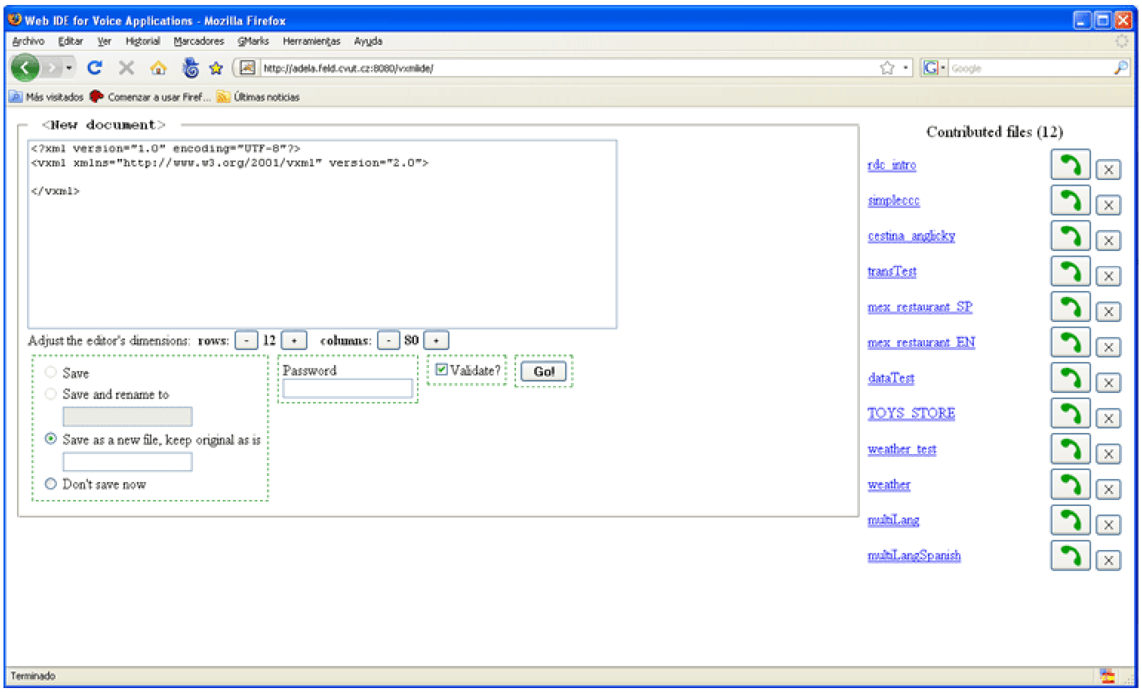
2.2.1 Real Time Voice Translator

This project was set the first in the priority list and it was the first Voice Application to be started. We decided it so because many of the steps done to realize this project will be required in any other Voice Project we purpose to carry out in the future.

The main purpose of this project is to develop a Voice Application which was able to do a real time translation of a phone conversation between two callers who speak different languages. This is the project that was assigned to me, Xavi Garcia, and which I have been carrying out during all these months. This project will be explained deeply in section 4.

2.2.2 VXML IDE

Despite it is not a Voice Application properly speaking, it is within the purposes of the ACC Project. It was in the second position in the ACC Team priority list, so it means this project is very important for us. The final aim of this project is developing a Web based IDE (Integrated Development Environment) for VoiceXML Applications. Its purpose is making available to a wide range of programmers VoiceXML technology and supply them an easy way to develop their own Voice Applications. The most important requirement is that it has to be a Web Application; therefore it will be available for everybody without any kind of local installation, what the user will only need is a web browser. This project is currently under development and is carried out by Tomas Mikula.


Web IDE for Voice Applications - Mozilla Firefox
Archivo Editar Ver Historial Marcadores GMarks Herramientas Ayuda
http://adela.feld.cvut.cz:8080/vxmlide/
Más visitados Comenzar a usar Firef... Últimas noticias
Terminado

<New document>

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0">

</vxml>
```

Adjust the editor's dimensions: rows: - 12 + columns: - 80 +

- ○ Save
- ○ Save and rename to
- ● Save as a new file, keep original as is
- ○ Don't save now

Password

☑ Validate? Go!

Contributed files (12)

- rdc_intro
- simpleccc
- cestina_anglicky
- transTest
- mex_restaurant_SP
- mex_restaurant_EN
- dataTest
- TOYS_STORE
- weather_test
- weather
- multiLang
- multiLangSpanish

Figure 2-3. VXML IDE interface.

2.2.3 ACC security

This is another non Voice Application, but it is extremely important for our Project. The aim of this project is provide security to all our project applications and protect our infrastructure against attacks. This task is being accomplished by Michal Vanek and he is focusing on providing the firewall rules and doing the proper software configuration (Asterisk server …) to avoid DoS attacks.

2.2.4 Others

ACC Team has planned to carry out some other projects in the future. They have not started yet because either we have not accomplished some their prerequisites or we have not enough team members to do all the projects at the same time. Some of these future projects are: Phone Wikipedia Project, and third party applications.

Phone Wikipedia Project pretends to make available all the contents of Wikipedia through a mobile phone call. That is, everybody who have a mobile phone would be able to read (in this case would be able to listen) any Wikipedia article available in internet.

There are other Projects that came up during the development of other tasks. This is the case of Weather Application, which was developed to test the VoiceXML DATA tag. This application provides the current weather of any European capital.

3 Real Time Voice Translator

The Real Time Voice Translator is one of the Projects carried out for the ACC Project Team. This document will focus on it more deeply than other ACC projects because it is the project that was assigned to be carried out by me. In the following sections I will describe the different parts of it and how it has been fulfilled.

3.1 Objectives

The purpose of this project is to develop a Voice Application that was able to do a real time translation from a spoken sentence in one language to a spoken sentence in other language. The basic idea is to enable that two people, who are talking through a voice call in two different countries, were able to understand each other speaking different languages due to an automatic real time translation done by a machine.

3.2 Requirements

In this project, like in every project; before carrying out it, it is needed to establish some requirements that should be accomplished. Once all these requirements had been achieved, we also will be able to say that the aims of the Real Time Voice Translator have been achieved. These requirements are the followings:

- The application must be autonomous. It should work without the interaction of any operator.
- It is needed to generate dynamic VXML files dynamically. It is not possible to store a list of possible results. All the answers will be generated after the user said his sentence.
- The application should be able to understand everything the user said. User should be able to talk about whatever he wants and the application would be able to translate what he has spoken.
- It should be able to translate simple sentences to/from at least two languages.
- It is needed to use a Translator Engine which was able to do the translation at real time.
- The delay between the moment the caller ends his sentence and the moment the translated sentence is delivered to the callee should be as short as possible to make possible a smooth conversation.

3.3 Market Analysis

Before beginning this project, we did an internet research to find out other possible services, either commercial or free services, which were currently offering the same features we attempt to offer with the Real Time Voice Application.

This Market Analysis shows that, although there are some attempts to give a real time translation, they do not approach the problem in the same way we want to do in our application. Some of them try to solve the problem using a mobile machine where the user talks and it replies with the translated sentence. Others offer translation services for some written Instant Messaging (IM) services. And Skype has introduced a new service to provide Instant Translation in phone calls done through a human interpreter.

In the following sections I will explain the features of each of the most important services that try to approach the translation problem in communications. These services are: Skype Personal Interpreter Service, Google Talk Bots, NEC Automatic Translator for mobiles.

3.3.1 Skype Personal Interpreter Service

Skype is now offering real-time voice translation services for Skype calls. Language Line Personal Interpreter offers SkypeOut customers near-instantaneous access to professional voice translation. This is an access to live interpreters, not machine translation. The cost is $2.99 per minute, which is relatively high, but with this type of service you are paying for the convenience first and foremost. The use of this service is on the fly, with no scheduling. You can get an interpreter on average in 45 seconds after an initial request.

Features:

Price	$2.99/min
Scope	Phone calls using Skype
Key points	– Live interpreters, not machine translation – 150 supported languages – No scheduling
More info	www.skype.com

Table 3-1. Skype Personal Interpreter Service features

3.3.2 Google talk bots for real-time translations in IM (Instant Messaging)

Recently, on December 2007, Google announced the availability of real-time translations for their IM (Instant Messaging) service Google Talk. The translations are made by bots that translate the text messages in the Google Talk conversations. For each language there are two bots, one for each translation direction. For example, for Spanish to English translation there are the boots en2es@bot.talk.google.com and es2en@bot.talk.google.com. This service supports 23 translation directions (bots).

Features:

Price	Free
Scope	Instant Messaging conversations using Google Talk
Key points	– Text translation only – 23 translation directions supported
More info	http://googletalk.blogspot.com/2007/12/merry-christmas-god-jul-and.html

Table 3-2. Google Talk Bots features.

3.3.3 NEC Automatic Translator for mobile phones

Japanese electronics company NEC has developed the first automatic translation software tailored for mobile phones. It only translates from speech Japanese to English, but this is the first time the translation technology is used in mobile phones without any external help. The software has a dictionary of 50,000 words and it is still under development.

Features:

Price	unknown
Scope	Mobile phone users
Key points	– Japanese to English translation only (not English to Japanese) – Under development
More info	http://www.akihabaranews.com/en/news_details.php?id=15185

Table 3-3. NEC Automatic Translator features.

3.4 Choosing a Translation Engine

In order to carry out the Real Time Voice Translator Project, it is mandatory to have a real time translator which can provide us the translated sentences. As it is a very hard work to implement a translator engine, we decided to use one of the free translators available in Internet.

Hence, first of all we did a research to find out all the free translator engines available. After this research we had to decide which one we choose for our purposes. Like it is known for everybody, all the translation engines currently available have not a good accuracy. Therefore we decide to choose that one which had the best translation accuracy. To know which one of all the preselected translation services has the best accuracy, I did a test that give a punctuation to each one and finally I compare the results.

3.4.1 Web Translation services

After a deep research, we find out the most important free translation services currently available in internet. We have preselected all them to do our test and choose the best translation engine to use in our application. In Table 4-4 you can see a list of all these services and their URLs.

Service	Website
Altavista Babel Fish	http://babelfish.altavista.com/
Yahoo! Babelfish	http://babelfish.yahoo.com/
Google Translator	http://translate.google.com/translate_t?hl=en
InterTran	http://www.tranexp.com:2000/InterTran
FreeTranslation	http://www.freetranslation.com/
WorldLingo	http://www.worldlingo.com/en/products_services/worldlingo_translator.html
Dictionary.Com	http://translator.dictionary.com/text.html
PROMT	http://www.e-promt.com/
TraduceGratis.com *(Google engine)*	http://www.traducegratis.com/
Im translator *(PROMT engine)*	http://translation.paralink.com/

Table 3-4. Web translation services list.

3.4.2 Translation Web Services Accuracy benchmark

We decided to do an accuracy benchmark to each one of the preselected web translation services to be able to compare each other in a better way. This benchmark consist in translate several basic sentences from Spanish to English. I have selected some common sentences that are used in a basic conversation when someone is trying to contract some service. I specifically have selected the basic dialog for booking a taxi and the basic dialog for ordering a pizza. I have to notice that the accuracy mark is an absolutely subjective mark and it is only based on my personal opinion.

3.4.2.1 Spanish sentences

Here are listed the Spanish sentences of the dialogs which our benchmark is made of:

1. Buenos días le atiende Juan González, ¿en qué puedo ayudarle?
2. Desearía pedir un taxi para las 22 horas en la dirección: calle San Ramón número 2.
3. De acuerdo, su reserva ha sido realizada.
4. Desearía pedir una pizza tropical tamaño familiar.
5. ¿Desea añadir algún ingrediente más?
6. Sí. Extra de queso, jamón y peperoni.
7. Muy bien, ¿me puede decir su número de teléfono y su dirección?
8. Mi número de teléfono es el 777.777.777 y mi dirección es: calle Frank Kafka número 132.
9. Perfecto, en 30 minutos recibirá su pedido. El precio final es de 180 coronas.

In the following tables there are the result of the benchmark we passed on each translator engine and the mark they obtained on the translation of each sentence. At the end you can find a summary table with the global mark of all the engines.

Altavista Babelfish (Babelfish engine)		
	Translation result	Accuracy
1	Good morning it takes care of Juan to him González, in what I can help him?	45
2	It would wish to request a taxi for the 22 hours in the direction: street San Ramon number 2	90
3	In agreement, its reserve has been made.	80
4	Pizza would wish to request one tropical familiar size.	50
5	It wishes to add some ingredient more?	95

6	Yes. Cheese extra, jamón and peperoni.	75
7	Very well, it can say to its telephone number and its direction to me?	70
8	My telephone number is the 777.777.777 and my direction is: street Frank Kafka number 132.	85
9	Perfect, in 30 minutes it will receive its order. The final price is of 180 crowns.	85

Table 3-5. Altavista Babelfish Spanish-English benchmark results.

Yahoo! Babelfish (Babelfish engine)		
	Translation result	Accuracy
1	Good morning it takes care of Juan to him González, in what I can help him?	45
2	It would wish to request a taxi for the 22 hours in the direction: street San Ramon number 2.	90
3	In agreement, its reserve has been made.	80
4	Pizza would wish to request one tropical familiar size.	50
5	It wishes to add some ingredient more?	95
6	Yes. Cheese extra, jamón and peperoni.	75
7	Very well, it can say to its telephone number and its direction to me?	70
8	My telephone number is the 777.777.777 and my direction is: street Frank Kafka number 132.	85
9	Perfect, in 30 minutes it will receive its order. The final price is of 180 crowns.	85

Table 3-6. Yahoo! Babelfish Spanish-English benchmark results.

Google Translator		
	Translation result	Accuracy
1	Good morning he attends Juan Gonzalez, how can I help?	80
2	I would like to ask a taxi for 22 hours at the following address: Calle San Ramon number 2.	100
3	Okay, your reservation has been made.	100

4	I would like to ask a pizza tropical family size.	90
5	Want to add one more ingredient?	90
6	Yes. Extra cheese, ham and peperoni.	100
7	Well, what I can say your phone number and your address?	65
8	My telephone number is 777,777,777 and my address is: Frank Kafka street number 132.	100
9	Perfect, in 30 minutes you should receive your order. The final price was 180 kronor.	95

Table 3-7. Google Translator Spanish-English benchmark results.

InterTran		
	Translation result	Accuracy
1	Good morning him atiende John González , wherein it can lend a helping hand?	20
2	Hunger for order one taxicab in order to the 22 hours on the steerage : street San Ramón numeral 2.	35
3	In agreement , her reservation has been achieved.	65
4	Hunger for order one pizza tropical size home-like.	40
5	¿Desea append any ingredient further?	35
6	yes Extra of cheese , ham and peperoni.	85
7	Well done ¿me can you tell me her numeral of telephone and her steerage?	30
8	E numeral of telephone is the 777.777.777 and E steerage is : street Outspoken Kafka numeral 132.	20
9	Perfect , at 30 minutes she'll receive her request. The price eventual is of 180 halos.	60

Table 3-8. InterTran Spanish-English benchmark results.

FreeTranslation		
	Translation result	Accuracy
1	Good morning Juan attends him González, ¿in what can I help him?	45
2	It would desire to ask a taxi for the 22 hours in the direction: street San Ramón number	80

	2.	
3	In agreement, its reserve has been carried out.	80
4	It would desire to ask a pizza tropical family size.	80
5	It desires to add some ingredient more?	85
6	Yes. Extra of cheese, ham and peperoni.	100
7	Very well, ¿can tell me its phone number and its direction?	85
8	My phone number is the 777.777.777 and my direction is: street Frank Kafka number 132.	85
9	Perfect, in 30 minutes will receive its order. The final price is of 180 crowns.	85

Table 3-9. FreeTranslation Spanish-English benchmark results.

WorldLingo		
	Translation result	Accuracy
1	Good morning it takes care of Juan to him González, in what I can help him?	60
2	It would wish to request a taxi for the 22 hours in the direction: street San Ramon number 2.	80
3	In agreement, its reserve has been made.	90
4	Pizza would wish to request one tropical familiar size.	70
5	Desea to add some ingredient more?	65
6	Yes. Cheese extra, jamón and peperoni.	60
7	Very well, can say to its telephone number and its direction to me?	80
8	My telephone number is the 777.777.777 and my direction is: street Frank Kafka number 132.	90
9	Perfect, in 30 minutes it will receive its order. The final price is of 180 crowns.	90

Table 3-10. WorldLingo Spanish-English benchmark results.

Dictionary.Com		
	Translation result	Accuracy
1	Good morning it takes care of Juan to him González, in what I can help him?	60
2	It would wish to request a taxi for the 22 hours in the direction: street San Ramon number 2.	80
3	In agreement, its reserve has been realized.	75
4	So large relative would wish to request a tropical pizza.	50
5	It wishes to add some ingredient more?	80
6	Yes. Cheese extra, jamón and peperoni.	50
7	Very well, it can say to its telephone number and its direction to me?	75
8	My telephone number is the 777.777.777 and my direction is: street Frank Kafka number 132.	80
9	Perfect, in 30 minutes it will receive its order. The final price is of 180 crowns.	85

Table 3-11. Dictionary.com Spanish-English benchmark results.

PROMT		
	Translation result	Accuracy
1	Good morning Juan González attends to him: in what can I help him?	75
2	He would want to ask for a taxi for 22 hours in the direction: there is quiet San Ramón number 2.	50
3	In agreement, his reservation has been realized.	85
4	Familiar size would want to ask for a tropical pizza.	25
5	Does he want to add any ingredient more?	75
6	Yes. Extra of cheese, ham and pepperoni.	100
7	Very well: can he say to me his telephone number and his direction?	80
8	My telephone number is 777.777.777 and my direction is: there is quiet Frank Kafka number 132.	70
9	Perfect, in 30 minutes it will receive his order. The final price is 180 crowns.	85

Table 3-12. PROMT Spanish-English benchmark results.

Summary								
	Altavista	Yahoo!	Google	InterTran	FreeTranslation	WorldLingo	Dictionary.Com	PROMT
1	45	45	80	20	45	60	60	70
2	90	90	100	35	80	80	80	50
3	80	80	100	65	80	90	75	85
4	50	50	90	40	80	70	50	25
5	95	95	90	35	85	65	80	70
6	75	75	100	85	100	60	50	100
7	70	70	65	30	85	80	75	80
8	85	85	100	20	85	90	80	70
9	85	85	95	60	85	90	85	85
Avg	75	75	91	43	81	76	71	70

Table 3-13. Benchmarks summary.

In order to differentiate better the engines scores and have a clearer idea about them, it is required to have some statistics charts. The following graphs compares the average, maximum and minimum accuracy of each engine. It is highlighted with red color the best scored engine.

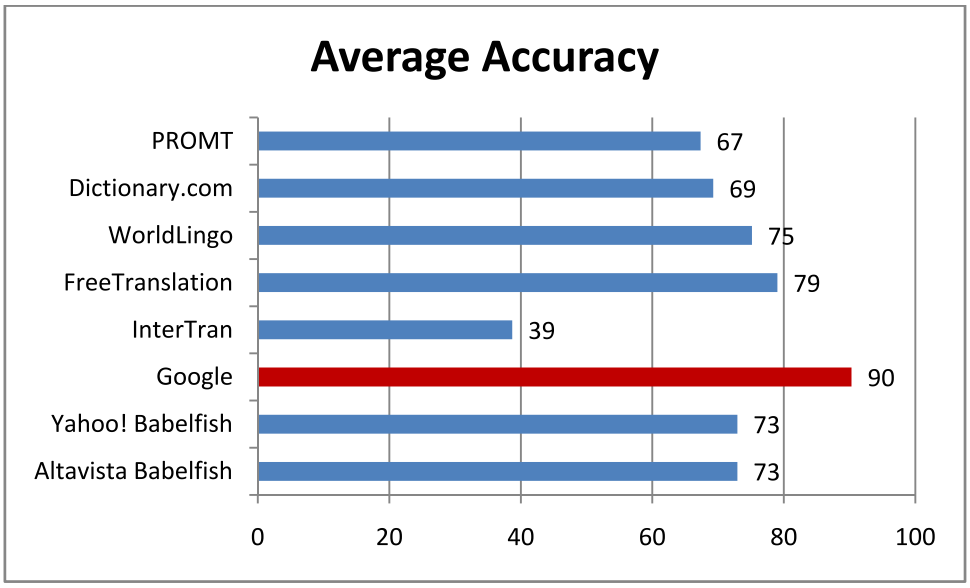


Figure 3-1. Average accuracy graph in Spanish-English Benchmark.

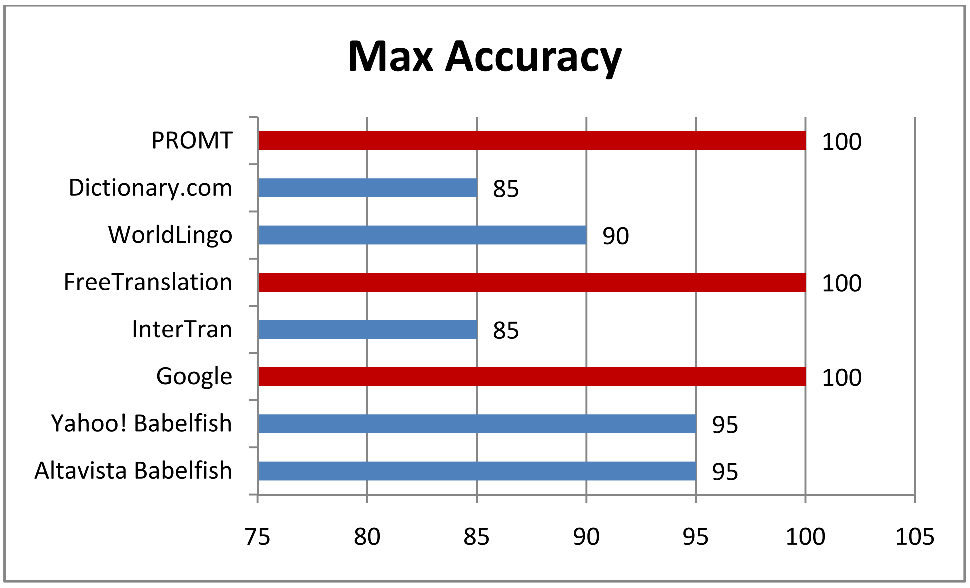


Figure 3-2. Maximum accuracy graph in Spanish-English Benchmark.

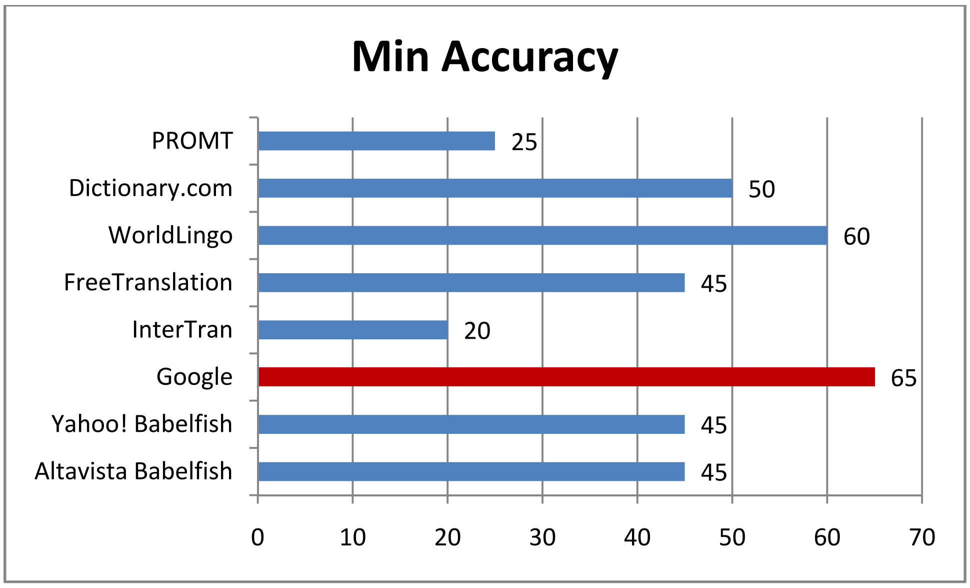


Figure 3-3. Minimum accuracy graph in Spanish-English Benchmark.

In Figure 3-4 you can see the range of marks obtained for each engine and where it is the average accuracy mark within it.

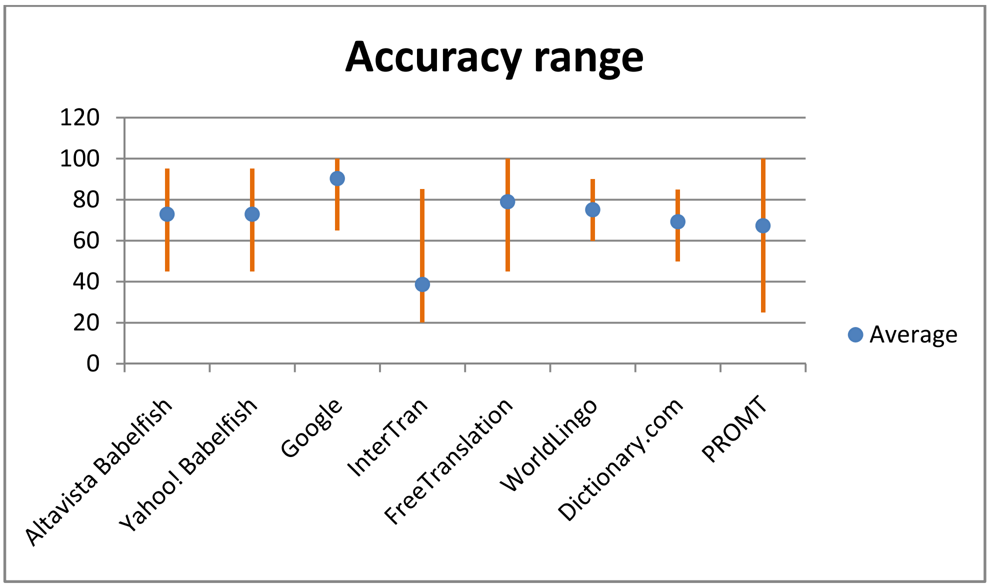


Figure 3-4. Accuracy range chart in Spanish-English Benchmark.

3.4.2.2 English sentences

Here are listed the English sentences of the dialogs which our benchmark is made of:

1. Good morning, it's Juan John Smith, what can I help you?
2. I would like to ask for a taxi at 22 hours at the address: Fifth Avenue number 2.
3. Ok, your reservation has been made.
4. I would like to order a Tropical pizza family size.
5. Do you want to add some other ingredient?
6. Yes. Extra cheese, ham and pepperoni.
7. All right, can you tell me your telephone number and your address?
8. My telephone number is 777.777.777 and my address is: Frank Kafka street number 132.
9. All right, you will receive your order in 30 minutes. The final price is 180 crowns.

Looking at the results of the Spanish-English benchmark, we can notice that there are several engines which have a very low accuracy. Therefore, we decided to do the English-Spanish benchmark only on the three engines which have the highest marks in the previous test.

In the following tables there are the benchmark results of each translator engine and the marks they obtained on the translation of each sentence.

Google Translator		
	Translation result	Accuracy
1	Buenos días, es Juan de John Smith, ¿qué puedo hacer para ayudarle?	80
2	Me gustaría pedir un taxi a las 22 horas en la dirección: Quinta Avenida número 2.	100
3	Ok, su reserva se ha realizado.	100
4	Me gustaría pedir un pizza Tropical tamaño de la familia.	95
5	¿Deseas añadir algún otro ingrediente?	100
6	Sí. Extra queso, jamón y pepperoni.	100
7	Está bien, puede usted me dice su número de teléfono y su dirección?	85
8	Mi número de teléfono es el 777.777.777 y mi dirección es: calle Frank Kafka número 132.	100
9	Muy bien, usted recibirá su pedido en 30 minutos. El precio final es de 180 coronas.	100

Table 3-14. Google Translator English-Spanish benchmark results.

Altavista & Yahoo! (BabelFish Engine)		
	Translation result	Accuracy
1	¿Buena mañana, él es Juan John Smith, cuál puede yo ayudarle?	40
2	Quisiera pedir un taxi en 22 horas en la dirección: Quinta Avenida número 2.	50
3	Se ha hecho la autorización, su reservación.	40
4	Quisiera pedir un tama o de la familia tropical de la pizza.	30
5	¿Usted quiere agregar un poco de otro ingrediente?	75
6	Sí. Queso, jamón y salchichones adicionales.	80
7	¿Todo a la derecha, puede usted decirme su número de teléfono y su dirección?	70
8	Mi número de teléfono es 777.777.777 y mi dirección es: Calle número 132 de Frank Kafka.	100
9	Todo a la derecha, usted recibirá su orden en 30 minutos. El precio final es 180 coronas.	80

Table 3-15. BabelFish engine English-Spanish benchmark results.

FreeTranslation		
	Translation result	Accuracy
1	¿Buenos días, es Juan John Smith, le qué puedo ayudar yo?	70
2	Querría pedir un taxi en 22 horas en la dirección: Quinto número de la Avenida 2.	60
3	Bueno, su reservación ha sido hecha.	80
4	Querría ordenar una pizza Tropical el tamaño familiar.	85
5	Quiere usted agregar algún otro ingrediente?	100
6	Sí. El queso extra, el jamón y la salchicha.	100
7	¿Bueno, puede decir me usted su número de teléfono y su dirección?	95
8	Mi número de teléfono es 777.777.777 y mi dirección es: El número franco de la calle de Kafka 132.	60
9	9. Bueno, usted recibirá su orden en 30 minutos. El precio final es 180 coronas.	80

Table 3-16. FreeTranslation English-Spanish benchmark results.

In the following graphs, you can see the statistics comparison between the different scored engines. In each graph, it is highlighted with red color the best scored engine. In the accuracy range chart, you can see the range of marks obtained for each engine and where it is the average accuracy mark within it.

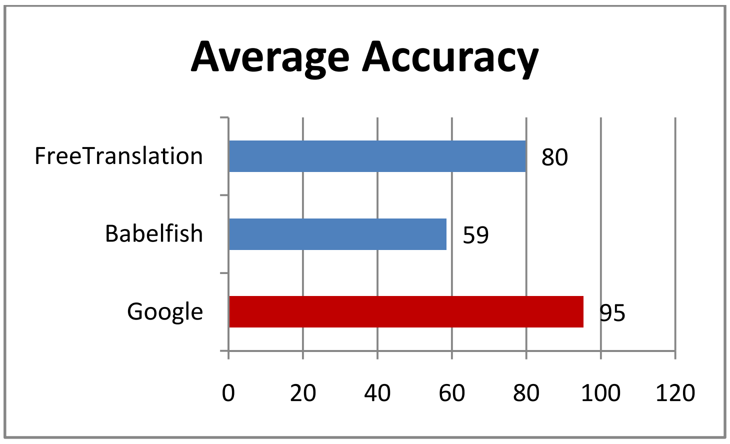


Figure 3-5. Average accuracy graph in English-Spanish Benchmark.

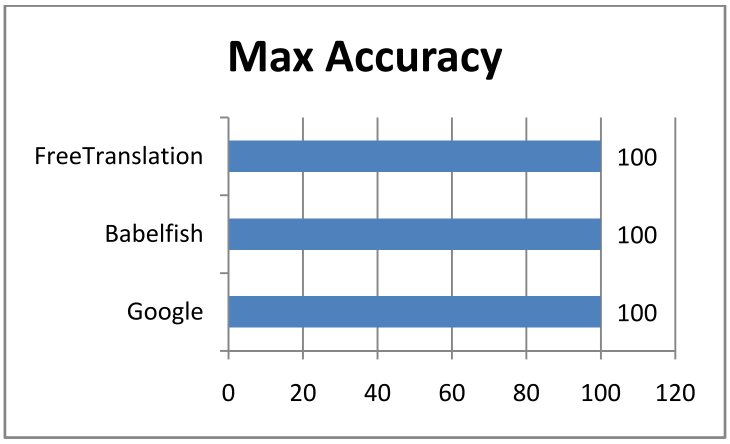


Figure 3-6. Maximum accuracy graph in English-Spanish Benchmark.

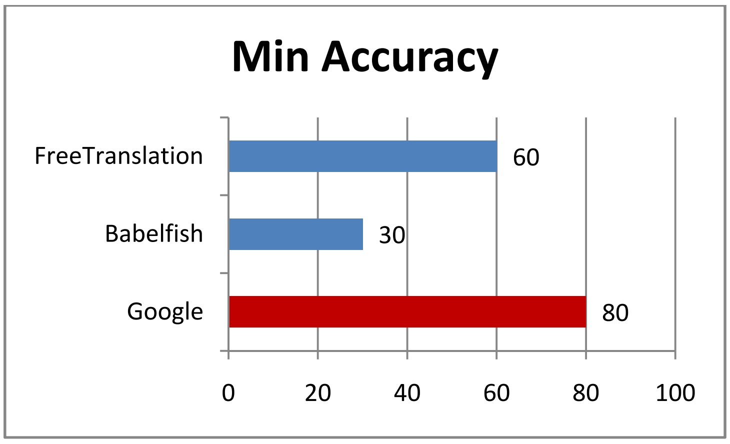


Figure 3-7. Minimum accuracy graph in English-Spanish Benchmark.

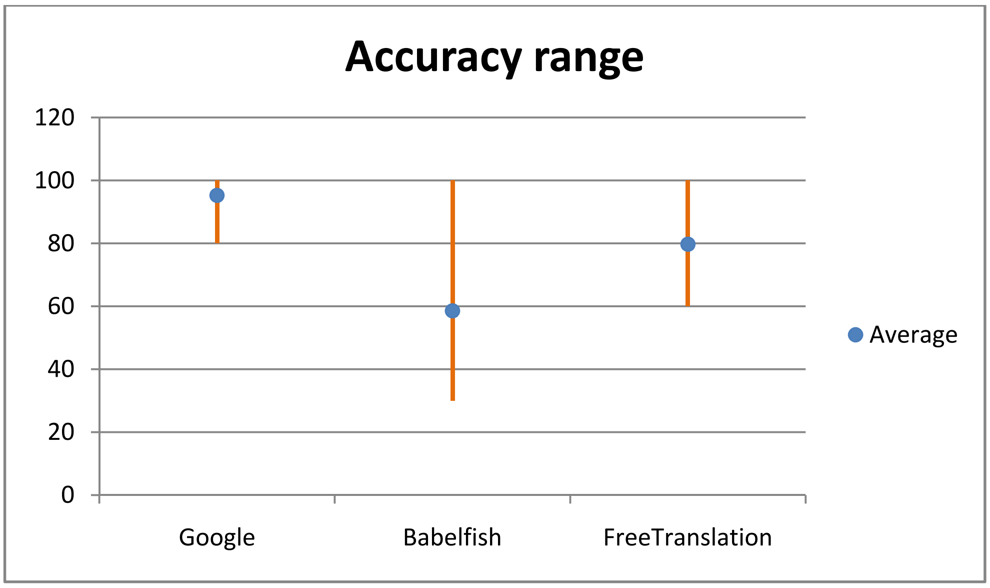


Figure 3-8. Accuracy range chart in English-Spanish Benchmark.

3.4.3 Conclusion

Looking at the both benchmark results, we can see that Google Translate stands out as the best web translator service. The translations it provides are quite understandable in all cases, whilst the results provided for other engines are not in many cases. For this reason, we chose Google Translate Engine for using in our project.

3.5 Software Tools

For the development of the Real Time Voice Translator it has been necessary to use several software tools. This tools cover a wide range of software applications, from HTTP servers to a programming IDEs. Each tool has been used to accomplish a task within the large number of parts our project is made of. In the following sections I am going to explain each of these software applications, their function inside the project architecture and how we have used and configured them.

3.5.1 IBM WebSphere Application Server

IBM WebSphere Application Server (WAS) is the flagship product within IBM's WebSphere brand. WAS is built using open standards such as Java EE, XML, and Web Services. It works with a number of Web servers including Apache HTTP Server, Netscape Enterprise Server, Microsoft Internet Information Services (IIS), IBM HTTP Server for i5/OS, IBM HTTP Server for z/OS, and IBM HTTP Server for AIX/Linux/Microsoft Windows/Solaris. WAS V6.1 is the foundation of the IBM WebSphere software platform. It delivers the secure, scalable, resilient application infrastructure that is needed for a Service Oriented Architecture (SOA).

We installed WebSphere Application Server V5.1 on a Linux system for using the Voice Enabler and Voice Engines servers it provides in our project. As it is explained in section *2.1-Software*, we have deployed WebSphere Application Server in two servers: adela and was.

3.5.2 Java EE

Java EE is the version of the Java Platform used to develop Enterprise Applications, focusing on a distributed server environment. The platform was known as Java 2 Platform, Enterprise Edition or J2EE until the name was changed to Java EE in version 5. The current version is called Java EE 5 whilst the previous version is called J2EE 1.4.

Java EE builds on the solid foundation of Java Platform, Standard Edition (Java SE) and is the industry standard for implementing enterprise-class service-oriented architecture (SOA) and next-generation web applications. The SDKs contain Sun GlassFish Enterprise Server, previously named Sun Java System Application Server, and provide support for Java EE 5 specifications.

The Java EE Platform differs from the Standard Edition (SE) of Java in that it adds libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server. It includes several API specifications, such as JDBC, RMI, e-mail, JMS, web services, XML, etc, and defines how to coordinate them. Java EE also features some specifications unique to Java EE for components. These include Enterprise JavaBeans, servlets, portlets (following the Java Portlet specification), JavaServer Pages and several web service technologies.

We chose Java EE platform to develop our application. Our dynamic VoiceXML applications use the power of Java Servlets to interact with other web services available in internet and manage data from different sources to create new voice services. We are currently using Java EE 5 version.

3.5.3 Apache Tomcat HTTP Server

Apache Tomcat is a Servlet container developed at the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run. Tomcat should not be confused with the Apache web server, which is a C implementation of a HTTP web server; these two HTTP web servers are not bundled together.

Apache Tomcat includes tools for configuration and management, but can also be configured by editing configuration files that are normally XML-formatted.

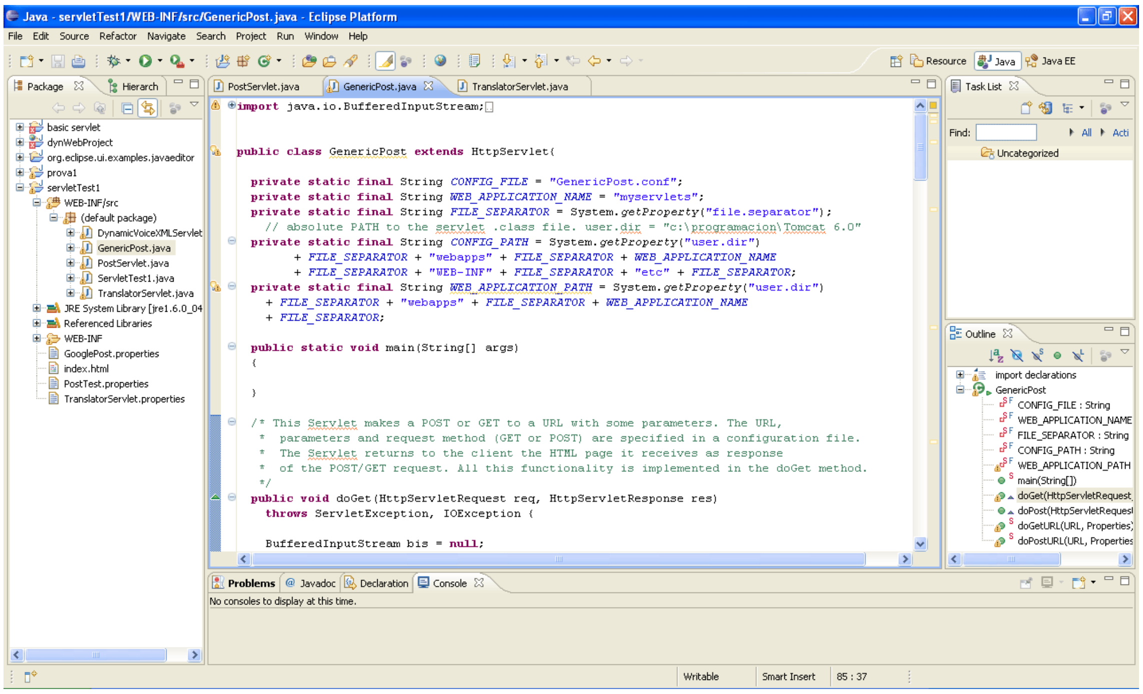
As I mentioned in previous sections, we use Java Servlets for implementing most of the applications in our project. For this reason, we need to use a Servlet container and we chose the Apache Tomcat one. We are currently running Apache Tomcat 6 for Linux on adela server. This version is designed to use Java EE 5.

3.5.4 Eclipse IDE

Eclipse is an integrated development environment (IDE) written primarily in Java. The initial codebase originated from VisualAge, a family of computer integrated development environments from IBM, which included support for a few popular (and not so popular) computer programming languages. In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Language packs provide translations into over a dozen natural languages. It is open source software and it is available for several operation systems.

The Eclipse Project is supported by the most important industry leaders like Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft and Webgain.

I have been using Eclipse 3.3 IDE for developing all the java servlets for our Project. In section *4.5.6 Creating and running a dynamic VoiceXML application* I will explain how I use it to develop VoiceXML applications.



```java
import java.io.BufferedInputStream;


public class GenericPost extends HttpServlet{

  private static final String CONFIG_FILE = "GenericPost.conf";
  private static final String WEB_APPLICATION_NAME = "myservlets";
  private static final String FILE_SEPARATOR = System.getProperty("file.separator");
    // absolute PATH to the servlet .class file. user.dir = "c:\programacion\Tomcat 6.0"
  private static final String CONFIG_PATH = System.getProperty("user.dir")
        + FILE_SEPARATOR + "webapps" + FILE_SEPARATOR + WEB_APPLICATION_NAME
        + FILE_SEPARATOR + "WEB-INF" + FILE_SEPARATOR + "etc" + FILE_SEPARATOR;
  private static final String WEB_APPLICATION_PATH = System.getProperty("user.dir")
    + FILE_SEPARATOR + "webapps" + FILE_SEPARATOR + WEB_APPLICATION_NAME
    + FILE_SEPARATOR;

  public static void main(String[] args)
  {

  }

  /* This Servlet makes a POST or GET to a URL with some parameters. The URL,
   *  parameters and request method (GET or POST) are specified in a configuration file.
   *  The Servlet returns to the client the HTML page it receives as response
   *  of the POST/GET request. All this functionality is implemented in the doGet method.
   */
  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    BufferedInputStream bis = null;
```

```
No consoles to display at this time.
```

Figure 3-9. Eclipse IDE screen shot.

3.5.5 IBM Rational Web Developer & Voice Toolkit

IBM Rational Application Developer for WebSphere Software extends Eclipse with visual construction development. It helps Java developers rapidly design, develop, assemble, test, profile and deploy high quality Java/J2EE, Portal, Web, Web services and SOA applications.

We use IBM Rational Web Developer (RWD) in ACC Project to develop static Voice Applications. It is really easy and intuitive to create a whole application using the visual construct environment that RWD provides. For this purpose it is needed to install the Voice Toolkit which integrates with RWD environment and provides the Communication Flow Builder tool. Using the Communication Flow Builder it is very easy to create the flow diagram of the voice application only dragging and dropping the visual elements and connecting them. Once we have created the Communication Flow, the RWD creates the VXML application file automatically.

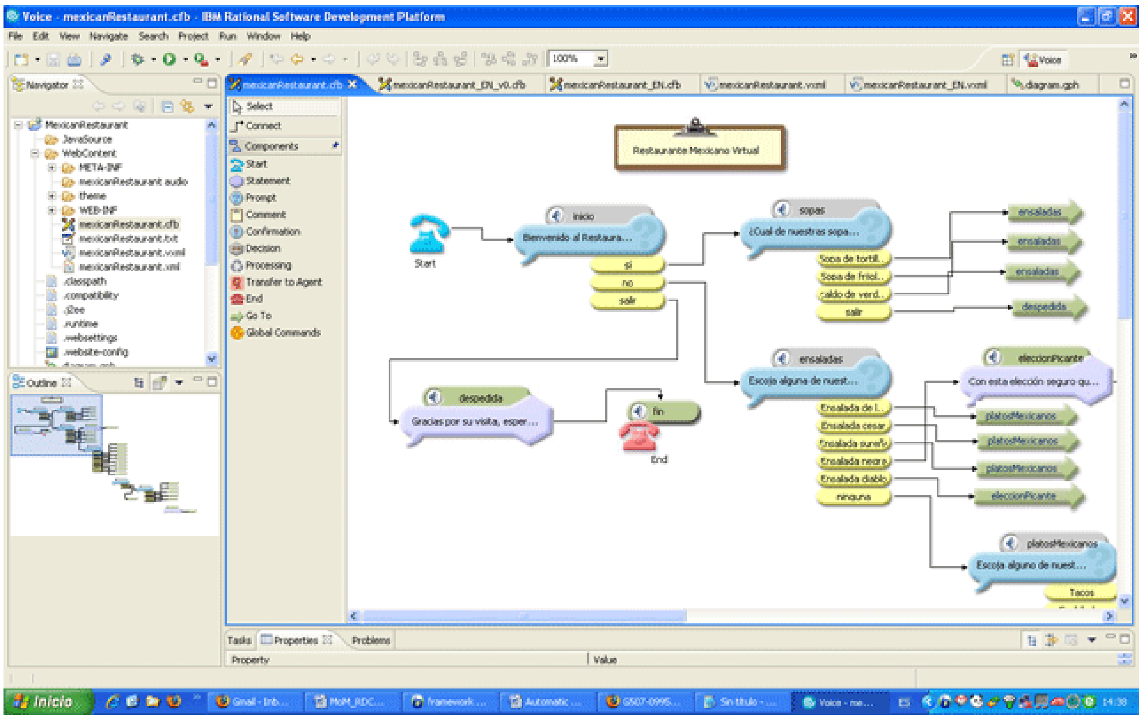


Figure 3-10. IBM Rational Web Developer and Communication Flow Builder screen shot.

3.6 How I configure the software tools

3.6.1 Tomcat 6 Installation and configuration on Fedora Linux

This section will describe step by step what it is needed to do to install and configure Apache Tomcat 6 on our Fedora Linux server.

3.6.1.1 Download binaries and prerequisites

JDK (Java Development Kit)

Tomcat 6.0 was designed to run on J2SE 5.0. You can download Java EE SDK from http://java.sun.com.

Tomcat

Binary downloads of the Tomcat server are available from http://tomcat.apache.org/download-60.cgi.

Set an environment variable CATALINA_HOME that contains the pathname to the directory in which Tomcat 6 has been installed.

Ant

Binary downloads of the Ant build tool are available from http://ant.apache.org/bindownload.cgi. Download and install Ant from the distribution directory mentioned above. Then, add the bin directory of the Ant distribution to your PATH environment variable, following the standard practices for your operating system platform. Once you have done this, you will be able to execute the ant shell command directly. You should use a command like it is shown below and modify your .bashrc file in your home directory:

```
PATH=$PATH:/opt/SDK/lib/ant/bin
```

3.6.1.2 Environment variables required

CATALINA_HOME
It may point at your Catalina "build" directory.

CATALINA_BASE (Optional)
It is the base directory for resolving dynamic portions of a Catalina installation. If not present, resolves to the same directory that CATALINA_HOME points to.

JAVA_HOME
It must point at your Java Development Kit installation. Required to run the with the "debug" or "javac" argument. To set the JAVA_HOME environment variable it is necessary to execute the following command:

```
export JAVA_HOME=/opt/SDK/jdk
```

You can add this line in the .bashrc file of your home directory to do it automatically each time the system runs.

JRE_HOME
It must point at your Java Development Kit installation. Defaults to JAVA_HOME if empty.

3.6.1.3 Configuration files

Server.xml
This is the Tomcat server main configuration file. By default it is at conf/server.xml. We will not modify it and will leave it with the default values.

Build.xml
It is necessary to modify the default build.xml file provided by tomcat at the following line:

```
<property name="catalina.home" value="../../../.."/> <!-- UPDATE THIS! -
->
```

In our case, the replaced line will become like this:

```
<property name="catalina.home" value="/opt/apache-tomcat-6.0.16"/>
```

This modification would be necessary if we would be using the ant tool to manage the compilation of our Java source code files, and creation of the deployment hierarchy. Ant operates under the control of a build file, normally called build.xml, that defines the processing steps required. This file is stored in the top-level directory of your source code hierarchy, and should be checked in to your source code control system.

Web.xml

This is the web application configuration file and controls all the parameters related with each web application. This file will be discussed in section *4.5.6 Creating and running a dynamic VoiceXML application.* There is one web.xml file for each application in the directory path:

```
$CATALINA_HOME/webapps/{my_web_application}/WEB-INF
```

3.6.1.4 Standard directory layout

A web application is defined as a hierarchy of directories and files in a standard layout. Such a hierarchy can be accessed in its "unpacked" form, where each directory and file exists in the filesystem separately, or in a "packed" form known as a Web ARchive, or WAR file. The former format is more useful during development, while the latter is used when you distribute your application to be installed.

To run a web application on Tomcat you will need to arrange your application files in the following hierarchy in your application's "document root" directory:

- *.html, *.jsp, etc. - The HTML and JSP pages, along with other files that must be visible to the client browser (such as JavaScript, stylesheet files, and images) for your application. In larger applications you may choose to divide these files into a subdirectory hierarchy, but for smaller apps, it is generally much simpler to maintain only a single directory for these files.
- /WEB-INF/web.xml - The Web Application Deployment Descriptor for your application. This is an XML file describing the servlets and other components that make up your application, along with any initialization parameters and container-managed security constraints that you want the server to enforce for you. This file is discussed in more detail in the following sections.
- /WEB-INF/classes/ - This directory contains any Java class files (and associated resources) required for your application, including both servlet and non-servlet classes, that are not combined into JAR files. If your classes are organized into Java packages, you must reflect this in the directory hierarchy under /WEB-INF/classes/. For example, a Java class named com.mycompany.mypackage.MyServlet would need to be stored in a file named /WEB-INF/classes/com/mycompany/mypackage/MyServlet.class.
- /WEB-INF/lib/ - This directory contains JAR files that contain Java class files (and associated resources) required for your application, such as third party class libraries or JDBC drivers.

Like most servlet containers, Tomcat 6 also supports mechanisms to install library JAR files (or unpacked classes) once, and make them visible to all installed web applications (without having to be included inside the web application itself.

- $CATALINA_HOME/common/lib - JAR files placed here are visible both to web applications and internal Tomcat code. This is a good place to put JDBC drivers that are required for both your application and internal Tomcat use (such as for a JDBCRealm).

- $CATALINA_BASE/shared/lib - JAR files placed here are visible to all web applications, but not to internal Tomcat code. This is the right place for shared libraries that are specific to your application.

3.6.2 IBM RWD Voice Language and Voice Engines Configuration

Once the Communication flow is done and before the generation of the VXML file, it is needed to configure the Voice Engines and the Speech Language our application is going to use. For doing this, you need to do the following steps:

Menu Window>Preferences

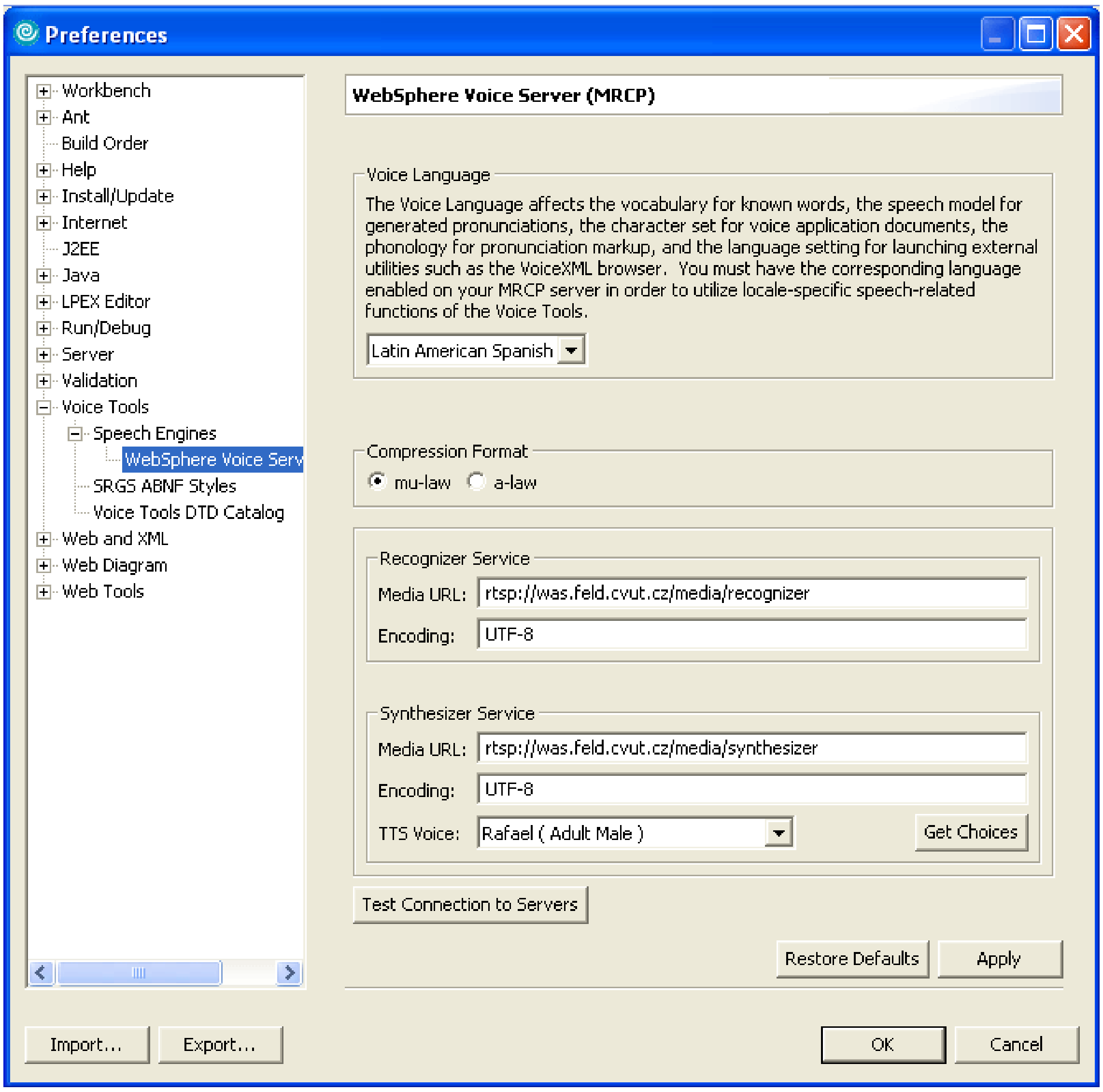
At the left panel of the window that appears select:

Voice Tools > Speech Engines > WebSphere Voice Server

Voice Language: US English / Latin America Spanish

Compression format: mu-law

Recognizer Service
Media url: rtsp://was.feld.cvut.cz/media/recognizer
Encoding: UTF-8
Synthesizer Service
Media url: rtsp://was.feld.cvut.cz/media/synthesizer
Encoding: UTF-8
TTS Voice: Default



WebSphere Voice Server (MRCP)

Voice Language

The Voice Language affects the vocabulary for known words, the speech model for generated pronunciations, the character set for voice application documents, the phonology for pronunciation markup, and the language setting for launching external utilities such as the VoiceXML browser. You must have the corresponding language enabled on your MRCP server in order to utilize locale-specific speech-related functions of the Voice Tools.

Latin American Spanish

Compression Format

[✓ mu-law] a-law

Recognizer Service

Media URL: rtsp://was.feld.cvut.cz/media/recognizer

Encoding: UTF-8

Synthesizer Service

Media URL: rtsp://was.feld.cvut.cz/media/synthesizer

Encoding: UTF-8

TTS Voice: Rafael (Adult Male)

Figure 3-11. IBM RWD Voice Language and Voice Engines configuration.

3.7 Creating and running a dynamic VoiceXML application

In this section I am going to explain how to use all the software tools explained in previous sections to create a VoiceXML application and run it on our platform. This section is written like a step by step manual in order it were easier to follow all the procedure and to show clearly how to use each tool.

Briefly the procedure is: write the servlet code in Eclipse IDE, copy the generated servlet binary files into the server (inside the Tomcat directory) and update Tomcat and VE to recognize the new application. See the following subsections where there are explained more detailed each step.

3.7.1 Create a new Java Project in Eclipse IDE

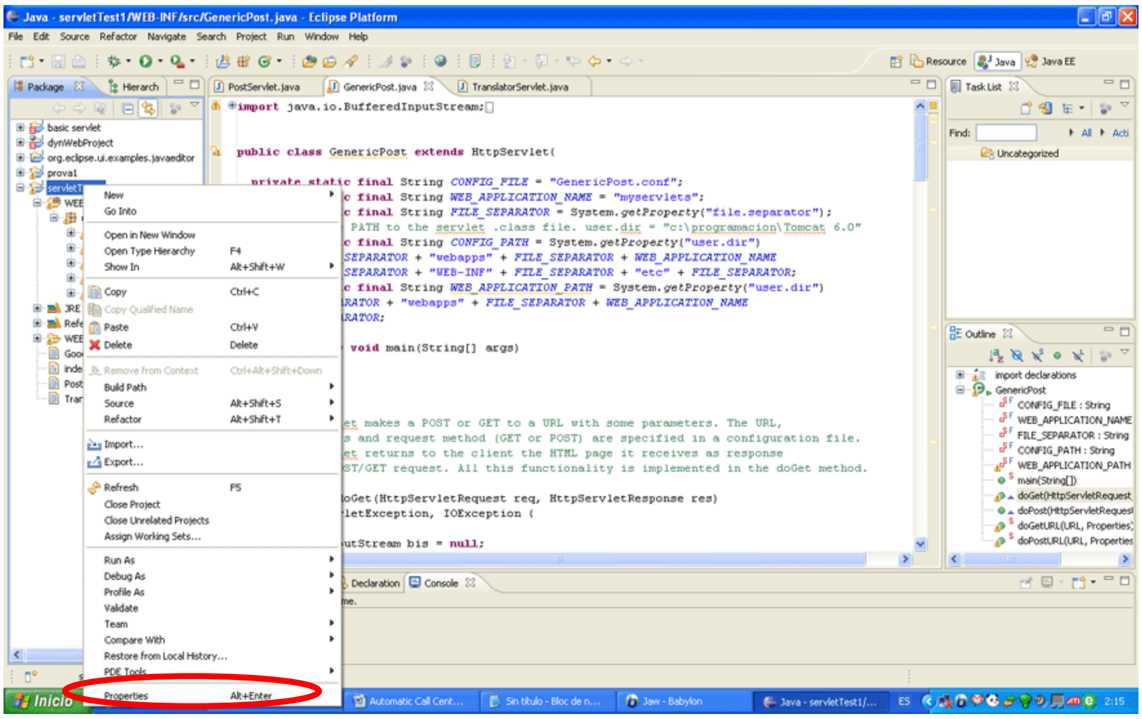
Although Eclipse gives you the chance to create a Dynamic Web Application Project when you try to create new project, we will not use this option. If you choose this, Eclipse assist you in all the tasks involved in the creation of this kind of applications, but all the procedures within this kind of applications are very complex and take much time to understand how to use them. Therefore, like for our purposes we do not need most of this assistance, we are going to create a simply Java Project, which will provide us what we need and it is much easier to understand.

To create the new project, you simply have to go to the menu “File > New > Java Project” and follow the wizard. After that, you can begin adding new classes and writing the servlets code.

3.7.2 Adding JEE libraries in Eclipse IDE

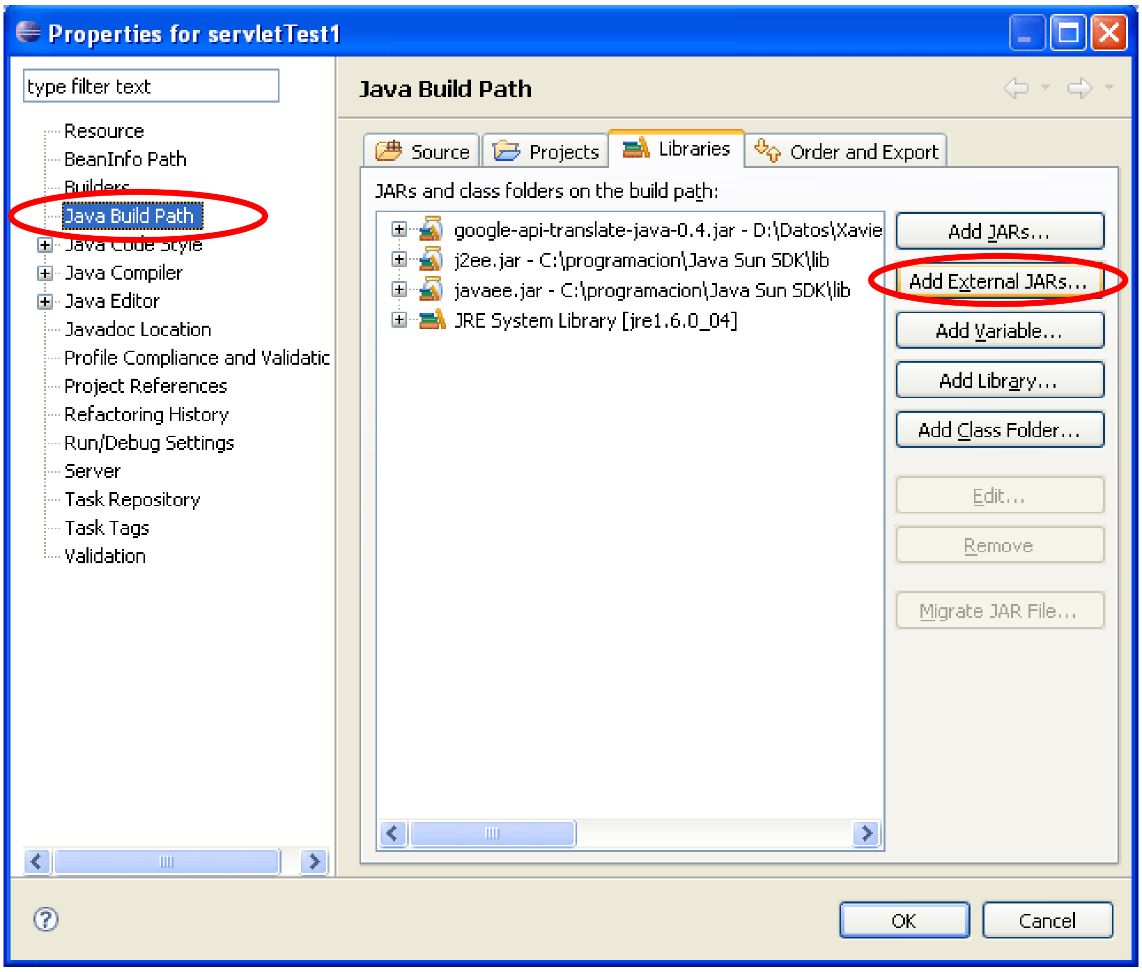
In the package explorer, right click on the project name which you are going to develop the servlet in. Select Properties on the popup menu that appears. On the left side of the properties windows select “Java Build Path” and then on the right side of the properties windows choose the Libraries tab. Click the button called “Add External JARs…” and then select the javaee.jar file (or j2ee.jar in some JDK versions) located in:

```
$JAVA_EE_SDK/lib/javaee.jar
```



```
import java.io.BufferedInputStream;

public class GenericPost extends HttpServlet{

  private static final String CONFIG_FILE = "GenericPost.conf";
c final String WEB_APPLICATION_NAME = "myservlets";
c final String FILE_SEPARATOR = System.getProperty("file.separator");
 PATH to the servlet .class file. user.dir = "c:\programacion\Tomcat 6.0"
c final String CONFIG_PATH = System.getProperty("user.dir")
SEPARATOR + "webapps" + FILE_SEPARATOR + WEB_APPLICATION_NAME
SEPARATOR + "WEB-INF" + FILE_SEPARATOR + "etc" + FILE_SEPARATOR;
c final String WEB_APPLICATION_PATH = System.getProperty("user.dir")
RATOR + "webapps" + FILE_SEPARATOR + WEB_APPLICATION_NAME
RATOR;

 void main(String[] args)

et makes a POST or GET to a URL with some parameters. The URL,
s and request method (GET or POST) are specified in a configuration file.
et returns to the client the HTML page it receives as response
ST/GET request. All this functionality is implemented in the doGet method.

oGet(HttpServletRequest req, HttpServletResponse res)
letException, IOException {

utStream bis = null;
```

Figure 3-12. Adding libraries step 1.


Properties for servletTest1

type filter text

- Resource
- BeanInfo Path
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Profile Compliance and Validatic
- Project References
- Refactoring History
- Run/Debug Settings
- Server
- Task Repository
- Task Tags
- Validation

Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

- google-api-translate-java-0.4.jar - D:\Datos\Xavie
- j2ee.jar - C:\programacion\Java Sun SDK\lib
- javaee.jar - C:\programacion\Java Sun SDK\lib
- JRE System Library [jre1.6.0_04]

Figure 3-13. Adding libraries step 2.

3.7.3 Create the main procedure in the Servlet class

It is needed to create an empty public void main procedure in the HTTPServlet class, due if it is not present, eclipse will not compile the class using the Java perspective. We can use the JavaEE perspective to create an HTTPServlet (it would be better) instead, but it is more complex to understand how this perspective works. If we use the Java Perspective with classes with Main procedure, we can work with Eclipse like we were writing a normal Java application and, later, copy the files manually in the Tomcat web server directory structure.

3.7.4 Copy the binary files to the server

Once, you have compiled the java code (choose “Run Java Application” if you are asked, not “Run on Server”) it is need to copy the generated class file to the server (adela) inside the Apache-Tomcat webapps directory. The destination path for the class file should be:

```
$TOMCAT_DIR/webapps/{my_web_application}/WEB-INF/classes
```

If you have used some external API not included neither in the Java Standard SDK nor in the JavaEE SDK, you should include the JAR file with the API packages in the following path:

```
$TOMCAT_DIR/webapps/{my_web_application}/WEB-INF/lib
```

You should restart the server every time you copy a new file or if an existing file has been modified.

3.7.5 Tomcat application configuration

You need to configure your Tomcat Application to allow the new servlet be reachable for the users. You need to modify the `web.xml` file which is at

```
$TOMCAT_DIR/webapps/{my_web_application}/WEB-INF
```

Adding the following lines (it is used the TransaltorServlet configuration as example):

```
<servlet>
        <servlet-name>TranslatorServlet</servlet-name>
        <servlet-class>TranslatorServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>TranslatorServlet</servlet-name>
        <url-pattern>/translator.vxml</url-pattern>
</servlet-mapping>
```

Figure 3-14. Lines to be added in the application's web.xml file.

Where each tag means:

Tag	Description
<servlet>	In this section we define internal configuration (used inside Tomcat server) for each servlet.
<servlet-mapping>	In this section we define the relative URL where each servlet will be reachable.
<servlet-name>	Specifies the name it is known a servlet inside the web.xml file. It is defined with this same tag inside the <servlet> section.
<servlet-class>	In the <servlet> section it defines the class file which is associated with a servlet-name.
<url-pattern>	Defines the relative url for a specified servlet where the servlet will be reachable. It is used inside the <servlet-mapping> section. The value of this tag will be "/{servlet_relative_url}", which means that the servlet will be reachable at: http://myserver.com/{myWebApplication}/{ servlet_relative_url }

Table 3-17. Web.xml file tags.

If the servlet have to provide a VXML output to be interpreted by the Voice Enabler (VE), as is the case of all VoiceXML applications; the <url-pattern> has to end with “.vxml”, because the VE only is able to read this file format. Although the output of the servlet would be the same VXML code, if the <url-pattern> don’t end with “.vxml” the VE will not process it.

You should restart the server every time the `web.xml` is modified before the changes take effect.

Once it has been done, the new servlet will be reachable at the following URL (in this example):

```
http://myserver.com/{myWebApplication}/translator.vxml
```

The VoiceXML applications I have created so far, need the Tomcat server were restarted from the $CATALINA_BASE directory. It is because the servlets need to know the absolute path where there are located their binary files. Maybe this issue could be solved in the future.

Therefore, run the restart executable file from the path $CATALINA_BASE, by doing:

```
cd /opt/apache-tomcat-6.0.16
```

And once you are in this path you have to execute the following command:

```
./bin/restart.sh
```

3.7.6 Configuring VE to use the VXML Application generated by the servlet

At this point, it only remains to modify the VE configuration files to make available the new VoiceXML application in a specified phone number. When it was done and the user calls to the assigned phone number, VE will launch our Voice Application to attend him.

To assign a phone number to the Voice Application generated by the servlet, you should modify the `DefaultInboundCCXML.jsp` VE configuration file. It could be found in the following path in `adela` server:

```
/opt/WebSphere/AppServer/installedApps/adela/
        WVX5.1-adela.ear/wvxweb.war/DefaultInboundCCXML.jsp
```

Inside this file you should find the text block showed in

Figure **4-5**. You have to add a new line at the end of this block, following the examples of the other lines. In the new line you should set an available phone number and link it with the URL where the VXML file which contains the Voice Application can be found.

```
<!-- Prepare the VoiceXML application based on called number -->
<!--      700 = IBM default voice application     -->
<!--      701 = Identity application      -->
<!--      702 = Mexican Restaurant application    -->
<!--      703 = CVUT Call Center application      -->
<!--      704 = Dynamic Time application  -->
<!--      705 = Translator application -->
<!--      777 = Icecream Shop application -->
<!--      7xx = Voice application input as parameter to VE        -->

<script><![CDATA[
  var number = extractNumber(evt.connection.local);
  if(!isNaN(number)) switch(number){
    case 700: vxml_app = "DefaultVXML.jsp"; break;
    case 701: vxml_app = "http://localhost/vxml/identity.vxml"; break;
    case 702: vxml_app = "http://localhost/vxml/mexicanRestaurant_v2.vxml"; break;
    case 703: vxml_app = "http://localhost/vxml/cvutCallCenter.vxml"; break;
    case 704: vxml_app = "http://localhost:8080/myservlets/dynamicVXML_hour.vxml";
break;
    case 705: vxml_app = "http://localhost:8080/myservlets/translator.vxml"; break;
    case 706: vxml_app = "http://localhost/vxml/mexicanRestaurant_v2_EN.vxml"; break;
    case 777: vxml_app = "http://localhost/vxml/icecream_alt.vxml"; break;
    default: vxml_app = "http://localhost:8080/vxmlide/file?phone=" + number; break;
  }
]]></script>
```

Figure 3-15. Text block to be modified in DefaultInboundCCXML.jsp file.

3.8 Application schema

If you want to understand the way I have followed to accomplish the aim of Real Time Speech Translator Project, it would be really interesting to have a clear idea of the different parts this big application is made up. Knowing this you will be able to understand de different step explained in section *4.8 Achieved goals* because each of them has the purpose to accomplish one of these tasks. In Figure 4-6, you can see the whole schema of Real Time Speech Application and its main parts.

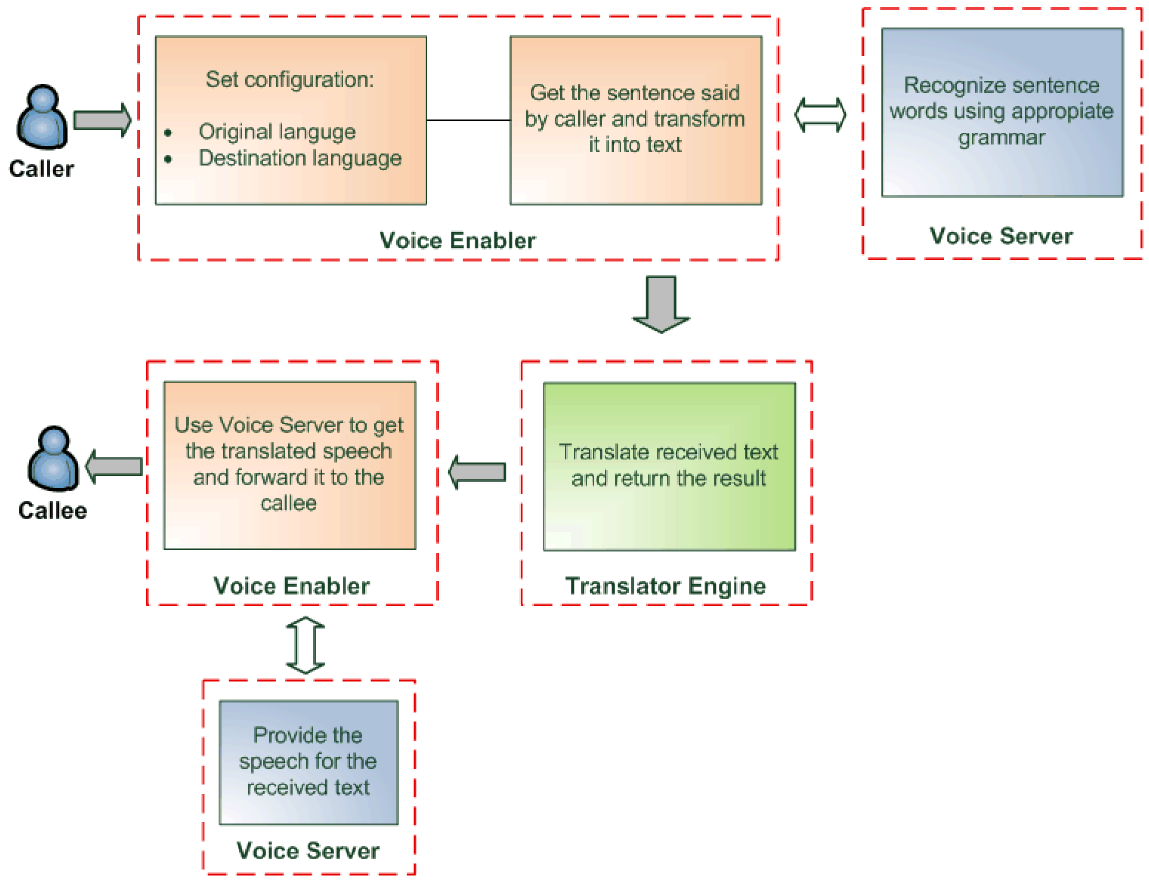

Caller
Set configuration:
Original languge
Destination language
Get the sentence said by caller and transform it into text
Voice Enabler
Recognize sentence words using appropiate grammar
Voice Server
Callee
Use Voice Server to get the translated speech and forward it to the callee
Voice Enabler
Translate received text and return the result
Translator Engine
Provide the speech for the received text
Voice Server

Figure 3-16. Real Time Speech Translator Application schema.

3.9 Text To Speech and Speech Recognition

This point is where we found one of the biggest problems to cope with during the development of this project. When we begin this project, we thought the Speech Recognition Engine transformed internally the speech to text and then it compares the obtained text with the text written in the VXML code. This assumption was wrong and if we thought this, was due our lack of knowledge with this technology. Actually, the Speech recognition is done by converting the written text in the VXML file into audio and then comparing this with the audio coming from the caller. Therefore, it is needed to have a grammar file which contains all the words and sentences you want your application were able to recognize. This is a big obstacle we have not solved yet, because for our purposes we need to have a full recognition of whatever the caller says and it is almost impossible to generate such a big grammar.

3.10 Achieved goals

In the following subsections I am going to explain the aims achieved in the Real Time Voice Application so far. As it is a very complex project, I have divided it in several simpler parts. Each of these parts has the proposal of solving one of the requisites or implements one of the features needed for the Real Time Voice Translator. The idea is to do small steps and adding each time a new feature to finally joint all them to create the whole application.

The following subsections are sorted from the first and simpler achieved aim to the last and more complex accomplished aim. Each time one of these goals is achieved, a new step forward is done to reach successfully the objective of the Real Time Voice Translator Project.

3.10.1 Setup WebSphere Server to run VoiceXML Applications

Our team first aim was to know if all the ACC Project Platform was working well. To check it, it was needed to create a simply Voice Application and demonstrate it works well on our platform. For this purpose we set up the ICE Cream Shop Application, which is supplied as an example application with IBM Rational Web Developer, and we ran it on our servers successfully.

With this task, we demonstrate that our servers and the software installed on them were working correctly and then we were able to create more Voice Applications.

3.10.2 Use IBM Rational Web Developer to implement VoiceXML Applications

Next step, were learning to use the IBM Rational Web Developer, the software tool supplied us by IBM to create Voice Applications. For this purpose, our team creates the ACC Call Center Application. This application simulates the university call center and was able to attend a student call.

Now we had the basic knowledge to create simply static Voice Applications.

3.10.3 Configure and use different languages within our Voice Engine

The Voice Server provided us for IBM has support for several languages, but so far, we only had done tests using US-English language. Our proposal now was checking that other supported languages work correctly and get the knowledge to create voice applications in other languages. For this aim, we develop the Mexican Restaurant Application. This is a Spanish voice application that simulates the call center of a Mexican restaurant where clients could call to order their meal and choose the most typical dishes of the Mexican cuisine. We also create a clone of Mexican Restaurant application using the English language. It was done to be able in the future to give the choice to the caller of being attended in English or Spanish.

3.10.4 Generate dynamic VXML code

Until now we have developed several different voice applications, but all they are static voice applications. It means that all the call flow have to be predicted in the implementation time. It also means that all the information they can supply and the options they offer to the caller has to be known at the development time. Once the application is running, there are not any chance to do anything it was not being considered when the programmer wrote the application. This is because VoiceXML applications are based in static files written in VXML language.

This issue is a big obstacle to create many Voice Applications which would be able to provide very interesting services that are not accessible or known at development time. In the case of Real Time Voice Application, the developer will know neither which sentence will the caller want to translate nor the translated sentence to return.

For this reason, our next step was getting the knowledge to create dynamic voice applications. To achieve it, I do a deep research within many different technologies to find out which ones we needed for our purposes and which tools we need to use. After this research, I created the Dynamic Time Application, our first Dynamic Voice Application. This is a simple application that says the current time to the caller. For doing it, the application gets the current time from the machine and generates on the fly the VXML file which will be sent to the VE to be interpreted and finally be played to the caller.

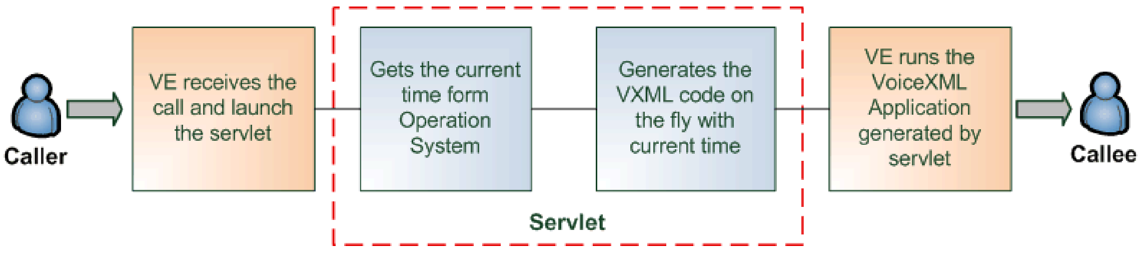

Caller
VE receives the call and launch the servlet
Gets the current time form Operation System
Generates the VXML code on the fly with current time
VE runs the VoiceXML Application generated by servlet
Callee
Servlet

Figure 3-17. Dynamic Time Application schema.

3.10.5 Make queries to websites and process the response

Once we have the capability to create dynamic voice applications, we have to use this knowledge for going forward to our aims in the project. To do this, next step is having the capability to obtain data form a website. This would be very interesting for a big range of applications that were based on web services. Some examples would be Real Time Voice Translator, which needs to get the translated sentences from web translation services; Wikipedia Voice Project, which enables to get the information from Wikipedia and play it on your mobile phone; the Weather Forecast Application which gets the weather forecast form a website and plays it back to the user;…

To achieve this aim I created the GenericPost servlet. This is a servlet that implements all the steps necessary to carry out a Voice Application based on web services. This servlet is able to make a query to any website with independence of whatever HTTP queries it uses: HTTP POST or HTTP GET. GenericPost servlet makes a query to the website, gets the data from it, and forwards it to the user's web browser. In our Voice Applications we will use this same schema but we will forward the data obtained from the website to the VE instead to a web browser.

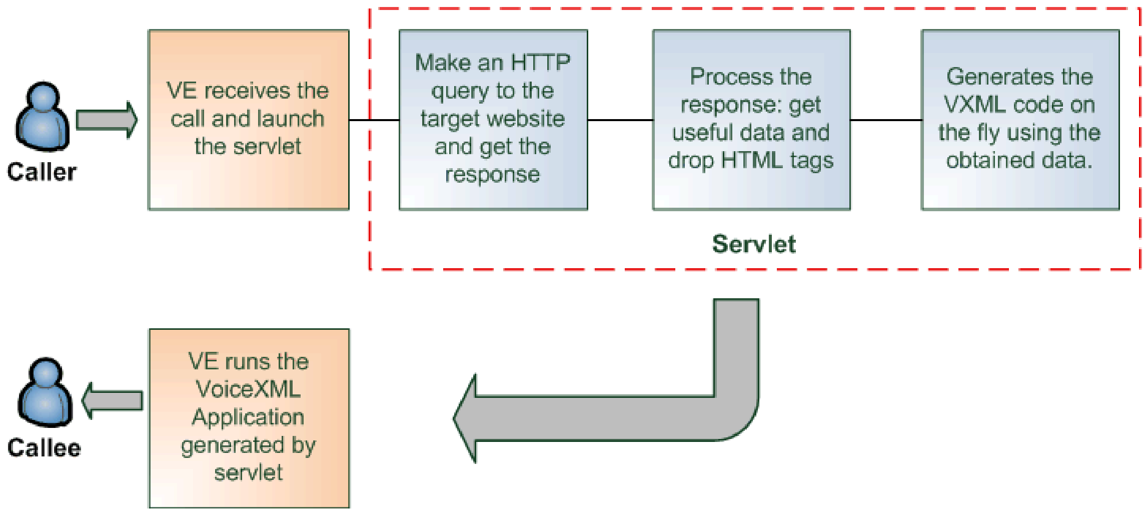

Caller
VE receives the call and launch the servlet
Make an HTTP query to the target website and get the response
Process the response: get useful data and drop HTML tags
Generates the VXML code on the fly using the obtained data.
Servlet
Callee
VE runs the VoiceXML Application generated by servlet

Figure 3-18. Schema for a Voice Application which needs getting data from a website.

3.10.6 Make queries to Google Translator

At this point, we were very close to one of the main aims of our Project: given a sentence, translate it and plays it back to the user. To accomplish this goal, we had to modify the GenericPost servlet to make queries to Google Translate service and integrate it in a Voice Application which plays the translated sentence to the user.

It appeared to be a simply step, but after modifying the GenericPost servlet for our purposes Google Translate denied us the access to its service. After some research, I find out that all Google Services denies access to their services to all automatic applications. They only accept human users to avoid DoS attacks. To avoid this handicap, I modified the behavior of GenericPost servlet to emulate the behavior of a human user which uses a Mozilla web browser. I change the HTTP header our application uses to connect with the server for cheating it. In this way, now we were able to access to all Google web services.

Despite, at this point, we already had a manner to get the translated sentences using the GenericPost servlet, I continue my research and I find out a better way to do this. I find out a beta API for Java that enables to use Google Translate Services, without doing HTTP queries.

3.10.6.1 Google Translator API

Google Translator API for Java is an API for Java programming language developed for Google to enable Java applications to use their services. Although it is currently a beta version, the tests I done showed that it works well for our purposes. Using this API is a better way to access Google Translator service, because it save us to process the returned HTML code to get the translated sentence and drop the rest of HTML code that generates the visual interface of the page. Another advantage is that we do not depend on the interface changes done in the web in the future, which would modify the HTML code and would force us to change the algorithms we use to process it.

3.10.7 Get dynamic data from a VoiceXML Application

Although we have the way to get data from any web service and play it back to the user, we also need to know the way for integrating it in a whole Voice Application. It means to know how to launch our servlets, which get this data and return it to the user, from the VXML code of a Voice Application. For this purpose we decided to use the DATA tag of VXML language which enables a Voice Application to get data from a web service through a HTTP query.

Therefore, we needed to make our servlets be reachable for this tag. To realize this, I modified all servlets done up to now to were able to receive the configuration parameters they need from a HTTP query (HTTP GET or HTTP POST). Before doing this, they get the parameters from a configuration file. With this modification, Translator Application, for example, is able to receive the sentence to be translated and the original and destination languages through a HTTP query.

4 Conclusions

When we began this project several months ago, it was a big challenge for all our team. For me, it was the first time I use some of these technologies, like VoiceXML. We all have had to do a hard work to learn how to manage them and reach our aims. It has been a very nice experience for me, and I have had the possibility to do a true team work with other very qualified people. This has permitted us to benefit each other and, in my case, improve my personal skills quickly.

Regarding the Real Time Voice Translator, we achieved important goals. Although I have not reach the final aim of creating an application which was able to translate in real time a conversation between two callers, I have done most of the necessary steps to achieve it. It is a very complex Project and requires a lot of intermediate steps, which I was not able to imagine at the beginning due my lack of knowledge on the technologies involved in. Despite this, I think that linking the achieved mid-steps and adding few more features I could reach the goal we proposed at the beginning. I also think that, with the knowledge acquired during this time, achieve the new aims would be more easy and quickly due to the most difficult parts have already been done.

For a future evolution of my work on the Real Time Voice Translator Project, I propose to do the following steps to solve the problems I have found or to add new features:

- Create a Multilanguage application, i.e. use different languages in a same VXML file. It will be necessary for example to give the caller the chance to choose his language and, after his choice, use the selected language to speak with him.
- Find out how to launch a Voice Application from another Voice Application. It will be necessary to use this in many situations. For example, we could create an application which gives the chance to the caller to choose which application he wants to use, and after it, redirect the call to the chosen application.
- Find out the way to do a cyclic application, i.e. an application that never ends until the user wants. It will be necessary to implement this feature in many Voice Applications. For example, it would be necessary in the Real Time Speech Translator to be able to translate all the sentences the user says indefinitely until he wants to finish the conversation.
- Try different ways to get data into a VoiceXML Application. Now we have only tested the DATA tag, but it would be necessary to try other options to know the benefits of each one and know when it is better to use one or other.
- Find out an efficient algorithm to process the HTML code and get the useful data within it for our purposes.
- Find the ways to improve the waiting time when we perform a query to a web site. Maybe it would be possible to use a multithreading solution to make simultaneous queries to several websites at the same time or use a buffer to store the responses.

- Solve the grammar problem to achieve full conversation recognition without the restriction of a set of words fixed in advance as it is the case of the current applications.

During the development of this project we have faced a lot of obstacles that were coming up and we turned out successful in most of them. I think the hardest part of the way has been done and now we have the chance to continue improving our Voice Applications and create new and even more complex ones applying the knowledge we have obtained up to now.

I think that VoiceXML technology has an amazing future and there are a wide range of applications where it could be used. ACC team has demonstrated its good skills and it is carrying out very ambitious projects; hence I think it has a good chance to become a reference Team in Voice Applications development.

5 APPENDIX A: Definitions

Private Branch eXchange (PBX)

Private Branch eXchange (PBX) is a telephone exchange that serves a particular business or office, as opposed to one that a common carrier or telephone company operates for many businesses or for the general public. PBXs are also referred to as:

- **PABX** - Private Automatic Branch eXchange.
- **EPABX** - Electronic Private Automatic Branch eXchange.

Trunk Line

When dealing with a PBX, trunk lines are the phone lines coming into the PBX from the telephone provide

Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. It can be used to create two-party, multiparty, or multicast sessions that include Internet telephone calls, multimedia distribution, and multimedia conferences (RFC 3261). SIP is designed to be independent of the underlying transport layer; it can run on TCP, UDP, or SCTP.

Media Resource Control Protocol (MRCP)

MRCP is the Media Resource Control Protocol proposed to the IETF. It is a communication protocol which allows speech servers to provide various speech services (such as speech recognition and speech synthesis) to its clients. Typically, this means the server software will be running on one computer and clients can send MRCP messages to the server over a network, usually on top of another protocol, such as RTSP or TCP. There are two versions of the MRCP protocol. Version 2 uses SIP as the control protocol, whereas version 1 uses RTSP.

Time Streaming Protocol (RTSP)

The Real Time Streaming Protocol (RTSP), developed by the IETF and created in 1998 as RFC 2326, is a protocol for use in streaming media systems which allows a client to remotely control a streaming media server, issuing VCR-like commands such as "play" and "pause", and allowing time-based access to files on a server.

VoiceXML (VXML)

VoiceXML (VXML) is the W3C's standard XML format for specifying interactive voice dialogues between a human and a computer. It allows voice applications to be developed and deployed in an analogous way to HTML for visual applications. Just as HTML documents are interpreted by a visual web browser, VoiceXML documents are interpreted by a voice browser. A common architecture is to deploy banks of voice browsers attached to the public switched telephone network (PSTN) so that users can use a telephone to interact with voice applications.

Call Control eXtensible Markup Language (CCXML)

Call Control eXtensible Markup Language (CCXML) is an XML standard designed to provide telephony support to VoiceXML. Its current status is a W3C Working Draft, adopted 19 January 2007. Where as VoiceXML is designed to provide a Voice User Interface to a voice browser, CCXML is designed to inform the voice browser how to handle the telephony control of the voice channel. The two XML applications are wholly separate and are not required by each other to be implemented.

One critical thing to understand is that CCXML is not a media/dialog language like VoiceXML. It only provides support to move calls around and connect them to dialog resources. CCXML does not provide any dialog resources on its own. (Note: A dialog resource is anything that interacts with a caller via voice, such as a VoiceXML platform or even a second caller at another location.)

Voice browser

A voice browser is a web browser that presents an interactive voice user interface to the user. In addition, it typically provides an interface to the PSTN or a PBX. Just as a visual web browser works with HTML pages, a voice browser operates on pages that specify voice dialogues. Typically these pages are written in VoiceXML, the W3C's standard voice dialog markup language, but other proprietary voice dialogue languages remain in use.

A voice browser presents information aurally, using pre-recorded audio file playback or using text-to-speech software to render textual information as audio. A voice browser obtains information using speech recognition and keypad entry (e.g., DTMF detection).

As speech recognition and web technologies have matured over the past decade, thousands of voice applications have been deployed commercially and voice browsers are supplanting traditional proprietary IVR systems. Scores of companies provide voice browsers. These take the form of software, packaged hardware/software solutions or hosted solutions.

Voice Enabler

Hardware and software that enables PC to PC voice communications or PC to telephone long distance communications.

Voice Server

It’s the combination of hardware and software that enables voice communications between users. It can manage simultaneously many calls flows to make possible the communication between one user and his speakers.

CallXML

Voxeo proprietary Call Control Language. Its aim is the same as CCXML.

Interactive Voice Response (IVR)

In telephony, interactive voice response, or IVR, is a phone technology that allows a computer to detect voice and touch tones using a normal phone call. The IVR system can respond with pre-recorded or dynamically generated audio to further direct callers on how to proceed. IVR systems can be used to control almost any function where the interface can be broken down into a series of simple menu choices. Once constructed IVR systems generally scale well to handle large call volumes.

Dual-tone multi-frequency (DTMF)

Dual-tone multi-frequency (DTMF) signaling is used for telephone signaling over the line in the voice-frequency band to the call switching center. The version of DTMF used for telephone tone dialing is known by the trademarked term Touch-Tone, and is standardised by ITU-T Recommendation Q.23.

Speech recognition

Speech recognition (in many contexts also known as automatic speech recognition, computer speech recognition or erroneously as voice recognition) is the process of converting a speech signal to a sequence of words in the form of digital data, by means of an algorithm implemented as a computer program.

Hidden Markov model (HMM)-based speech recognition

Modern general-purpose speech recognition systems are generally based on HMMs. These are statistical models which output a sequence of symbols or quantities. One possible reason why HMMs are used in speech recognition is that a speech signal could be viewed as a piece-wise stationary signal or a short-time stationary signal. That is, one could assume in a short-time in the range of 10 milliseconds, speech could be approximated as a stationary process. Speech could thus be thought as a Markov model for many stochastic processes (known as states).

Dynamic time warping (DTW)-based speech recognition

Dynamic time warping is an approach that was historically used for speech recognition but has now largely been displaced by the more successful HMM-based approach. Dynamic time warping is an algorithm for measuring similarity between two sequences which may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another they were walking more quickly, or even if there were accelerations and decelerations during the course of one observation. DTW has been applied to video, audio, and graphics -- indeed, any data which can be turned into a linear representation can be analyzed with DTW.

State Chart XML (SCXML)

SCXML stands for State Chart XML : State Machine Notation for Control Abstraction. It is an XML language which provides a generic state-machine based execution environment based on Harel statecharts.

SCXML is able to describe complex state-machines. For example, it is possible to describe notions such as sub-states, parallel states, synchronization, or concurrency, in SCXML.

Extensible Stylesheet Language Transformations (XSLT)

Extensible Stylesheet Language Transformations (XSLT) is an XML-based language used for the transformation of XML documents into other XML or "human-readable" documents. The original document is not changed; rather, a new document is created based on the content of an existing one.[2] The new document may be serialized (output) by the processor in standard XML syntax or in another format, such as HTML or plain text.[3] XSLT is most often used to convert data between different XML schemas or to convert XML data into HTML or XHTML documents for web pages, creating a dynamic web page, or into an intermediate XML format that can be converted to PDF documents.

6 APPENDIX B: Acronyms

VXML
VoiceXML (VXML)

CCXML
Call Control eXtensible Markup Language

IVR
Interactive Voice Response

DNIS
Dialed number information service

DTMF
Dual-tone multi-frequency

HMM
Hidden Markov model

DTW
Dynamic time warping

CCXML
Call Control eXtensible Markup Language

VoiceXML
Voice eXtensible Markup Language

SRGS
Speech Recognition Grammar Specification

SISR
Semantic Interpretation for Speech Recognition

SSML
Speech Synthesis Markup Language

PLS
Pronunciation Lexicon Specification

ECMAScript
Scripting language supported by most voice browsers

SCXML
State Chart XML

XSLT
Extensible Stylesheet Language Transformations

MRCP
Media Resource Control Protocol

RTSP
Time Streaming Protocol

SIP
Session Initiation Protocol

PBX
Private Branch eXchange

PABX
Private Automatic Branch eXchange

EPABX
Electronic Private Automatic Branch eXchange