A Flexible Client-Server Model for Multilingual CTS/TTS Development

Tibor Ferenczi*, Géza Németh*, Gábor Olaszy**, Zoltán Gáspár*

**Department of Telecommunications and Telematics,*
Technical University of Budapest, Hungary
E-mail: ferenczi@ss20.ttt.bme.hu
nemeth@ttt.bme.hu
gaspar@ss20.ttt.bme.hu
***Phonetics Laboratory, Linguistics Institute of the Hungarian Academy of Sciences, Budapest, Hungary*
E-mail: olaszy@ttt-202.ttt.bme.hu

ABSTRACT

The efficiency of the development of CTS/TTS systems is influenced by the features and services of the software development tools used in the development process. A development system should be highly flexible, informative and user friendly to fulfil all or almost all the requirements the researcher could have. In this paper we present a development system, MVoxDev, that can provide an informative and flexible environment for the development of multilingual CTS/TTS systems. The development system gives aid to inspect and modify all the constituent parts of the CTS/TTS system as a client of the developed CTS/TTS system.

1 INTRODUCTION

In this paper a highly flexible tool for multilingual CTS/TTS development is presented. The aims that were addressed at the creation of the development system were to provide multilevel tools for researchers, technology for multilingual development and a convenient way to the application programmers to interface with the speech synthesizer. As the area of application is getting broader and broader and the application developer can be an expert on the field of application (e.g. reading advertisements, weather forecasts, applications in industry, stock exchange, telephony, handicapped people etc.) and thus he or she can tailor the acoustic database, the linguistic rules and exceptions to his or her needs.

The requirement to have a multilevel development system comes from the need, that the different representations (tagged text, analysed text, sound codes, physical representations of speech data, etc.) of text to be uttered should be presented to the researcher at the same time, synchronized to each other and also there has to be the opportunity to manipulate these objects.

Finally we wanted to separate the run-time application environment from the development functions. Figure 1 gives an overview about the system using the client/server model:

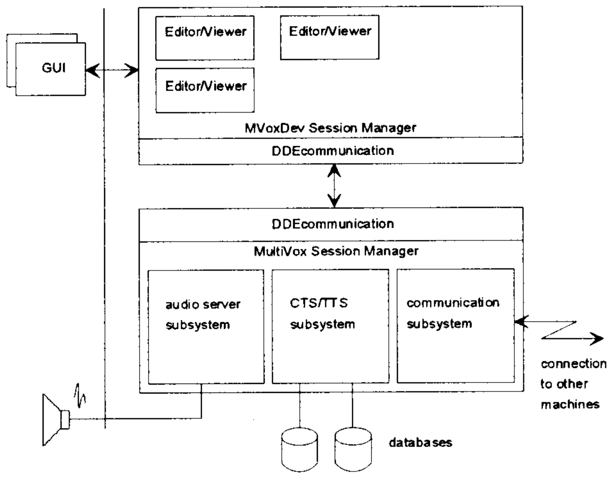

Figure 1 - Block diagram of the MultiVox CTS/TTS multilingual development system. The development system acts as a client of the run-time CTS/TTS system to be developed.

2 FUNCTIONS FOR CTS/TTS DEVELOPMENT

We introduce below the functions of the multilevel CTS/TTS tools for linguistic design and that of the tools for database development.

Development systems such as Speech Maker [1] are mainly for the development of the text-analysis part of the TTS system. Although COMPOST [2] and Festival [3] provides tools for the research of both the text-analysis component and client/server application development, it does not give aid for the development of the acoustic database part of the TTS system.

The services of the MVoxDev development system are categorized by their main functions. So we grouped them into two categories. The first gives tools for multilevel inspection and manipulation of all kind of data to the researcher during the inspection and resynthesis process. The other group provides services for database development.

2.1 MULTILEVEL INSPECTION AND MANIPULATION TOOLS

We modelled the speech generation process as a chain of components which take part in the generation of uttered speech (Fig. 2). The whole speech synthesizer consists of three types of components. Some components as e.g. the letter-to-sound converter, the prosody generator part, the speech synthesizer are viewed as active parts of the system, meaning they act upon the received information from the previous component of the chain with the aid of some external information (e.g. letter-to-sound rules, intonation pattern information, acoustic database etc.) that we regard as passive parts of the CTS/TTS system. The third kind of components are the buffers.

The active parts have inner variables to desribe the state of the active component and hold the original input and the transformed data. The passive components are also loaded into the active components and hence they can be viewed as state variables, but they own the property of persistency (i.e. they have to be stored on storage media such as harddisk drives between consecutive sessions) and because of that they should be handled in a different way.

The MVoxDev development system provides access to the inner variables for the researcher and the ability to modify them on the fly, but he or she can store only information which is persistant, like the acoustic database or letter-to-sound rules. Giving access to the variables in the active component - displaying the states of inner state variables and providing mechanisms for modifying them - makes the decision easier regarding what parts of the active component need further development, improvement or correction.

We had to draw a line between development, research and debugging. This resulted in the architecture that the system gives an insight view into the active component while direct tools for passive components.

The third kind of components are the buffers. Buffers are data storage units to hold data for the active components. They contain their input and output. These buffers are categorized into two classes. The „default" buffers are part of the processing chain.

The other class contains the referable buffers. Reference means that the data to be processed has additional information about other sources of data (like how the `<sound src=>` tag refers to a sound source in SSML [4]). The data stored in buffers can be of any kind and have to be „understandable" for the component and the

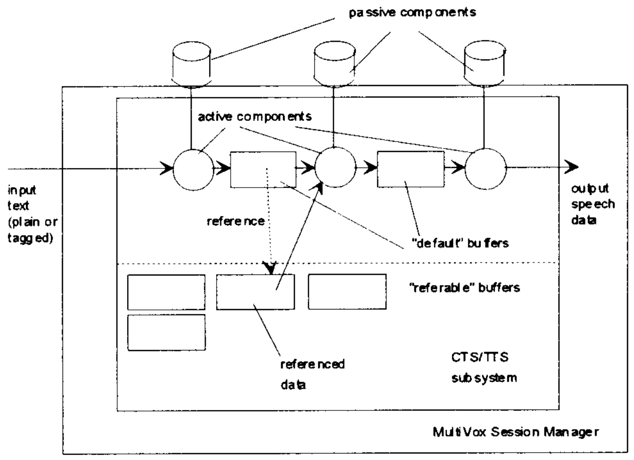

Figure 2 - Block diagram of the MultiVox CTS/TTS multilingual synthesizer. The synthesizer has three kind of components: active, passive components and buffers.

rest of the processing chain behind the component. The data is represented in an SGML based language (similar to SSML) called MVML (MultiVox Markup Language). The data expressed in the MVML form could have references to other data buffers. Data could be viewed by a component as data to process or as command sequences. In command sequences it is obvious that it has to refer to other data to process. This makes it possible to construct complex scenarios for dialogue systems.

The referred data can be tagged text (e.g. with tagging described in [5]), analyzed text with additional information about its contents, sound codes with sound level control codes, sound codes with physical parameter codes (it is called intonation matrix in [5]), speech data without intonation (concatenated elements of acoustic units e.g. diphones), speech data with additional information (sound properties - voiced-unvoiced, sound boundary information etc.). The repository of data types is extendable to other kind of data.

In the development system each data type has an assigned viewer/editor. The development system is responsible to ask the synthesizer for sending the requested data to the development system and to give it to the proper viewer/editor. The reverse direction (i.e. downloading data into the synthesizer) operates in a similar way.

The development system presents the related information in parallel windows. This is achieved by using so called data renderers to describe how data objects are connected to each other. Data renderers are responsible for presenting the data in the most appropriate form to the user. If the user wishes, it can provide him or her the data in different forms. E.g. the F0 contour could be presented in graphical way and/or tabular way, where the numbers represent the pitch variation and the columns are the places where the pitch variation begins and ends.

It is an important feature of the system that it provides a mechanism to handle the (logical and time based) connections between the displayed data representations. The editors and the connectivity of objects (e.g. orthographic, sound code, intonation submatrix, wave data portion representations of words) give help to the user to inspect and manipulate data objects in place. Connectivity means, if the user marks a portion of one representation of the speech data (e.g. sound codes), this is reflected in other representations. This is a key issue to achieve faster research - experiment - evaluation cycle. This is supported by the real-time synthesizer built into the system.

2.2 DATABASE DEVELOPMENT

To give a well suited technology for the creation of synthesizers for different languages, the system is organized to handle language dependent parts in a unified way using a defined method to handle the differences. For one hand it means that the passive components of the system are modelled by one reference model. On the other hand the model generalizes the meaning of the element of a database, thus giving opportunity to make the database customizable. This gives the keeps the development system simple and gives aid at the same time for different development purposes. The passive components are handled as they have „regular parts" and exceptions. At the grammatical level this means to have common rules and exceptions having their original meaning.

At the acoustic database level this means, that a database can have diphones or triphones as „regular" elements, they can be accessed using by rules, and should have been in the system during normal operation. Exceptions are other elements in the same data representation (e.g. waveform data with proper tagging to help further processing) the database could have. This property is granted by the generalized and flexible database architecture used in MultiVox.

This kind of flexibility of the system enables either to use diphones or even triphones for synthesis and/or to use phrases, sentences or larger texts and applying intonation curves on them using a synthesizer. This latter technology (canned messages with synthetic speech [6]) is becoming more popular in dialogue systems where the quality of the system is a key issue. With this technology diphone based synthesis can coexists with concatenation technology.

The database development tool gives three services to the user.

At the first stage of database development, the researcher has to create the first version of the database from spoken corpora. This procedure is to be semi automatised, where the segmentation of the elements will be done by manually inputting the textual representation of the spoken text. The researcher has to have the raw waveform data to be marked, to extract useful infromation from the spoken text (to have information about where a sound begins and ends, whether a sound is voiced, where are the beginnings of periods in a voiced sound, etc.). This tool helps the database developer to extract the elements (here diphones and triphones) from these marked speech waveforms. The development system provides additional services to correct the waveforms. Automatic amplitude equalization, manual manipulation capabilities are supported as well. These are introduced below.

For those data, that have any common property which can be adjusted by an appropriate process, the development system gives the opportunity to make this adjustment automatically. This automatic adjustment currently is under testing for amplitude adjustment for one speaker and one language. This part of the system is flexible enough to accept modules implementing different algorithms for adjustment. Adjustment should be applied for F0 and for the length of the element. These parameter equalisations in the database should be applied for new databases too, and also for other speakers and other languages. This could ensure, that the generated speech of the speech synthesizer sounds at approximately the same quality. This means that if we want to have several databases with approximately the same quality - with the same loudness, pitch - (parameters included F0, average energy, etc.) generated from different spoken corpora we have to adjust them regarding the parameters mentioned above. The functions, implementing the different algorithms use predefined initializaton files to enable manual adjustment on parameters. These tools are also very useful to get information about the database (e.g. average energy, F0, etc.).

After the automatically created database a fine tuning process has to be done using human knowledge and experience. This knowledge can be put into reality by using the data manipulation tools. These tools enable the developer to manually modify each element. Modification means shortening and lengthening an element, pasting some parts to it, adjusting amplitude, etc. During resynthesis the developer can easily check whether the modification yields a better result or not and what other elements are to be adjusted next.

This technological process helps the developer to get acquainted with the database creation after a short training (no deep phonetic knowledge is needed).

Because of the fact that nowadays most dialogue systems use speech technology, it is reasonable to support that task.

3 CLIENT/SERVER ARCHITECTURE

As it was mentioned in the introduction, the system uses a client/server architecture. We have chosen this architecture because we wanted to make the development system flexible, modular and to shorten phase between the research and application.

Figure 1 shows the MVoxDev-MultiVox client/server model. The application is the client and the synthesizer is the server. In the development system, the software bounding the development tools also acts as a client. It has the advantage that the development system can be used as a kind of debugger (although this is not its main role).

Communication between the client and the server is carried out in three ways: DDE communication between the client and the server as the base communication mode. This is for getting in connection with the server, because the rest is carried out at a higher level. The higher layer is implemented above that. It implements an SGML based markup language, called MVML. Data written using this language is interpreted by the server. E.g. the client can ask the server to construct a synthesizer using the letter-to-sound converter, a waveform based synthesizer, a specified database and to name it SYNTHESIZER1, then use it for a marked up text. An example of data sent to the synthesizer using MVML:

```
<MVML>
<COMMAND>
<ALLOCATE SYNTHESIZER=SYNTHESIZER1>
<PARAMETER LANGUAGE=GERMAN>
<INSERT ELEM=LETTER2SOUND_CONVERTER>
<PARAMETER STYLE=NORMAL>
<INSERT ELEM=WAVEFORM_SYNTH>
<PARAMETER SPEAKER=MALE2>
</COMMAND>
<USE SYNTHESIZER=SYNTHESIZER1>
<TG TYPE=5>
Mehr dazu finden Sie
<FOCUS>
LINKS im blauen fenster
<TG TYPE=.>
</TG>
</MVML>
```

Besides these two ways of communication the system uses callback functions to exploit real-time capabilities of the host operating system and thus provide fast feedback for the developer.

4 CONCLUSION

The baseline of the MVoxDev development system described above is currently running on the Windows platform, insertion of functions, modules and the transportation to UNIX are underway. The services and tools provided by the system to the researcher or developer can shorten the process of development by giving many informative data in a well organized way to him or her. The system is extendable and this further broadens this previously mentioned property.

The architecture of the development system fulfiles the needs of the researcher by providing useful tools for linguistic research, of the database developer by giving automatic database inspection tools and adjustment services and finally of the application developer, who wants to build an application using and perhaps customizing the synthesizer.

REFERENCES:

[1] Hugo C. van Leeuwen - Enrico te Lindert: Speech Maker: A flexible frameworkfor constructing text-to-speech systems. In Vincent J. van Heuven and Louis C. W. Pols (editors): Analysis and Synthesis of Speech, pp. 317-338, Mouton de Gruyter, 1993.
[2] Alissali, M., Bailly, G.: COMPOST: A Client-Server Model For Applications Using Text-to-Speech Systems. Proceedings of Eurospeech '93, Vol. 3 pp. 2095-2098, Berlin, Sept. 1993.
[3] Black, Alan W., Taylor, Paul: The Festival Speech Synthesis System, System documentation, 1997. http://www.cstr.ed.ac.uk/projects/festival.html
[4] Isard, Amy: SSML: A Markup Language for Speech Synthesis, thesis, 1995 http://www.sil.org/sgml/gen-apps.html#ssml
[5] Gábor Olaszy, Géza Németh,: Prosody generation for German CTS/TTS systems (from theoretical intonation patterns to practical realisation). Speech Communication, pp. 37-60, Issue 21, 1997.
[6] Pearson, S., Holm F., Hata, K.: Combining Concatenation and Formant Synthesis for Improved Intelligibility and Naturalness in Text-to-Speech Systems. International Journal of Speech Technology, pp. 103-107, Vol. 1, No. 2, 1997.