ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

CONCATENATIVE SPEECH SYNTHESIS FOR AMHARIC USING UNIT SELECTION METHOD

By: Eyob Bayou

A Project Report Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

June, 2011

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

CONCATENATIVE SPEECH SYNTHESIS FOR AMHARIC USING UNIT SELECTION METHOD

By

Eyob Bayou

APPROVED BY

EXAMINING BOARD:

1. Dr. Yaregal Assabie, Advisor _____

2. _____ _____

3. _____ _____

ACKNOWLEDGEMENT

I would like to use this opportunity to thank many people who were very helpful throughout the development of this project.

First, I would like to thank Dr. Yaregal Assabie for his support, thoughtful ideas and guidance that made the completion of this work possible. I also like to take the chance to say thank you to Nirayo Hailu and Abraham Wubie for sharing me their materials. My friends and graduate students in Computer Science Department of Addis Ababa University also deserve the mention for participating in the MOS evaluation.

Finally, I shall also acknowledge the continuous emotional and spiritual support I have enjoyed from my family and my beloved wife.

Table of Contents

List of Tables

List of Figures

ABBREVIATIONS

DSP	Digital Signal Processing
MOS	Mean Opinion Score
TTS	Text To Speech
NLP	Natural Language Processing
MIT	Massachusetts Institute of Technology
HMM	Hidden Markov Model
YNQ	Yes/No Questions
WHQ	WH Questions
UML	Unified Modeling Language
VODER	Voice Operating Demonstrator
DAVO	Dynamic Analog of the Vocal tract
POS	Part Of Speech
LPC	Linear Predictive Coding

ABSTRACT

Speech synthesis takes text as input and generates acoustic signal as output. In the process, the input text is preprocessed to tokenize it into words or other meaningful tokens and to transliterate numbers, abbreviations and acronyms. Text-analysis follows text preprocessing to identify grammatical structures and context. Once the text analysis phase is completed the next step is to convert graphical representation of sounds to their phonetic representation. A phoneme usually has multiple phones that are used in different contexts.

Amharic language's orthography is phonemical in the sense that a grapheme represents exactly one phoneme. However, this statement is true as long as epenthesis and geminations are not considered. The language's orthography does not also show suprasegmental information that is required to properly model speaking styles. Even though converting grapheme to phoneme is easy in Amharic, converting phoneme to phone is very difficult because of the two necessary and yet orthographically unrepresented components of the language – epenthesis and gemination. Modeling prosodic features of various speaking styles is also the other challenging task in developing Amharic TTS. This is challenging because, in one hand, the task of modeling human speech is very challenging in itself and in the other hand, research works done for Amharic language are relatively few.

This project work has tried to address epenthesis and gemination, which are phonologically very important features of the language, by studying and implementing techniques found in various literatures. Making use of orthographic property of verbs in their perfect form, this work introduces rules that can be used to locate phones that need to be stressed. The grapheme to phoneme conversion algorithm also addresses epenthesis. Prosodic differences of declarative and interrogative utterances are represented by making use of unique sentence-final phones recorded and segmented for this purpose. Transliteration of numerals and abbreviations is also addressed in the text preprocessing phase of the system.

The results found after being evaluated by ten fluent speakers of the language are encouraging.

1. Introduction

Technology has enabled human beings to do many tasks and it is facilitating their day-to-day activities. Technologies significantly affect human beings' ability to control and adapt to their environments. It increases productivity and efficiency. With the advent of electronic devices in general and computers in particular, technology has become so pervasive that it is being used in many areas of human activity. Computers, nowadays, are the major actors in manufacturing, entertainment, communication, education, health, commerce, government, etc.

Artificial intelligence is one of the youngest study areas of computer science. Natural language processing as sub field of artificial intelligence deals with understanding and processing human languages. Today there are many systems that are able to read and pronounce written languages such as English, Spanish, French, Japanese and other widely used languages around the world. Such systems are called Text-to-Speech synthesis systems. One of the basic features of technology is to enable individuals perform tasks that are difficult to perform without it; and computer as part of technology is about making things readily available with less adjustment from the human side. With this in mind, the contribution that NLP in general and Text-to-Speech synthesis in particular makes is unavoidably significant. TTS enables people and especially physically impaired people to interact with computers with little adjustment form their side. Today, there are many people in and outside Ethiopia who use computers to process text written in Geez (Amharic) alphabets. As the number of such users increase, the demand for Amharic TTS systems also increases. Amharic, being a Semitic language that has many speakers next to Arabic, is worthy of such attention.

1.1 Statement of the Problem

Amharic language, being the official language of Ethiopia, is used by much of Ethiopian population. There are lots of organizations, institutions and offices using the language as their working language. There are many books – fiction and nonfiction written in the language and of course, Amharic is a Semitic language that has the largest number of speakers after Arabic. However, compared to its use, it will not be exaggerating if it is said that it is one of the least supported and least researched languages in the area of Natural Language Processing. Particularly, researches conducted on language technologies like speech synthesis and the application of such technologies are very limited or unavailable [3]. Amharic is one of the most appropriate languages for concatenative synthesis approach because of its phonetic nature. However Amharic TTS synthesizers are not that available on the market for potential users of computers and users with physical impairments. Even though following concatenative synthesis approach eases the process of developing Amharic speech synthesizer that renders better naturalness, the intelligibility of the sound produced is dependent on many factors such as intonations, stresses and punctuations. Converting numbers and abbreviations to speech is also another factor that exacerbates Amharic speech synthesizer development [2]. The project tries to develop Amharic synthesizer that addresses naturalness and intelligibility by following unit selection concatenative synthesis approach which makes use of multiple recorded phones for addressing context and sentence structure prosody.

1.2 Objective of the Project

General Objective

The general objective of the project is to design and develop Amharic TTS system that produces sounds that render naturalness and intelligibility by using word structure/context and punctuation marks as intonation information.

Specific Objectives

The project will address intonations that vary according to punctuation and phoneme location. The project tries to achieve this objective by:

- Studying Amharic orthography, phonology and morphology
- Reviewing and adopting techniques explored in various literatures
- Identifying and employing rules that control epenthesis and gemination[1]
- Studying and applying intonation variations reflected in declarative and Yes/No interrogative utterances
- Developing number and abbreviation transliterating schemes

1.3 Scope of the Project

The TTS system developed is a generic system that tries to pronounce virtually any Amharic Text. In doing so, it makes consideration of epenthesis, gemination and sentence structure modeling. The task on gemination is focused on verbs in their perfect form. Known rules on epenthesis are covered excluding rules that dictate epenthesis in the presence of geminates. Disambiguation of heteronyms and symbols or disambiguation that requires the use of lexica or morphological analyzers is out of the scope of this project because of the time allotted for this

[1] Descriptions of epenthesis and gemination are given in Section 3.2.1 and 3.2.2

project. Symbols that serve only the purpose of signaling pauses and abbreviations will be considered in the project. Intonation information other than those which can be obtained from the kind of punctuation used at the end of the sentence and the location of phonemes are not considered in this project. The two punctuation marks to be used for modeling appropriate sentence prosody are “?” and “::”.

1.4 Contribution of the Project

In this project issues like epenthesis, gemination, declarative versus interrogative prosody modeling are covered. Various rules and principles explored in many different research works are considered for addressing these issues. The labeling schemes and algorithms developed for locating target sound units address one of the major difficulties of the unit selection approach, which is selecting the right unit form multiple sound units. Sound units are recorded and segmented in such a way that the same phoneme’s allophones are captured. Allophones are made available for properly representing epenthesis, gemination and sentence level prosody. Text-preprocessing rules to properly tokenize text, transliterate numerals and abbreviations using Amharic punctuation marks are also included in the development. Grapheme-to-phoneme conversion algorithms are also developed to locate phones that require stress and consonant clusters that require the insertion of the epenthetic vowel. In an effort to make the system easy to use, a plug-in tool for Ms-Word 2007 is also developed.

1.5 Significance of the Project

The system from this project can be used in communication and broadcasting organizations such as Ethio Telecom and Ethiopian Radio and Television Agency. In addition to such organizations, individuals with physical (vocal or visual) impairments can use it to simplify their interaction with computers. It helps any person by reading electronic documents without requiring his/her

visual attention/function. The other application of the software can be its use in preparing audio books which are becoming very popular in this multimedia generation who prefers listening to reading.

1.6 Methods and Tools

The project made use of concatenative synthesis by unit selection to yield naturalness and intelligibility which are evaluation criteria of the quality and accuracy of a speech synthesizer system [3, 4]. A rule-based (not a dictionary) approach is employed to produce the phonetic representation from the given text. Prerecorded sound samples are used in converting phonetic transcriptions into sounds. Multiple recorded phones (allophones) are made available to better represent words that differ in pronunciation when placed in sentences with different punctuation.

Tools used:

- Microsoft's Visual Studio 2010 with C# is used for interface design and coding
- jetAudio™, Audacity™, Praat™, and Wavesurfer™ are used for recording, manipulating and analyzing the samples

1.7 Organization of the Report

The document is organized into six chapters. Chapter One deals with introduction of the project. Literature and related works on Amharic language are reviewed in Chapter Two. Chapter Three gives some explanation of the language's orthographical and phonological analysis. Chapter Four which deals with the requirement specification and design of the system, is followed by Chapter Five that discusses the implementation issues and procedures. Challenges, conclusion, and recommendations are given in Chapter Six of this report.

2. Literature Review

The size and versatility of the help human beings are enjoying from the use of machines in their daily activities is enormous. The use of machines ranging from manual to automatic, mechanical to electronic, handheld to room-filling have assisted people to exercise problem solving effectively and efficiently. The next generation of machines that human beings are so ambitious about is intelligent machines. An important function of these machines is that they master speech input and output to communicate with human beings naturally using natural languages. They understand human language and generate human like language [1]. Speech technology as one part of natural language processing (NLP) is the field that aims at the development of technologies that allow human beings to use natural languages in order to communicate with computers and other devices. Some of the most important applications of NLP technology are speech recognition (converting speech into text), text-to-speech synthesis (converting text to speech), and speech/text understanding (maps words into action and plans system-initiated actions) [1].

2.1 Text-to-speech synthesis

Speech synthesis is the ability of a computer to produce spoken words. Computer speech can be produced either by joining prerecorded words together or by having the computer produce the sounds that make up spoken words. As described in [5], there are several levels in the whole process as shown in *Figure 2.1*. Generally, a first step, called ***text pre-processing***, transforms numbers, abbreviations, acronyms, etc. into orthographic segments. Capital letters, hyphenations, etc., are also processed at this level. Then, ***text analysis*** aims at determining the grammatical characteristics of the words and the syntactic structure of the sentence which are necessary for both the phonemic conversion (disambiguation of heteronyms, homographs, liaisons, lexical

stress, etc.) and the prosodic modeling. The ***grapheme-to-phoneme*** conversion transforms the text into a string of phonemes. Today, most of the converters make wide use of dictionaries with orthographic inputs and phonetic outputs. In most cases, it is somewhat difficult to separate text analysis and phonemic conversion, since huge word dictionaries may give simultaneously the phonemic conversion, the grammatical category, the stress position, and so forth. ***Prosodic modeling*** assigns a duration and melodic contours to the sentence and to its constituting syllables. The applied prosodic patterns depend on observations made in a language on the one hand, and on a symbolic annotation of the syntactic structure to be synthesized on the other hand. The duration and the melodic contours of a segment can be calculated by rules. Finally, ***sound generation***; it is the process that transforms a string of symbols into an acoustic output. Rule-based (formant) synthesis assigns target values to the formants of each phonetic unit and then smoothes the transitions in-between, depending on units of duration. A widely used alternative makes use of coded speech segments pre-stored in dictionaries. Elementary speech segments can be syllables, diphones, polyphones, etc., or a mixture of them. Those units can be simply digitized, or coded in terms of formant frequencies, articulatory parameters, etc. If the elementary units are simply digitized, signal processing techniques are first applied to the speech units so that fundamental frequency, duration, and intensity match those given by the prosodic model. If the dictionary is made of coded speech segments, prosodic transformation is applied to the parameters before they are concatenated, and then decoded into a speech signal.

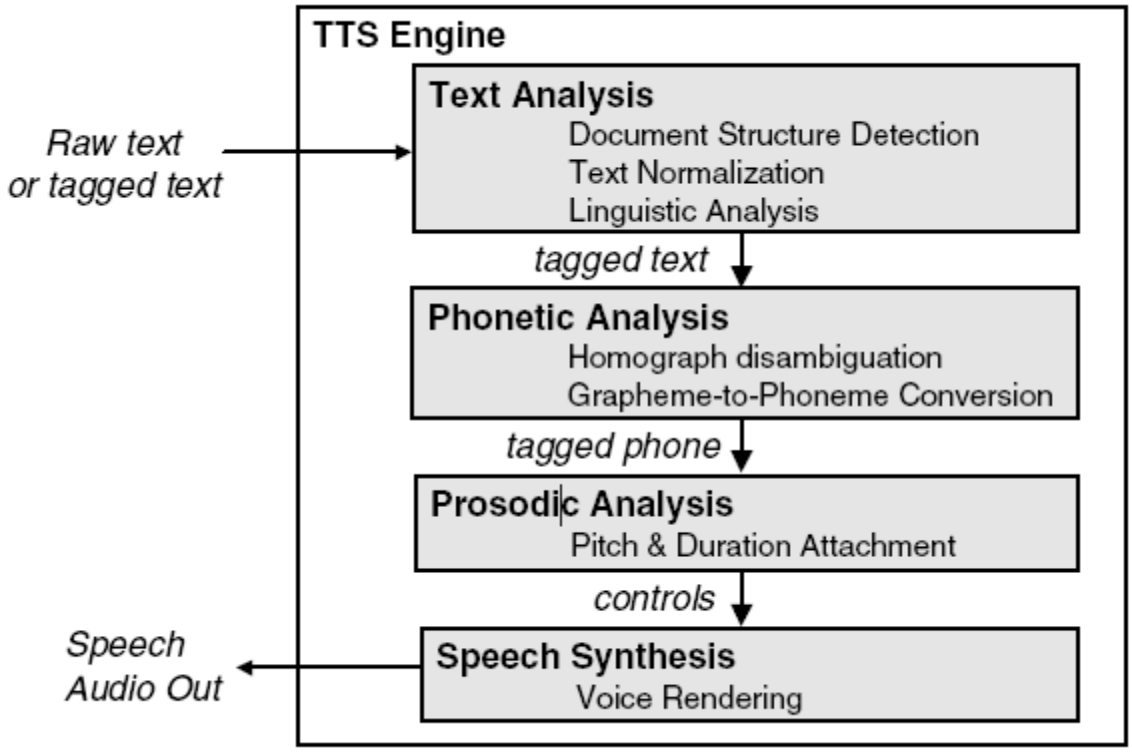


Figure 2.1 Block diagram of a general text-to-speech system [9].

2.1.1 Evaluation of TTS Systems

Synthetic speech can be compared and evaluated with respect to intelligibility, naturalness, and suitability [8]. High quality synthesis of different spoken languages has been a challenge to TTS developers and engineers. This has become more important where the task is to develop unlimited text to speech synthesis system. Several techniques based on simulation of articulatory models, electrical analogues (formant synthesizers) and concatenation of natural speech waveform samples from a large corpus have been developed and used with varying success. However, since the success of TTS synthesizers are largely measured by *naturalness* and

intelligibility, extensive research is still going on to produce natural sounding intelligible synthetic speech with desired accents, mood, situation and quality [6].

2.1.2 Intonation Generation

Intonation generation is a critical part of any Text to Speech Synthesis system. Intonation is important for intelligibility and critically important for naturalness. But it has also been historically one of the less well developed components of most TTS systems. Functionally intonation generation is often divided into two parts [7]. The first is locating and characterizing accents. This may be considered as primarily a linguistic problem since it depends in large part on the syntax and semantics of the sentence to be synthesized. The second functional part is assigning suitable parameters for the realization of those accents given their specifications.

2.2 History and Development of Speech Synthesis

Artificial speech has been a dream of the humankind for centuries. In this topic, the history of synthesized speech from the first mechanical efforts to systems that form the basis for today's high-quality synthesizers is highlighted. Sami Lemmetty's [8] paper is followed to categorically present the developments.

2.2.1 From Mechanical to Electrical Synthesis

The earliest efforts to produce synthetic speech were made over two hundred years ago. In St. Petersburg 1779 Russian Professor Christian Kratzenstein explained physiological differences between five long vowels (/a/, /e/, /i/, /o/, and /u/) and made apparatus to produce them artificially. A few years later in Vienna in 1791, Wolfgang von Kempelen introduced his "Acoustic-Mechanical Speech Machine", which was able to produce single sounds and some sound combinations. In about mid 1800's Charles Wheatstone constructed his famous version of

von Kempelen's speaking machine. It was a bit more complicated and was capable to produce vowels and most of the consonant sounds. Some sound combinations and even full words were also possible to produce. In late 1800's Alexander Graham Bell with his father, inspired by Wheatstone's speaking machine, constructed same kind of speaking machine. The research and experiments with mechanical and semi-electrical analogs of vocal system were made until 1960's, but with no remarkable success [8].

2.2.2 Development of Electrical Synthesizers

The first full electrical synthesis device was introduced by Stewart in 1922. The synthesizer had a buzzer as excitation and two resonant circuits to model the acoustic resonances of the vocal tract. The machine was able to generate single static vowel sounds with two lowest formants, but not any consonants or connected utterances. The first device to be considered as a speech synthesizer was VODER (Voice Operating Demonstrator) introduced by Homer Dudley in New York World's Fair 1939. VODER was inspired by VOCODER (Voice Coder) developed at Bell Laboratories in the mid-thirties. The original VOCODER was a device for analyzing speech into slowly varying acoustic parameters that could then drive a synthesizer to reconstruct the approximation of the original speech signal. After demonstration of VODER the scientific world became more and more interested in speech synthesis. It was finally shown that intelligible speech can be produced artificially. Actually, the basic structure and idea of VODER is very similar to present systems which are based on source-filter model of speech.

In 1951, Franklin Cooper and his associates developed a Pattern Playback synthesizer at the Haskins Laboratories. It reconverted recorded spectrogram patterns into sounds, either in original or modified form. The spectrogram patterns were recorded optically on the transparent belt.

The first articulatory synthesizer was introduced in 1958 by George Rosen at the Massachusetts Institute of Technology, M.I.T. The DAVO (Dynamic Analog of the VOcal tract) was controlled by tape recording of control signals created by hand.

The first full text-to-speech system for English was developed in the Electrotehnical Laboratory, Japan 1968 by Noriko Umeda and his companions; it was based on an articulatory model and included a syntactic analysis module with sophisticated heuristics.

In 1979 Allen, Hunnicutt, and Klatt demonstrated the MITalk text-to-speech system developed at M.I.T. Two years later Dennis Klatt introduced his famous Klattalk system, which used a new sophisticated voicing source.

Modern speech synthesis technologies involve quite complicated and sophisticated methods and algorithms.

2.3 Speech Synthesis Methods and Techniques

Synthesized speech can be produced by several different methods, which have their own advantages and disadvantages. The methods are usually classified into three groups [8]:

- Articulatory synthesis – attempts to model the human speech production system.
- Formant synthesis – models the pole frequencies of speech signal or transfer function of vocal tract based on source-filter-model.
- Concatenative synthesis – uses different length prerecorded samples derived from natural speech.

2.3.1 Articulatory Synthesis

Articulatory synthesis consists on a class of techniques for synthesizing speech based on physical models of the human vocal tract and the articulation processing occurring there [40]. It is also one of the most difficult methods to implement and the computational load is also considerably higher than that of other common methods.

Articulatory synthesis typically involves models of the human articulators and vocal cords. The articulators are usually modeled with a set of area functions between glottis and mouth. For rule-based synthesis the articulatory control parameters may be for example lip aperture, lip protrusion, tongue tip height, tongue tip position, tongue height, and tongue position. Prosody or excitation parameters may be glottal aperture, cord tension, and lung pressure [8].

When speaking, the vocal tract muscles cause articulators to move and change shape of the vocal tract which causes different sounds. The data for articulatory model is usually derived from X-ray analysis of natural speech. However, this data is usually only 2-D, when the real vocal tract is naturally 3-D, so the rule-based articulatory synthesis is very difficult to optimize due to the unavailability of sufficient data of the motions of the articulators during speech. Also, the movements of tongue are so complicated that it is almost impossible to model them precisely.

Advantages of articulatory synthesis are that the vocal tract models allow accurate modeling of transients due to abrupt area changes.

2.3.2 Formant Synthesis

Probably the most widely used synthesis method during the last few decades has been formant synthesis which is based on the source-filter-model of speech. There are two basic structures in general, parallel and cascade, but for better performance some kind of combination of these is

usually used. Formant synthesis also provides infinite number of sounds which makes it more flexible than for example concatenation methods.

At least three formants are generally required to produce intelligible speech and up to five formants to produce high quality speech. Each formant is usually modeled with a two-pole resonator which enables both the formant frequency and its bandwidth to be specified. Rule-based formant synthesis is based on a set of rules used to determine the parameters necessary to synthesize a desired utterance using a formant synthesizer. The input parameters may be for example the following, where the open quotient means the ratio of the open-glottis time to the total period duration: [8]

- Voicing fundamental frequency (F0)
- Voiced excitation open quotient (OQ)
- Degree of voicing in excitation (VO)
- Formant frequencies and amplitudes (F1...F3 and A1...A3)
- Frequency of an additional low-frequency resonator (FN)
- Intensity of low- and high-frequency region (ALF, AHF)

2.3.3 Concatenative Synthesis

Connecting prerecorded natural utterances is probably the easiest way to produce intelligible and natural sounding synthetic speech. However, concatenative synthesizers are usually limited to one speaker and one voice and usually require more memory capacity than other methods. One of the most important aspects in concatenative synthesis is to find correct unit length. The selection is usually a trade-off between longer and shorter units. With longer units, high naturalness, less concatenation points and good control of coarticulation are achieved, but the

amount of required units and memory is increased. With shorter units, less memory is needed, but the sample collecting and labeling procedures become more difficult and complex. In present systems units used are usually words, syllables, demisyllables, phonemes, diphones, and sometimes even triphones [8].

Word is perhaps the most natural unit for written text and some messaging systems with very limited vocabulary. Concatenation of words is relatively easy to perform and coarticulation effects within a word are captured in the stored units. However, there is a great difference with words spoken in isolation and in continuous sentence which makes the continuous speech to sound very unnatural. Because there are hundreds of thousands of different words and proper names in each language, word is not a suitable unit for any kind of *unrestricted* TTS system [8].

The number of different syllables in each language is considerably smaller than the number of words, but the size of unit database is usually still too large for TTS systems. For example, there are about 10,000 syllables in English. At the moment, no word or syllable based full TTS system exists. The current synthesis systems are mostly based on using phonemes, diphones, demisyllables or some kind of combinations of these. Demisyllables represents the initial and final parts of syllables. One advantage of demisyllables is that, for example, only about 1,000 of them are needed to construct the 10,000 syllables of English. Using demisyllables, instead of for example phonemes and diphones, requires considerably less concatenation points. Demisyllables also take account of most transitions and then also a large number of coarticulation effects and also covers a large number of allophonic variations due to separation of initial and final consonant clusters. However, the memory requirements are still quite high, but tolerable. Compared to phonemes and diphones, the exact number of demisyllables in a language cannot be

defined. With purely demisyllable based system, all possible words cannot be synthesized properly. This problem is faced at least with some proper names.

Phonemes are probably the most commonly used units in speech synthesis because they are the normal linguistic presentation of speech. The inventory of basic units is usually between 40 and 50, which is clearly the smallest compared to other units. Using phonemes gives maximum flexibility with the rule-based systems. However, some phones that do not have a steady-state target position, such as plosives, are difficult to synthesize. The articulation must also be formulated as rules. Phonemes are sometimes used as an input for speech synthesizer to drive for example diphone based synthesizer.

Diphones (or dyads) are defined to extend the central point of the steady state part of the phone to the central point of the following one, so they contain the transitions between adjacent phones. That means that the concatenation point will be in the most steady state region of the signal, which reduces the distortion from concatenation points. Another advantage with diphones is that the coarticulation effect needs no more to be formulated as rules. In principle, the number of diphones is the square of the number of phonemes (plus allophones), but not all combinations of phonemes are needed [8].

Longer segmental units, such as triphones or tetraphones, are quite rarely used. Triphones are like diphones, but contains one phoneme between steady-state points (half phoneme - phoneme - half phoneme). In other words, a triphone is a phoneme with a specific left and right context.

Building the unit inventory consists of three main phases [8]. First, the natural speech must be recorded so that all used units (phonemes) within all possible contexts (allophones) are included.

After this, the units must be labeled or segmented from spoken speech data, and finally, the most appropriate units must be chosen.

Gathering the samples from natural speech is usually very time-consuming. However, some of this work may be done automatically by choosing the input text for analysis phase properly. The implementation of rules to select correct samples for concatenation must also be done very carefully.

There are several problems in concatenative synthesis compared to other methods.

- Distortion from discontinuities in concatenation points, which can be reduced using diphones or some special methods for smoothing signal.
- Memory requirements are usually very high, especially when long concatenation units are used, such as syllables or words.
- Data collecting and labeling of speech samples is usually time-consuming.

Unit selection synthesis techniques were introduced to alleviate problems encountered with older diphone concatenation techniques which made use of only one version of any particular diphone. Diphone synthesis tended to sound unnatural. Much effort in diphone concatenation synthesis was spent on unit selection. Units were selected for their ability to join well to neighboring units on average. This dissatisfaction, coupled with an increasing technological ability to build more memory and CPU-intensive speech synthesis systems motivated developers and researchers to pursue general unit selection approach deeper [12].

Unit selection based synthesizers choose suitable fragments from a database of speech recorded from a speaker and join them together with minimal signal modifications. Unit selection based synthesizers using minimal modification of the speech signal produce highly intelligible and

natural sounding utterances instead of buzzy or robotic sounding speech. Minimal modification in unit selection based synthesis does not only bring high synthesis quality, but also causes some problems. These problems are most of the time related to the fact that unit selection systems use minimal signal modification.

Often problems are caused by the discrepancy between phones asked for by a TTS front-end and phones selected from a labeled voice database [13]. Speech databases are usually labeled with phonemic symbols rather than phonetic ones. However, the same phoneme can be realized in different forms (*allophones*) depending on certain phone contexts. The phoneme /t/ in American English, for example, generates several allophones. There are two possible approaches to alleviate this problem [14]:

(1) specify greater allophonic (multiple recorded phones) detail in TTS front-end and database labels, or (2) identify contexts, such as pre-vocalic / post-vocalic positions within a syllable, that determine, in part, the allophonic variations. In the work specified on [15], such discrepancies were reduced by introducing allophones (multiple phones) in the phone set. The second alternative was also shown to be appealing on [14]. In the work, a new phone labeling method that creates better matches with phone realization in speech is shown to be effective. The new phone set includes the distinction of consonant variants dependent on their position in the syllable structure, pre-vocalic and post-vocalic, which reduces missing consonants and consonant confusion.

2.3.4 Other Methods

HMMs have been applied to speech recognition from late 1970's. For speech synthesis systems it has been used for about two decades. A Hidden Markov Model is a collection of states connected

by transitions with two sets of probabilities in each: a transition probability which provides the probability for taking this transition, and an output probability density function which defines the conditional probability of emitting each output symbol from a finite alphabet, given that the transition is taken [8][2].

Neural networks have been applied in speech synthesis for about ten years and the latest results have been quite promising. However, the potential of using neural networks have not been sufficiently explored. Like hidden markov models, neural networks are also used successfully in speech recognition.

2.4 Reviews on Popular TTS Frameworks and Platforms

In this section a concise review is given on some of the most popular text-to-speech synthesizer frameworks.

Edinburgh Speech Tools

The Edinburgh Speech Tools Library is library of general speech software, written at the Centre for Speech Technology Research at the University of Edinburgh. It is written in C++ and provides a range of tools for common tasks found in speech processing. The library provides a set of standalone executable programs and a set of library calls which can be linked into user programs. The Edinburgh Speech Tools Library has two main parts: a software library and a set of programs which use the library [16]. The Library contains:

- *Speech class:* includes tracks for storing sets of time aligned coefficients, and waves for digitally sampled speech waveforms.

[2] Sami Lemmetty [8] is cited frequently in this section because his work deals with review of speech synthesis technologies extensively

- *Linguistic class:* a comprehensive system for storing different kinds of linguistic information is given.
- *Audio playback:* easy to use routines to record and play audio data.
- *Signal processing:* commonly used signal processing algorithms such as including pitch tracking, cepstra and LPC, filtering, Fourier analysis etc.
- *Statistical functions*
- *Grammars*
- *Intonation:* software support for the Tilt intonation model
- *Speech Recognition*
- *Utility Functions and Classes:* useful classes such as lists, vectors, matrices, strings and functions for reading files, parsing command lines etc.

Festival TTS system

The Festival TTS system was developed in CSTR at the University of Edinburgh by Alan Black and Paul Taylor and in co-operation with CHATR, Japan. The current system is available for American and British English, Spanish, and Welsh. The system is written in C++. As a University program the system is available free for educational, research, and individual use. The system is developed for three different aspects. For those who want simply use the system from arbitrary text-to-speech, for people who are developing language systems and wish to include synthesis output, such as different voices, specific phrasing, dialog types and so on, and for those who are developing and testing new synthesis methods [17].

Flite

Flite was primarily developed to address one of the most common complaints about the Festival Speech Synthesis System. Festival is large and slow. Although much work was done to ensure Festival can be trimmed and run fast it still requires substantial resources per utterance to run. After some investigation to see if Festival itself could be trimmed down it became clear because there was a core set of functions that were sufficient for synthesis that a new implementation containing only those aspects that were necessary would be easier than trimming down Festival itself.

Flite is not intended as a research and development platform for speech synthesis, Festival is and will continue to be the platform for that. Flite however is designed as a run-time engine when an application needs to be delivered. It specifically addresses two communities. First as an engine for small devices such as PDAs and telephones where the memory and CPU power are limited and in some cases do not even have a conventional operating system. The second community is for those running synthesis servers for many clients. The Flite distribution consists of two distinct parts [18]:

- The Flite library containing the core synthesis code
- Voice(s) for flite. These contain three sub-parts
    - Language models: text processing, prosody models etc.
    - Lexicon and letter to sound rules
    - Unit database and voice definition

Java Speech API

Two core speech technologies are supported through the Java Speech API: *speech recognition* and *speech synthesis*. JSAPI is an extension to the Java platform. Extensions are packages of classes written in the Java programming language (and any associated native code) that application developers can use to extend the functionality of the core part of the Java platform.

The design goals for the Java Speech API included [19]:

- Provide support for speech synthesizers and for both command-and-control and dictation speech recognizers.
- Provide a cross-platform, cross-vendor interface to speech synthesis and speech recognition.
- Enable access to speech technology.
- Support integration with other capabilities of the Java platform, including the suite of Java Media APIs.

Microsoft Speech API

Microsoft Speech API (SAPI) is basically a managed API that allows developers to write speech enabled applications in .NET Framework 3.0 and above and is already integrated with Windows Vista and Windows 7. The API resides in System.Speech.Dll assembly [20]. The SAPI API provides a high-level interface between an application and speech engines. SAPI implements all the low-level details needed to control and manage the real-time operations of various speech engines. SAPI communicates with applications by sending events using standard callback mechanisms (Window Message, callback proc or Win32 Event) [21].

2.5 Review of Amharic TTS

Some conspicuous works on the development of TTS for Amharic are presented in the paragraphs that follow.

Few projects and theses are conducted in order to introduce tools and methods for developing Amharic text-to-speech synthesizers. Sebsibe *et al* [4] has shown how to use the Festvox framework for building unit selection voices for Amharic language. In their work they defined a transliteration scheme to work with Amharic scripts. By selecting the prompt-list from various sources they built a unit selection voice for Amharic by incorporating Amharic phone set, syllabification rules and letter to sound rules.

A synthesizer which follows a formant synthesis approach to generate a speech for a given Amharic input text was developed by Yibeltal Tefera [22]. The developer collected speech for voiced sounds and extracted parameters such as formants, bandwidth, pitch, etc from the collected speech. The unvoiced sounds were also stored by segmenting them from all Amharic syllables. He finally synthesized the speech by first generating the voiced sounds using the parameters from the inventory data and concatenating both the voiced and unvoiced sounds. The system works on word level.

A research on formant based speech synthesis for Amharic vowels was also conducted by Nadew Tademe [23]. He has introduced a synthesizer that produces vowels according to their context in a given word. In this work, parameters like formant bandwidth, fundamental frequency and pitch period were first extracted from any speech file and then these parameters were applied for synthesizing the artificial speech.

In the research work presented in [3], Tadesse *et al.* developed a syllabic based rule-based (formant) TTS system with prosodic control method for Amharic. The designed Amharic TTS (AmhTTS) is parametric and rule-based system that employs a Cepstral method and uses a Log Magnitude Approximation (LMA) filter. They claim that their study provides a total solution on prosodic information generation mainly by modeling the durations.

Habtamu Taye [25] also developed a TTS system for Amharic language by making use of the concatenative synthesis approach. He used syllables as basic units of concatenation. The system developed appears to be not making any consideration of prosodic/intonation features such as epenthesis and gemination.

Demeke Asres [10] used LPC-Residual technique to extract and adapt neutral, command, rough and Yes/No question prosodic features. In his work, he has shown how to transform one's speaking style to another speaking style by using the extracted features such as pitch, duration and energy for re-synthesis.

Bereket Kasay [11] had followed the Hidden Markov Model (HMM) to develop Amharic Speech Synthesizer. Parameters such as mel-cepstrum coefficients and fundamental frequencies along with festival's and festvox's utterance structure were used in training the model. Even though HMM is a good approach to model prosodic features of a language, such features were not included in the system. Some of the text normalization activities such as expansion of abbreviations are done manually.

3. Linguistic Features of Amharic Language

Amharic is a Semitic language and the official language of the government of the Federal Democratic Republic of Ethiopia (FDRE). It is the second most spoken Semitic language in the world, next to Arabic and is estimated to be spoken by over 20 million people as their first or second language [23]. Today, it is the second largest language in Ethiopia after Oromigna, a Cushitic language and possibly one of the five largest languages on the African continent. Amharic uses a unique script, which has originated from an ancient liturgical language called Ge'ez. Written Ge'ez can be traced back to at least the 4th century AD. The first versions of the language included consonants only, while the characters in later versions represent *consonant-vowel (CV) phoneme pairs* [23].

3.1 The Amharic Orthography

The Amharic orthography, as represented in the Amharic Character set - also called ፊደል (fidel)[3] consists of 276 distinct symbols. In addition, there are twenty numerals and eight punctuation marks [26]. There are thirty three (33*7=231) core orthographic symbols, each of which has seven different shapes, usually known as orders, to represent the seven vowels. Each consonant and the seven vowels in combination represent CV syllables [22]. Each of these consonant and vowel grapheme can appear independently or can form a combinant letter. Each consonant can form CV pattern except with the vowel /ix/ called epenthetic vowel [4]. There is also representation for diphthongs such as kwa (ኳ), gwa (ጓ), qwa (ቋ). In addition, there is the representation of numbers from 1 to 10 and multiples of 10 each with different symbols [22].

The basic 32 alphabets of Amharic are consonants having seven orders to show the seven vowels as shown in *Table 3.1*. Out of the seven derivatives six of them are CV (Consonant vowel)

[3] *The Latin alphabet notations used for representing Amharic text in this document can be referred in Appendix A.*

combinations while the sixth is the consonant itself [4]. Other symbols representing labialization, numerals, and punctuation marks are also available. The first characters from consonants' order of seven are called Ge'ez (ግዕዝ) that is to say first in Ge'ez. Similarly the remaining characters are named as Ka'ib (ካዕብ), Sali's (ሳልስ), Rab'i (ራብዕ), Hami's (ሀምስ), Sadi's (ሳድስ), and Sab'i (ሳብዕ) [23].

Order	1st	2nd	3r d	4th	5th	6th	7th
V / C	e	u	ii	a	ie	ix	o
/m/	መ	መ፡	ሚ	ማ	ሜ	ም	ሞ
/b/	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ

Table 3.1: The Amharic consonants with vowels (mainly adopted from [23])

In Amharic language, even though, epenthetic vowels and geminations are crucial factors in properly pronouncing a given word, they are not shown in the orthography. The next topic tries to deal with these two phonological parameters in detail.

3.2 Analysis of the Amharic Phonology

Amharic language is phonetic in nature, which means it is more or less one to one correspondence between the language's orthography and phonemic representation except for epenthetic vowels and geminations [26].

Let's further see the epenthesis and gemination issues in the language:

3.2.1 Epenthesis

Epenthesis is the process of inserting the epenthetic vowel between consonant clusters. Epenthetic vowel is a high central vowel which Amharic language uses to break up consonant

clusters. This vowel is phonetically represented as 'ɨ' or 'ə'. In Amharic, consonants (represented as the sixth order characters in the series) may or may not be followed by this vowel [27]. The language's orthography does not show the presence of the epenthetic vowel even though it plays a key role for proper pronunciation of speech and syllabification [26].

Many researchers had undertook the task of identifying rules that determine the insertion of the epenthetic vowel, Hudson [29] lists the following rules which are popular and widely accepted rules to identify allowed and disallowed consonant clusters. Here are the rules:

- No word-initial consonant sequences except C+w as in hwala "back" and Kwas "ball".
- Word-final sequence of at most two consonants. of which, typically, the sonority of the penultimate equal to or greater than that of the last, including final long consonant (thought of as CC) , for example bilt "clever", mist "wife", widd "expensive" , higg "law".
- Medial sequence of at most two consonants, including cases with long consonants counted as two consonants: alle "he is present", metta "he came" wesdo "he may take", beklo "mule".

From the above rule, it can be concluded that word initial clusters may not appear in Amharic; consonant clusters are allowed only in medial and final positions of a word as long as the cluster is of size two; and consonant clusters of size two are allowed at word final position if the sonority of the penultimate is equal to or greater than that of the last.

In case of clusters with three consonants, [29] dictates that epenthesis occurs between the second and the third consonant.

The sonority of a sound is relative loudness compared to other sounds. Speech sounds can be ranked in terms of their relative sonority: **STOPS** (VOICELESS) < **STOPS** (VOICED) <

FRICATIVES(VOICELESS) < **FRICATIVES**(VOICED) < **NASALS** < **LIQUIDS** < **GLIDES** < **SEMI-VOWEL** < **VOWELS** [28].

Mulugeta [29] summarizes the process of epenthesis in six rules and also gives further explanation on how to control epenthesis during gemination:

1. Word-initially no consonant cluster
2. If a word medial consonant cluster contains the geminate and singleton in sequence, the epenthesis is inserted after the geminate consonant.
3. If a word medial cluster of consonant contains a singleton and geminate in sequence, the epenthesis is inserted before the geminate consonant
4. If a word medial or final cluster of consonant contains two geminate consonants in sequence, the epenthesis is inserted between the two different geminates.
5. If three consonants appear in sequence word medially, the epenthesis will be inserted before the third consonant
6. If the sonority of the final consonant is greater than or equal to the preceding consonant, the epenthesis can be inserted between the final clusters.

3.2.2 Gemination

Gemination is the other critical factor that determines pronunciation and semantics in Amharic language. Like epenthesis gemination is also not shown in the language's orthography. However, unlike epenthesis, developing a set of rules for gemination in Amharic is very difficult. Tadesse, *et al.* [26] gives two broad foundations in order to determine the gemination of consonants: lexical and morphological.

According to the paper, lexical geminations cannot be easily predicted without making contextual disambiguation. However, morphological geminations can be, in most cases,

predicted by observing the orthography of the word. This prediction is especially successful on verbs. Gasser [27] has shown that such geminations can be inferred from morphological analyzers.

Getahun [30] explains three bases for determining geminations in Amharic words. The first is lexical geminations which are possessed by words because of their nature like *ቡና (bunna).* The second is morphological gemination which occurs because of morphemes; for example አዘረጋ (*azzeregga*) is a word found by joining *አስ (as*) and *ዘረጋ (zeregga).* The third is when two similar phones come contiguously like *አል_ለበሰም (al_lebbesem)* which changes to *አለበሰም* (*allebbesem*). While the first two bases are similar to [26] the last one is specialization of the second.

Verb Geminations

According to [29, 30], bi-radical (two consonant) words are geminated at their final radical during gemination. This is because, even though they appear as bi-radical verbs due to deleting one of their radicals, they are underlyingly tri-radical. Verbs may lose their medial or final radicals. Those verbs which lost their medial radical, verbs like ጻፈ (*s'afe*) and ሞተ (*mote*), will not be geminated. However verbs that lost their final radical, verbs like በራ (*berra*) and ገባ (*gebba*) will be geminated at their final radical.

Mulugeta [29], for the purpose of verb gemination analysis, divides verb roots into eight types according to their patterns of gemination in their perfect, future, manner and passive forms. While type 1 and type 2 verbs are bi-radical whose gemination rules are shown just one paragraph above, the verb types from 3 to 8 deal with mono- to quadri- radical verbs whose gemination rules are explained next:

Type 1: these are bi-radical verbs which are underlyingly tri-radicals. Since they have deleted their middle radical they will not be geminated in their perfect form. However, their first consonants are geminated in the passive and manner forms. Examples include: ጸፈ (*s'afe*)፣ ሞተ (*mote*).

Type 2: these are also bi-radical verbs which are underlyingly tri-radicals and have deleted their final radical. Hence, their final radical is geminated in the perfect, passive, future and manner forms. Examples include: በራ (*berra*) ፣ ገባ (*gebba*) ፣ ፈራ (*ferra*)

Type 3: these are tri-radical verbs that geminate the penultimate (the second in this case) radical in their perfect form. They differ in their future, passive and manner forms during gemination. Examples include: ሰበረ (*sebbere*) ፣ ወቀረ (*wekkere*) ፣ ተከለ (*tekkele*)

Type 4: these are quadri-radical verbs that geminate the penultimate (the third in this case) radical in the perfect form. They differ in the future, passive and manner forms during gemination. Examples include: ጠረጠረ (*t'eret't'ere*) ፣ መሰከረ (*mesekkere*) ፣ ከሰከሰ (*kesekkese*)

Type 5: these are mono-radical verbs that begin with a vowel. Their consonant is geminated in the perfect form. Examples include: አየ (*ajje*) ፣ አማ (*amma*) ፣ አጨ (*ac'c'e*)

Type 6: these are bi-radical verbs that begin with a vowel. The consonant that was found just next to the vowels is geminated in the perfect form. Examples include: አጨደ (*ac'c'ede*) ፣ አዘነ (*azzene*) ፣ አወቀ (*awweke*)

Type 7: these are tri-radical verbs that start with a vowel. They geminate their penultimate consonant in the perfect form. Examples include: አበደረ (*abeddere*) ፣ አመለጠ (*amellete*) ፣ አመረተ (*amerrete*)

Type 8: these are quadri-radical verbs that begin with a vowel. The penultimate radical is geminated during the perfect form. Examples include: አመሰገነ (*ameseggene*) ፣ አመለከተ (*amelekkete*) ፣ አመነዠከ (*amenez'z'eke*)

One can conclude from the previous listings that almost all root verbs geminate either the penultimate or the final radicals in the ***perfect*** form. The obvious exception from this rule is the bi-radicals (which are actually tri-radicals underneath) that delete their middle consonant or bi-radicals that do not begin with a vowel.

The gemination pattern for tri-radical roots in their perfect form is to geminate their penultimate consonant and for bi-radicals, it is to geminate their final radical. Anne and Klaus [31] further affirm that affixes are of no influence that the penultimate radical maintains its gemination during derivation. However, this seems to work only if the affixes do not alter the tense of the verb, i.e., the tense stays in its perfect form.

3.2.3 Prosodic Differences of Declarative and Interrogative Sentences

Prosodic features are most of the time beyond the segmental domain of speech. Prosody is often seen as the suprasegmental properties in speech. Such prosodic elements as tone, stress, rhythm, and intonation are at a level of a higher complexity than that of the mere segment. Pitch is the prosodic feature most centrally involved in intonation, and intonation can be defined as pitch movement. The *pitch* of a sound is the mental perception of fundamental frequency (F0); in general if a sound has a higher fundamental frequency we perceive it as having a higher pitch. The term 'fundamental frequency' refers to the number of repetitions of the regular waveform per second.

Intonation can be used to express personal relations, irony, one's real intention, emotion, and other information that cannot be shown in the transcription [32]. Intonation, being feature of sentences, is, most of the time, typically used to reflect differences among different kinds of sentences. Declarative utterances are different from their interrogative counterparts; and even among interrogative utterances, intonation differences are visible among WH questions and Yes/No questions. Ralf Kompe [33], summarized intonation differences among statements as the following:

- a falling intonation (pitch) marks an utterance as a declarative,
- a rising contour of pitch indicates a question, and
- Continuation-rise signals that there is a (prosodic) clause boundary, but that the speaker has not finished his turn yet (the use of commas).

Generally speaking, major differences of intonation are especially vivid at sentence-end positions. Meiko and Takako [32] agree with Kori when they cite his proposed five sentence-final intonation types, which are: questioning rising, emphasizing rising, falling, rising-falling, and flat. Among these *questioning rising* is the type typically observed at the end of interrogative sentences.

In some of the researches done on this regard [32], no significant difference was shown in the amount of rise in sentence-final intonation due to the type of interrogative sentence. Yuji Kawaguchi, *et al* [34] also argues that WH and Yes/No question follow similar sentence final intonation. In their paper they show that WH questions show rising pitch even at the beginning of the sentence which is at interrogative adjectives/adverbs such as "who", "what", "when" etc. Meiko and Takako, who researched intonation differences among interrogative sentences in Japanese language, argue that in their experiment they have observed that the Yes/No questions

were characterized by a smaller standard deviation of the pitch rise. This suggests that the distribution of the amount of rise for Yes/No question is relatively concentrated (concentrated at sentence-finals) and not scattered as WH questions. In the conclusion of [34], two prosodic positions are identified for interrogative utterances of the Turkish language. One on the interrogative adverbs of the sentence when the question is of type WH and the second on the sentence-final word which is actually shared by both the WH and Yes/No questions.

In another paper [35], Yosuke Igarashi gave summary of his experiment (done for Russian Language) as follows. First, the alignment of the peak at the end of the F0 rise was significantly later in Yes/No-Questions (YNQ) than in WH-Questions (WHQ). Second, the F0 value of the peak was significantly higher in YNQ than in WHQ. Third, the F0 value of the beginning of the utterance was significantly higher in WHQ than in YNQ. Finally, the F0 value of the end of the utterance tended to be higher in YNQ than in WHQ.

Even though the literatures referred are literatures done for languages other than Amharic language, they have universal applicability as noted by Joyce McDonough [36] when he said: "the distinction between declaratives and interrogative and focus constructions is among the most universal characteristics of intonation. We normally associate a fall in pitch with the end of declaratives indicating finality, a rise for interrogatives indicating continuation." Jun Xu [37], in his revision of different languages on this issue, also summarizes that interrogative utterances show rising terminal (go up at the end).

4. Requirement Specification and Design of the System

In this section, the requirement specification and the design of the AmahricTTS_Synthesizer are presented. Use cases, class diagrams, sequence diagrams, subsystem and deployment diagrams are used for the presentation.

Functional Requirements

1. Allow the user to type his/her text
2. Produce sound based on the given text

   2.1 Produce intelligible and natural-like sound by making use of multiple recorded voices

   2.2 Produce intelligible and natural-like sound by controlling epenthesis and gemination.

   2.3 Generate intonation/prosody by making use of punctuations and positions of syllables in a given word

Non-functional requirements/properties:

1. The application is a desktop application that requires standalone installation on a PC
2. The application is integrated with MS-Word
3. The system allows for addition of more rugged sound units as long as they follow the labeling scheme of the system
4. The machine that the synthesizer runs on should have at least 50MB free HD space
5. The text given for synthesis should follow the standard Amharic grammar and use of punctuation

4.1 Use Case Diagram:

In the figure below (*Figure 4.1*), the functional requirements of the system are presented.

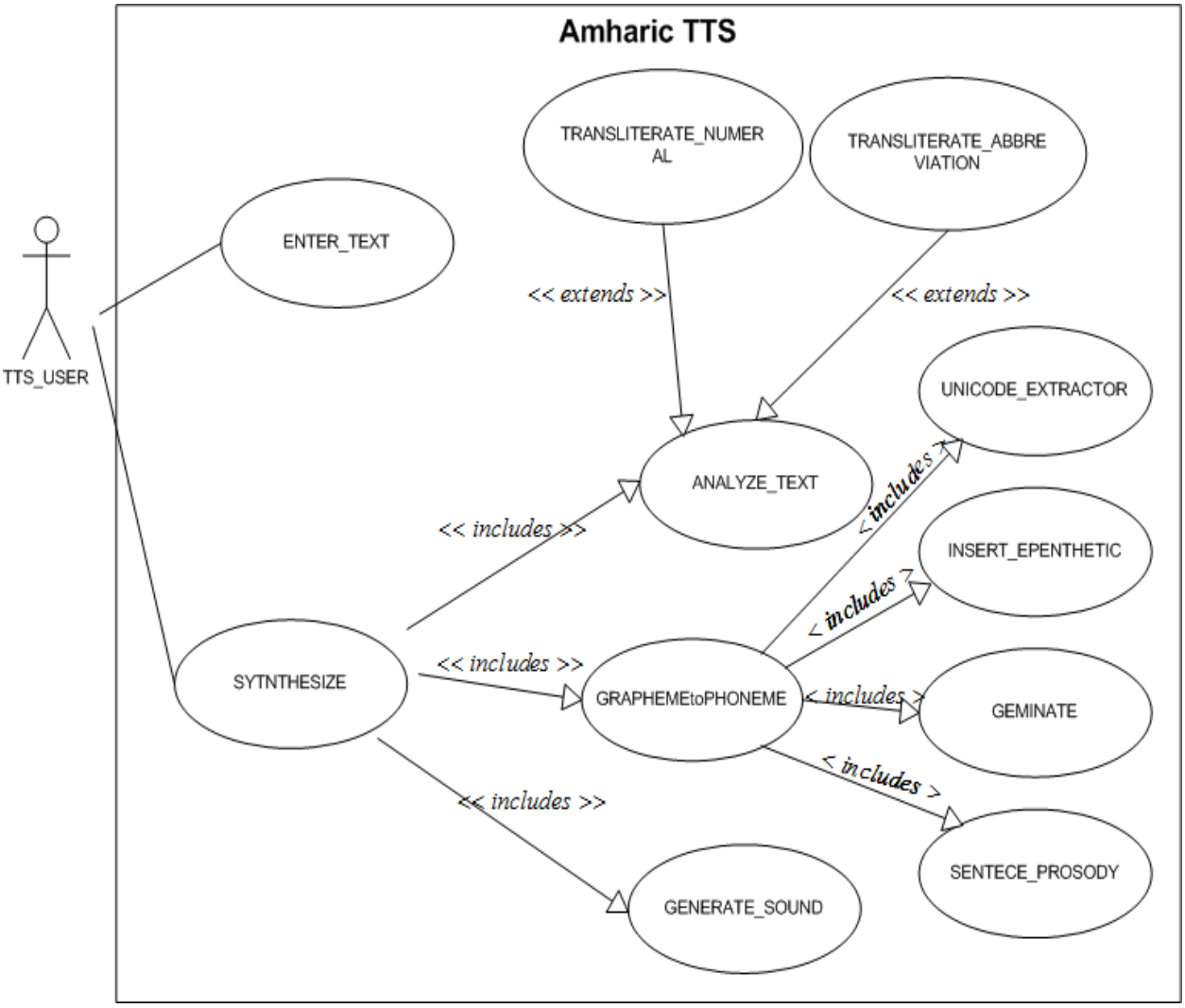


Figure 4.1 Use Case of AmharicTTS_Synthesizer

4.1.1 Description of the Actor

The TTS_USER is the individual who runs the application to listen to his/her text being pronounced.

Description of use cases:

Table 4.1 - ENTER_TEXT use case description

Use case Name	ENTER_TEXT	
Actor	TTS_USER	
Description	allows the TTS_USER to enter Amharic statements	
Precondition	the text typed-in should be in Amharic	
Flow of events		
	1	the user enters Amharic text
	2	the system assigns the text entered in to a variable and temporarily stores it for further processing
Post-condition	the given text is assigned to a variable	

Table 4.2 - SYNTHESIZE use case description

Use case Name	SYNTHESIZE	
Actor	TTS_USER	
Description	allows the TTS_USER to synthesize Amharic text	
Precondition	the variable shouldn't be null	
Flow of events		
	1	the system invokes the 'ANALYZE_TEXT' use case
	2	the system invokes the 'GRAPHEMEtoPHONEME' use case
	4	the system invokes the 'GENERATE_SOUND' use case
Post condition	the user listens as her text is being pronounced	

Table 4. 3- ANALYZE_TEXT use case description

Use case Name	ANALYZE_TEXT	
Description	allows the system to tokenize and filter Amharic text	
Precondition	A text with more than one word should contain space and/or other delimiting symbols	
Flow of events		
	1	the system gets the text entered
	2	the system extracts inaudible characters and truncates them from the text
	3	the system gets the symbols or punctuation marks used for delimiting words and sentences
	4	the system divides the text based on the given delimiters and assigns them to a string array
	5	the system identifies and transliterates numerals and abbreviations
Post condition	the given text will be tokenized and its punctuation marks identified	

Table 4.4 - GRAPHEMEtoPHONEME use case description

Use case Name	GRAPHEMEtoPHONEME	
Description	allows the system to convert the graphical representation of texts into phonemes	
Precondition	the text should be tokenized	
Flow of events		
	1	the system gets the tokenized words
	2	The system invokes the UNICODE_EXTRACTOR use case
	3	The system invokes INSERT_EPENTHETIC use case if the word contains consonants
	4	The system invokes GEMINATE use case if the word is a verb
	5	The system invokes the SENTENCE_PROSODY use case
Post condition	a phone set will be produced	

Table 4.5 – UNICODE_EXTRACTOR use case description

Use case Name	UNICODE_EXTRACTOR	
Description	extracts Unicode values of each character in the tokenized words	
Precondition	tokenized words are received first	
Flow of events		
	1	the system gets the tokenized words
	2	the system splits the each word to characters and retrieves their Unicode value
Post condition	Unicode value of each character is returned	

Table 4.6 – INSERT_EPENTHETIC use case description

Use case Name	INSERT_EPENTHETIC	
Description	Controls epenthesis	
Precondition	consonants are received first	
Flow of events		
	1	the system gets the consonant and its location
	2	The system checks the location of the consonant
	3	the system inserts the epenthetic vowel if the consonant is word initial
	4	The system inserts the epenthetic vowel after two consecutive consonants, if there are 3 consecutive consonants medially
	5	the system *checks the sonority* of the consonant against the word final consonant if the consonant is a penultimate consonant
Post condition	the epenthetic vowel is inserted where necessary	

Table 4.7 – CHECK_SONORITY use case description

Use case Name	CHECK_SONORITY	
Description	compares sonority of the penultimate character against the word-final	
Precondition	the penultimate and the final characters are received first	
Flow of events		
	1	the system assigns values to the consonants according to their level
	2	the system compares the values
	3	the system returns *false* if the sonority level of the final consonant is less than that of the penultimate, *true* otherwise
Post condition	*false* or *true* is returned	

Table 4.8 – GEMINATE use case description

Use case Name	GEMINATE	
Description	controls gemination on the perfect form of verbs in a given sentence	
Precondition	the verb should be located, and received	
Flow of events		
	1	the system uses a simple POS tagging process to determine if a given word is a verb
	2	the system determines if the verb is in its perfect form and retrieves its length
	3	the system detects suffixes and ignores them
	4	the system geminates the penultimate or final character according to the word's character size
Post condition	the verb's phone is geminated where appropriate	

Table 4.9 – SENTENCE_PROSODY use case description

Use case Name	SENTENCE_PROSODY	
Description	allows the system to assign intonations based on punctuation marks	
Precondition	the punctuation marks should be identified	
Flow of events		
	1	the system gets the punctuations
	2	the system generates the phone set of the last word in the sentence
Post condition	a phone set for the last word in the sentence will be produced	

Table 4.10 – GENERATE_SOUND use case description

Use case Name	GENERATE_SOUND	
Description	allows the system to generate sounds based on the phone set	
Precondition	the complete phone set should be produced	
Flow of events		
	1	the system gets the phone set produced
	2	the system selects and sequences the appropriate sound units
	3	the system plays back the sound units selected
Post condition	sound is heard from the computer's speakers	

4.2 Class Diagram

In this figure (*Fig 4.2*) the classes together with their attributes and methods of the system are shown using class diagram.

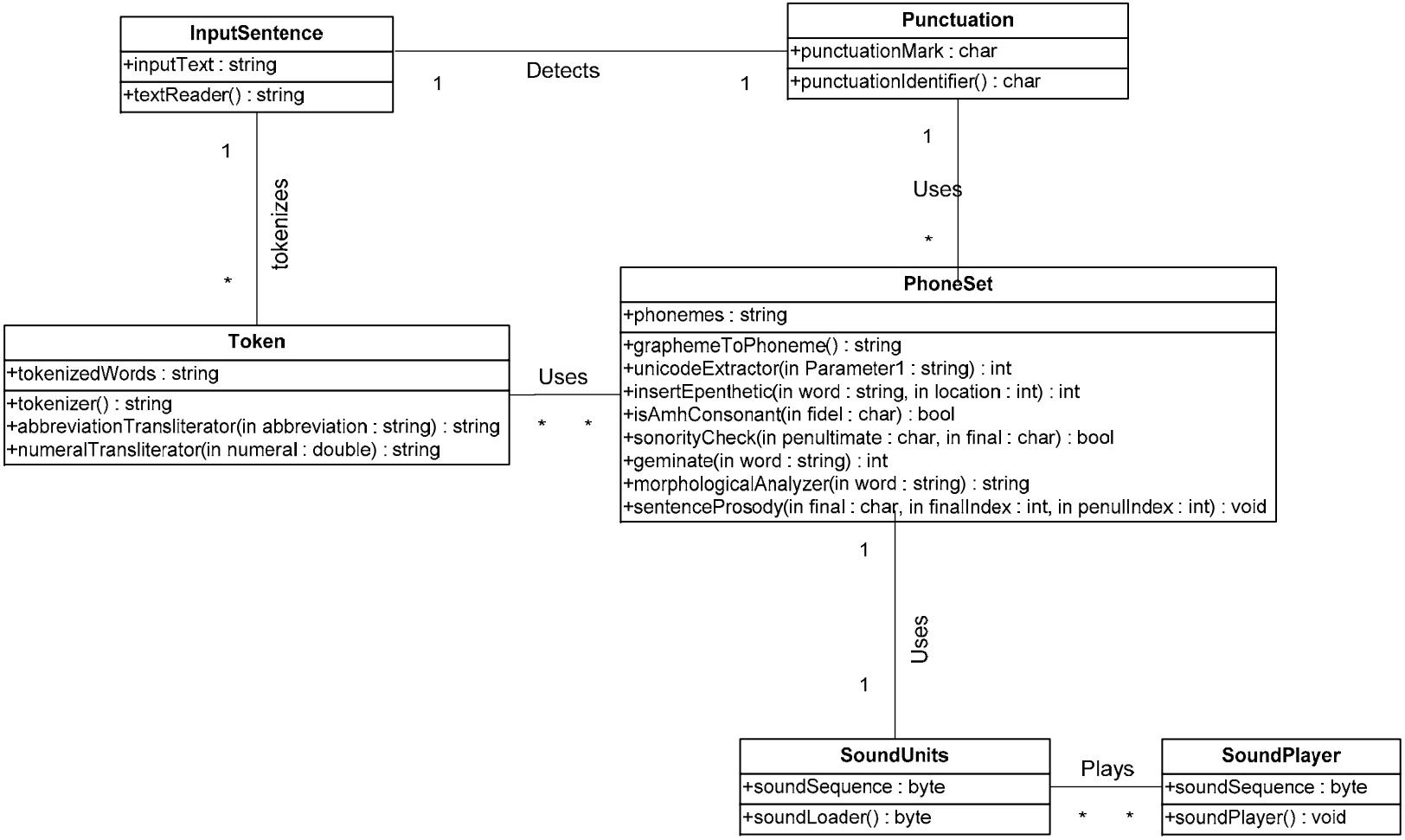


Figure 4.2 – class diagram for AmharicTTS_Synthesizer

4.2.1 Class Description

In this section, descriptions of the methods and attributes of the aforementioned classes are given.

Attributes' Description:

inputText – this is a private string variable that stores the text entered by the user for synthesis.

tokenizedWords – this is a private string array variable that stores the words extracted from the i*nputText*.

punctuationMark – this is a private character variable that stores the punctuation mark used in the text.

phonemes – this is a private string variable that stores the phonemic representation of the given text

soundSequence –this is a private byte variable that stores the sound units to be played to produce the sounds.

Methods' Description:

textReader() – this is a public method that reads in the user's text from the GUI and saves it in the *inputText* variable.

tokenizer()

input – *inputText*
output – *tokenizedWords, transliteratedTokenizedWords*
task – segments the *inputText* into words making use of delimiters such as space and Amharic punctuation marks

abbreviationTransliterator()

input – *inputText* (abbreviations)
output – *tranlisteratedWords*
task – checks the slash or dot symbol and transliterates abbreviations

numeralTransliterator()

input – *inputText* (numbers)
output – *tranlisteratedWords*
task – checks the presence of numerals and transliterates them

punctuationIdentifier()

input – *inputText*
output – *punctuationMark*
task – determines the end of a given sentence and identifies the punctuation mark used

graphemeToPhoneme()

input – *tokenizedWords, punctuationMark*
output – *phonemes*
task – converts the extracted words into their equivalent phonetic representations. This method uses epenthesis and gemination rules, and the punctuation marks to make appropriate phonetic representation.
uses – *unicodeExtractor(), insertEpethetic(), geminate(), isAmhConsonant(), sonorityCheck(), sentenceProsody(),* and *numericPhone()* methods.

unicodeExtracter() – assigns Unicode values to all characters in the given word

insertEpethetic() – inserts the epenthetic vowel where ever appropriate

isAmhConsonant() – determines if a character is Amharic consonant

sonorityCheck() – compares the sonority level of the penultimate and the final consonants of a word

geminate() – geminates the penultimate or the final consonant of a verb in its perfect form
uses – *morphologicalAnalyzer()*

morphologicalAnalyzer() – extracts a given verb into its stem and morpheme

sentenceProsody() – manipulates the final word in an interrogative sentence to reflect YNQ or declarative intonations

soundLoader()

input – *phonemes*
output – *soundSequence*
task – chooses the appropriate sound units from the sound base and sequences them for playback.

soundPlayer()

input – *soundSequence*
output – *sound*
task – plays back the *soundSequence*

4.3 Sequence Diagram

The figure (*Fig 4.3*) that shows the interaction among objects of the system is shown below

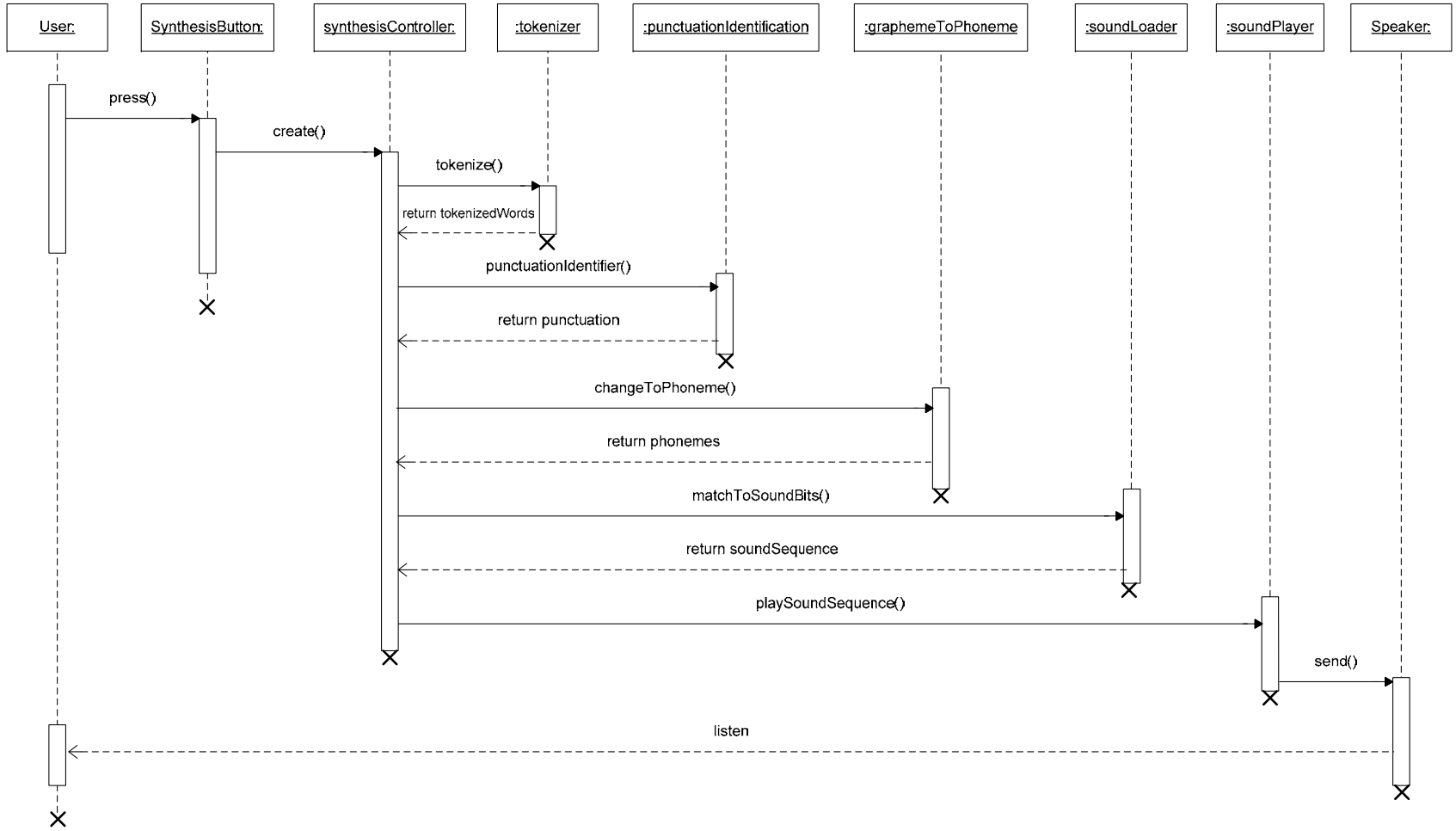


Figure 4.3 - Sequence Diagram for AmharicTTS_Syntherizer

4.4 System Design

Subsystem decomposition, software/hardware mapping and object design of the system are dealt in this section.

4.4.1 Subsystem Decomposition

The system has three subsystems: TEXT_ANALYSIS, GRAPHEME-TO-PHONEME, and SOUND_GENERATION subsystems.

The TEXT_ANALYSIS subsystem is responsible for text preprocessing tasks such as tokenizing, identifying sentence-end punctuations, truncating other insignificant symbols, and transliterating abbreviations and numerals.

The GRAPHEME-TO-PHONEME subsystem is responsible for generating the phonemic representation of the text being pronounced. In the process, this subsystem uses the gemination rules, the epenthesis rules and the sentence prosody features described in *Section 3.2*.

The SOUND_GENERATION subsystem is responsible for sequencing the appropriate phones in the sound bank and playing them according to their sequence.

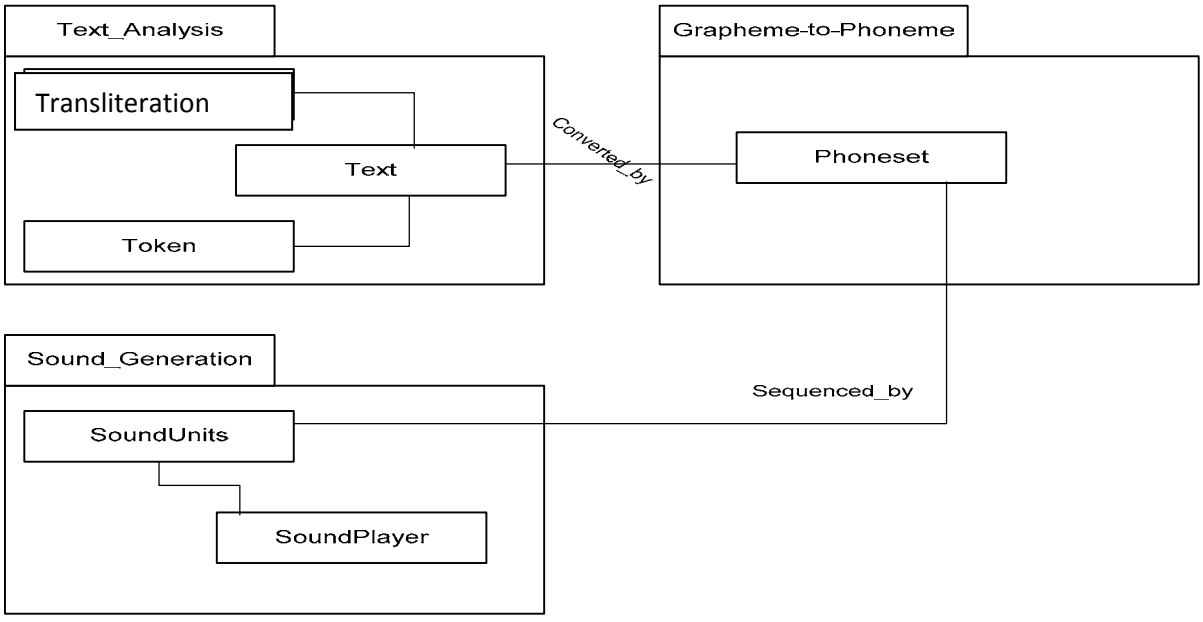


Figure 4.4 - Subsystem Decomposition of the AmharicTTS_Synthesizer

4.4.2 Deployment Diagram:

Since the system is meant to be just a desktop application, it requires only a personal computer to install and run it.

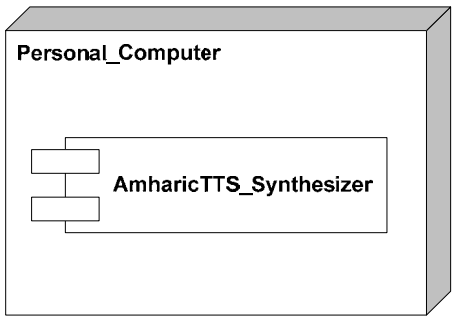


Figure 4.5 - Deployment Diagram of the AmharicTTS_Synthesizer

4.4.3 Object Design

This section deals with invariants, preconditions and post condition of the methods specified earlier in *Section 4.2* of this report.

Invariant:

inputText<>null
punctuationMark = '::' or '?'
tokenizedWords *shouldn't contain* punctuation symbols *and*/or space(s)

Conditions:

textReader():
precondition – text is available in the text field of the GUI
postcondition – text is copied to *inputText*

tokenizer()
precondition – *inputText* shouldn't be null
postcondition – the *inputText* is divided into words

punctuationIdentifier():
precondition – not more than one punctuation mark per sentence should be specified
postcondition – the *punctuationMark* is identified and stored.

numeralTransliterator():

precondition – numerals should exist

postcondition – the equivalent transliteration for the numeral is produced

abbreviationTransliterator():

precondition – the symbol “/” or “.” should exist

postcondition – the equivalent transliteration for the abbreviation is produced

graphemeToPhoneme():

precondition – *tokenizedWords* and *punctuationMark* should exist

postcondition – an equivalent phoneme is produced for each grapheme

soundLoader():

precondition – *phonemes* should exist

postcondition – appropriate sound units will be selected and sequenced

soundPlayer():

precondition – sequence of sound units should exist

postcondition –sound is produced and heard through the PC’s speakers

4.5 System Architecture

The following figure (*figure 4.6*) summarizes previously given details and presents the system's general architecture.

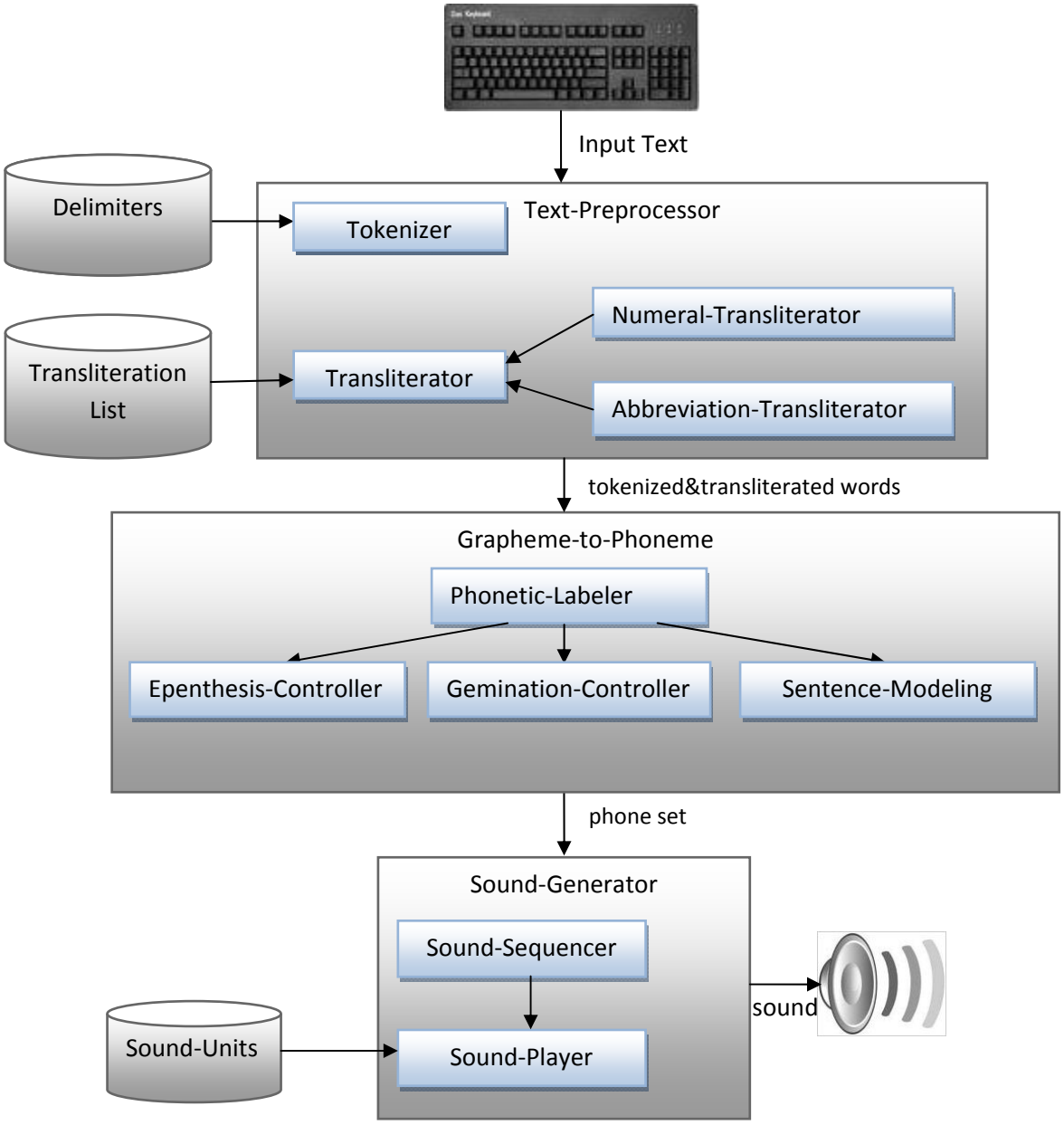


Figure 4.6 – System architecture

5. Implementation

This section deals with development tools, interface design, sound units production, coding, and Microsoft Word integration of the system.

5.1Tools

Various tools have been used while designing and developing the system. They are listed in accordance to their use.

Sound Units/Phones Production

JetAudio™ 7.5.5.25, WaveSurfer™ 1.8.8pl-1010071431 and *Praat™ 5.2.18* are used for voice recording, labeling, waveform analysis and manipulation. Praat and Wavesurfer are programs used for the analysis and reconstruction of acoustic speech signals, and also for doing phonetic analyses and sound manipulations.

Development

The system is developed using *Microsoft Visual Studio 2010* for interface design, development/coding, and Microsoft Word 2007 integration. Visual C# 2010 is the development language used throughout.

Documentation

Microsoft Word 2007 is used for generating the documents; and Microsoft Visio 2003 for producing Unified Modeling Language (UML) diagrams.

5.2 Building the Sound Units and User Interfaces

In this section details of the techniques and procedures followed for building the sound units and designing the user interface are presented.

5.2.1 Sound Units

All the recordings were made using the specified tools mentioned before with sampling rate of 48000 Hz by an individual who is a native speaker of the language.

One of the primary jobs of developing concatenative text-to-speech synthesis systems is building the appropriate phone set. It is from this set of phones that the developed algorithms make the appropriate selection for pronouncing a given text. The approach taken by this work for concatenative synthesis is the unit selection method. Since unit selection based synthesizers choose suitable fragments from a database of speech and join them together with minimal signal modifications, it is mandatory to have multiple phones recorded for the same phoneme in order to avoid/minimize signal modification.

Allophones

Each sound unit in the sound bank represents a phone or a syllable based on the CV template only. Each character in the Amharic alphabet is made to be represented by a minimum of two allophones. In addition to that, each of the first (ግዕዝ), second (ካልዕ), fourth (ራብዕ) and unique (diphthongs) orders has one additional allophone to represent the stressed version of that phoneme; and each of the ሳድስ (sixth) orders also has one additional allophone to represent the consonant phones without the epenthetic vowel ‘እ’ (‘ɨ’).

Labeling

The sound units recorded for each phoneme are given their corresponding Unicode value as their file name. However, since each phoneme has at least two phones associated with it, one extra character is appended to their Unicode value for differentiating purpose among the allophones.

The labeling scheme looks like:

[*Unicode_value*] – represents the unstressed version of the phonemes

[*Unicode_value*]+ ‘0’ – represents the consonant phone of the sixth order characters

[*Unicode_value*]+ ‘1’ – represents the geminated version of the phonemes

[*Unicode_value*]+ ‘2’ – represents the interrogative version of the phonemes

[*Unicode_value*]+ ‘3’ – represents the declarative version of the phonemes

Selection of the Corpus

The words used for producing the corpus are collected in such a way that all the required Amharic phones are included. The words collected in this process are cross-referenced with Amsalu Aklilu’s Amharic – English dictionary [38] to get their correct pronunciation.

Target and Joining Costs

One of the critical issues that determine the quality of a synthesized speech in unit selection methods is the cost related to finding the target sound units and their concatenation. Target cost is concerned with choosing units which are as similar to the desired output as possible. Joining/concatenation cost is concerned with minimizing perceptible acoustic mismatch between pairs of units to be concatenated [12]. In order to minimize these costs, i.e., to easily locate target sound units and minimize acoustic mismatch, two methods are employed. One is to give appropriate label to each sound unit, and the second is to record each phone at positions where its intonation can be captured effectively and modify the waveform.

Unstressed version of each phone is recorded and segmented both at the beginning and end of words and sentences. While the labeling scheme minimizes the cost of finding the target sound unit, multiple phones recorded at different positions help minimize the concatenation cost. In addition to the previous techniques, digital signal processing (DSP) technique is also used to reduce perceptible mismatch of concatenated units. DSP is performed using *WaveSurfer*™ by applying its "fade" functionality. This functionality is used to smooth/fade waveforms as they reach the end of their duration. *Figure 5.1* shows this process:

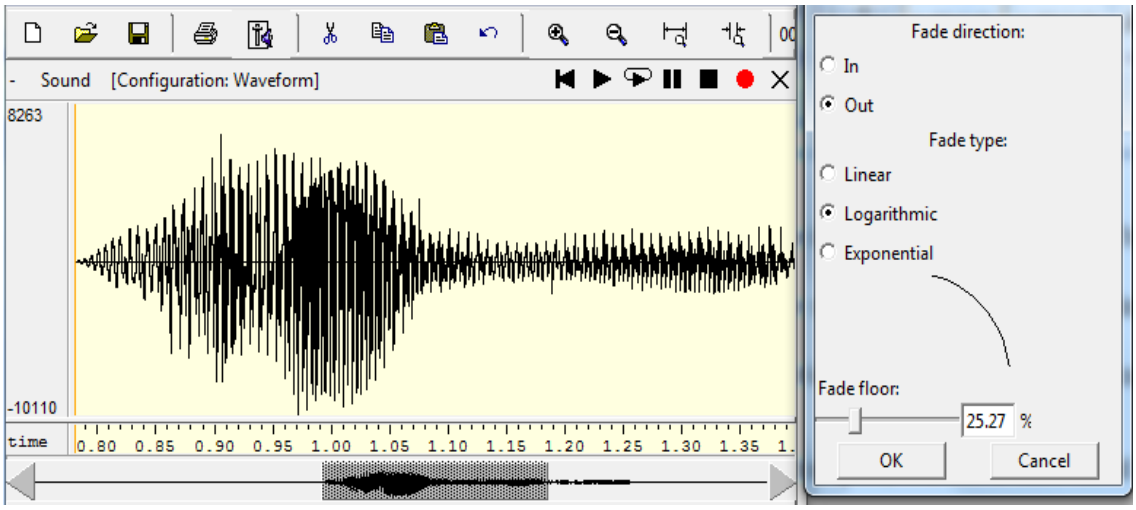


Figure 5.1 Sound units' waveform modification (DSP)

5.2.2 User Interface

The system has two user interfaces: one as a standalone and the other as a plug-in for Ms-Word 2007. The standard Windows approach is followed for the design and coloring of the forms. *Figures 5.2* and *5.3* show both interfaces with short description on the function of each control on the forms.

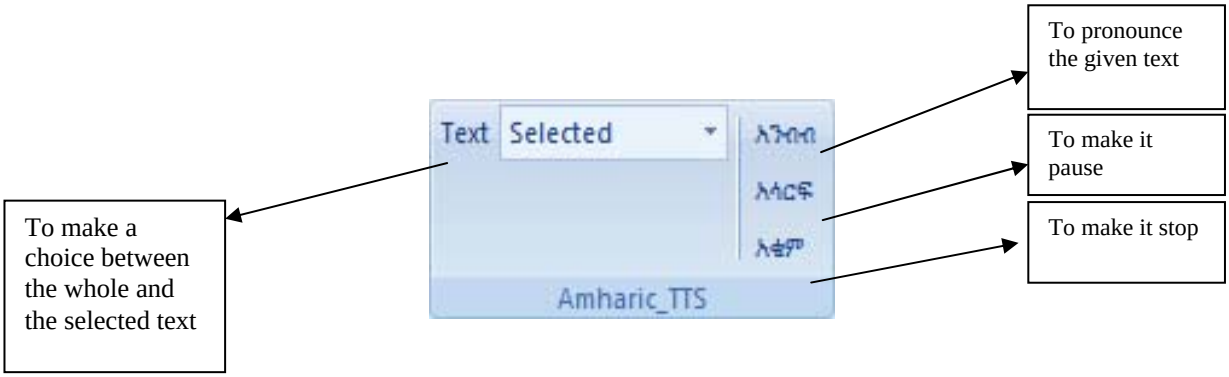


Figure 5.2 Snapshot of the AmharicTTS_Synthesizer MS-Word 2007 Add-In form

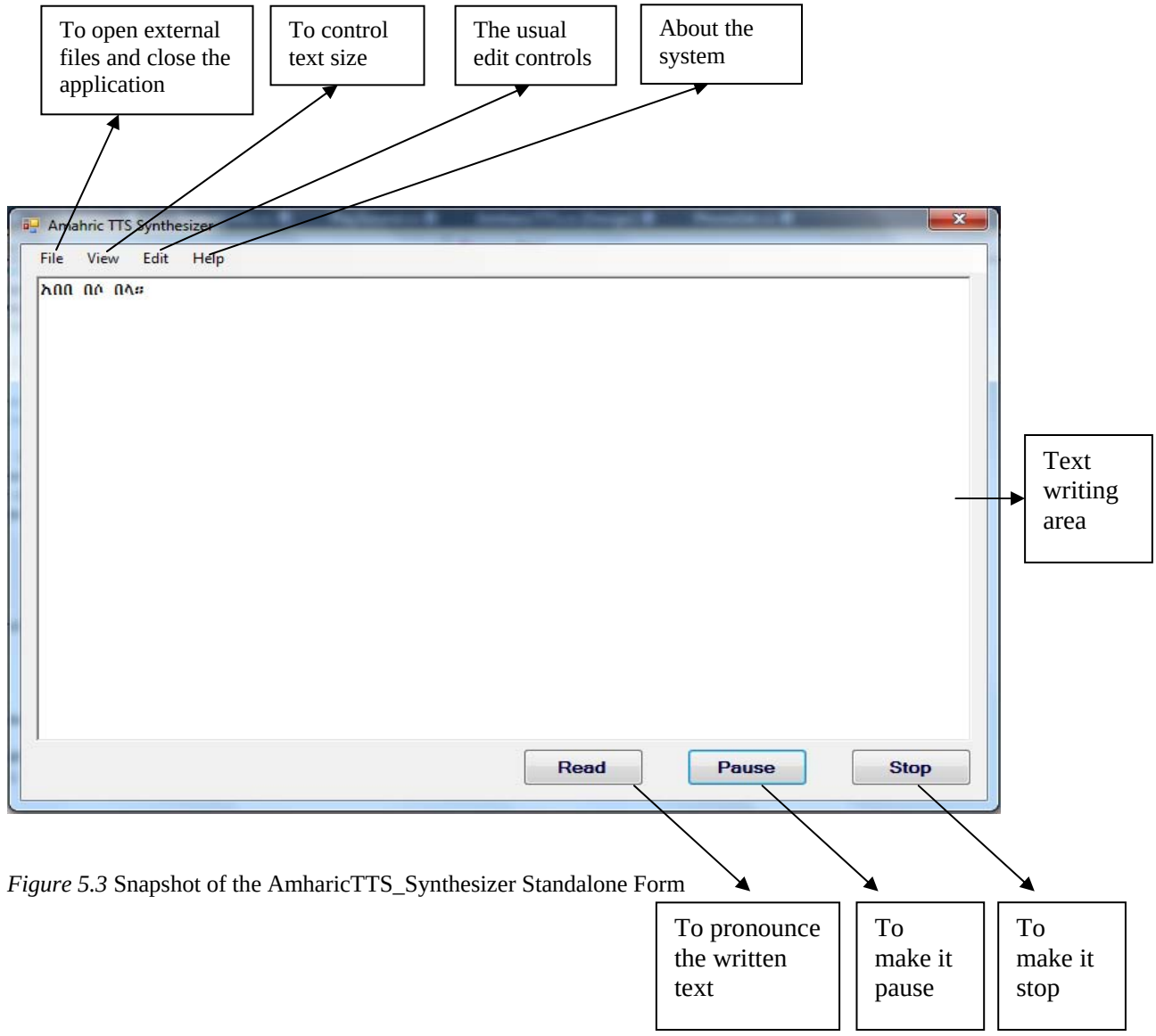


Figure 5.3 Snapshot of the AmharicTTS_Synthesizer Standalone Form

5.3 Text Preprocessing

Text preprocessing deals with segmentation of text into meaningful chunks and converting numerals and abbreviations into their equivalent orthographic representation.

Two major tasks are done in this phase: tokenizing text into words, transliterating numerals and abbreviations.

5.3.1 Tokenization

The system tokenizes any amount of text into its constituting words by taking Amharic symbols as delimiters. In the process, disambiguation of the symbol "." is done when used in abbreviations and decimals. The following table (*Table 5.1*) shows the Amharic symbols used for the tokenization.

Table 5.1 – Amharic Punctuation Marks

Amharic Symbols	፡	።	፤	፥	፦	፧	?
English Equivalent	Space	.	;	:	,	,	?

5.3.2 Transliteration of Numerals and Abbreviations

Even though old/traditional Amharic texts use the Geez numerals for representing numbers, these days, however, the trend has changed so much that virtually all Amharic writings follow the Arabic numerals for representing numbers. The developer of the system believes that it is more appropriate to follow this trend and develop transliterating schemes for the Arabic numerals in this project. In the process of transliterating numerals, considerations on Amharic alphabets that may precede numerals are made. The following characters are the widely used characters that precede numerals: በ (*be)*, ለ (*le)*, ከ (*ke)*, የ (*je)*. A relational table is used to

facilitate the transliterating process. The use of database table also allows modification and expansion of the transliterating scheme without touching the front end. *Table 5.2* shows the design and sample of the database table. Each digit from 0 to 9 is represented by 15 rows, which means the transliterating scheme works up to one hundred trillion.

Table 5.2 – Transliteration of numerals according to their power position

Digit	**Power Position**	**Value**
5	0	አምስት
5	1	ሀምሳ
5	2	አምስት መቶ
5	3	አምስት ሺ
5	4	ሀምሳ

Transliterating abbreviation also followed the same approach; a database table is implemented to expand abbreviations to their equivalent orthographic representations. Even though Amharic is known to use the forward slash symbol for abbreviating texts, it is now customary to use the dot symbol. An exhaustive list of all abbreviations in the language is not produced; however, the common ones are selected and implemented. The design followed for developing this feature allows the addition of more items to the list of abbreviations to make the system transliterate virtually all abbreviations. *Table 5.3* shows the schema and sample of the database table.

Table 5.3 – Transliteration of abbreviations with "/" or "." symbols

Abbreviations	**Value**
ዓ/ም	ዓመተ ምህረት
አ.አ.ዩ	አዲስ አበባ ዩኒቨርሲቲ

5.4 Grapheme to Phoneme Conversion Technique

One of the most difficult jobs of unit selection synthesis is choosing the appropriate phone from multiple phones. And this is largely determined by the grapheme to phoneme conversion

technique one employs. Selection of appropriate phones from a phone set needs to make quite a good consideration of context not merely phonemic representation. Phonemes differ according to context/position, part of speech, and neighboring punctuation. In the sections that follow explanation is given on how such a challenging and yet a critical process is handled.

Amharic grapheme to phoneme conversion is a challenging task. This is especially true when we think of epenthesis and gemination.

5.4.1 Epenthesis

The detail of its description and behavior in Amharic phonology is given in *Section 3.2.1*. Here, the implementation techniques employed in the Amharic TTS system will be discussed.

Mulugeta's and Getahun's [29, 30] work are used on dealing with the epenthetic vowel "ɨ". What is not included in the development of the project from their work is the rule of controlling epenthesis when there is/are geminated consonants in a given consonant cluster. *Listing 5.1* presents the algorithm used for this purpose:

1. Split the given word into character array
2. While there is character in the character array
3. Check if the character is consonant (which means it is the 6^{th} order in the alphabet series)
4. If the position of the consonant is word-initial, insert epenthetic vowel
5. If the position of the consonant is word-final, ignore epenthesis
6. If the position of the consonant is next to word initial and is not the penultimate consonant, ignore epenthesis

7. If the consonant is not the second and penultimate character, check its previous neighbor to determine if it violates the maximum consonant cluster allowed; if it violates then insert epenthetic vowel, else ignore epenthesis
8. If the consonant is the penultimate character and the final character is also a consonant then check their sonority level
9. If the sonority level of the penultimate is less than that of the final then insert epenthesis after the penultimate
10. End while

Listing 5.1 Algorithm to control epenthesis

The following table (*Table 5.4*) is used to determine the sonority level of consonants:

Table 5.4 – Sonority level of consonants according to their manner of articulation. The higher the assigned number the higher the sonority level.

Manner of Articulation		**Sonority Level**
Stops	Voiced	1
	Voiceless	2
	Glottalized	3
Fricatives	Voiced	4
	Voiceless	5
	Glottalized	6
Nasals		7
Liquids		8
Glides		9

5.4.2 Gemination

The detail of the description and behavior of gemination in Amharic phonology is given in *Section 3.2.2*. Here, the implementation details of it in the system are given.

The time and resource (resources such as Amharic lexicon with pronunciation information) given for this project have imposed the limitation that gemination rules are implemented on Amharic verbs only. And from Amharic verbs, the focus is on verbs in their ***perfect*** forms only. Of course, the inflection of those verbs in regard to ***person, gender*** *and* ***number*** *is also considered*. This scope is chosen for four reasons: first, because it is easy to determine the gemination of verbs only from the orthography of those verbs even during inflection [26]; second, all types of tri-radical verb roots geminate their penultimate radical in the perfect form only, this is easily seen from *Table 5.5*; third, it is easy to determine the location of Amharic verbs in any *simple* Amharic sentence; last, much of the literature reviewed happen to be dealing with only Amharic verbs extensively.

One other limitation that should be explained is why only the person, gender and number inflection forms are chosen. This is because these inflections are represented by adding only suffixes; and according to [31], suffixes are of no influence to the general gemination rule given in *Table 5.5*. Prefixes such as "አስ" (*as*) change/add-to the gemination rule because of assimilation processes. For example the word "ዘረጋ" (*zeregga*) will geminate "ዝ"(*z*) when prefixed with the bound morpheme "አስ"(*as*) because of the similar consonants "ዝ"(*z*) and "ስ"(*s*).

Table 5.5 – Gemination of tri-radical verbs. Adopted from [39]

Tense	**Type-A**	**Type-B**	**Type-C**
perfect	Gemination	Gemination	Gemination
imperfect	No-Gemination	Gemination	Gemination
participle	No-Gemination	Gemination	No-Gemination

Determining Gemination from Orthography

Since the system does not have a lexicon or another system that tells the type of verb under process, a technique should be developed to enforce gemination based on the orthographic

information available. From observation and by looking into the orthographic properties of the verbs used to show gemination rules in [29] and [38], the following orthographic-based rules can be derived given *that the verbs are in their perfect form and are only inflected according to person, gender and number.*

1. If the verb is bi-radical and ends with "ፅ" (*c'e*) then "ፅ" (*c'e*) is geminated.
2. If the verb is mono-radical and starts with vowel then the final radical is geminated.
3. If the verb is bi-radical and the penultimate radical is first order and the final is fourth order then the final radical is geminated.
4. If the verb is bi-radical and starts with a vowel, step 3 applies. But if the final radical is first order instead of being fourth order, the penultimate is geminated instead of the final.
5. If the verb is tri-radical, and all its radicals are first order, the penultimate is geminated.
6. If the verb is tri-radical (excluding the suffix) and it has one of the suffixes listed in *Table 5.6*, the penultimate is geminated.
7. If the verb is tri-radical or bi-radical that starts with a vowel and has none of the suffixes listed in *Table 5.6:*
    a. The final radical is geminated if its penultimate radical is first order and its final is fourth order
    b. Both the final and the penultimate radicals are geminated if both are fourth order and represent different phonemes[4].
    c. The final radical is geminated if both the final and penultimate radicals are fourth order and represent the same phoneme
8. If the verb has more than three radicals and it ends with one of the suffixes listed in *Table 5.6*:

[4] *It should be noted that meaningful Amharic verbs can be found without geminating the final radical. However, such verbs are not in their perfect form and are not the kind of verbs found at the end of simple statements.*

a. The penultimate radical of ***the word obtained by trimming the suffix*** is geminated if the trimmed word has three or more radicals, if its final radical is first/second/fourth/sixth/unique order and its penultimate radical is first order.
b. The final radical of ***the word obtained by trimming the suffix*** is geminated if the trimmed word is bi-radical and its final radical is fourth order

9. If the verb has more than three radicals and has none of the suffixes listed in *Table 5.6*, the penultimate is geminated.

Table 5.6 – suffixes of Amharic verbs based on person, gender and number inflections

Suffix Size	**Suffixes**
one character	ን፣ኝ፣ሽ፣ት፣ው፣ክ፣ህ፣ሁ፣ኩ፣ች
two character	ሽኝ፣ችኝ፣ክን፣ሽን፣ችን፣ኩህ፣ኩሽ፣ችሁ፣ችህ፣ንህ፣ችሽ፣ንሽ፣ኳት፣ነው፣ኋት፣ቻት፣ኩት፣ሻት፣ናት፣ቸው፣ኧኝ፣ክኝ፣ኩኝ፣ሁኝ፣ኧን፣ውህ፣ሁሽ፣ሁት፣ሁህ፣ለት፣በት፣ላት፣ባት፣ብህ፣ልህ፣ልሽ፣ብሽ፣ልኝ፣ብኝ፣ልን፣ብን፣ካት፣ከው
three character	ችሁኝ፣ኳችሁ፣ችሁን፣ናችሁ፣ቻችሁ፣ኳቸው፣ሻቸው፣ናቸው፣ቻቸው፣ኋችሁ፣ኋቸው፣ላቸው፣ባቸው፣ላችሁ፣ባችሁ፣ችበት፣ችለት፣ችባት፣ችላት፣ችብህ፣ችልህ፣ችልሽ፣ችብሽ፣ችብኝ፣ችልኝ፣ችብን፣ችልን፣ክበት፣ህበት፣ክለት፣ህለት፣ህባት፣ህላት፣ክባት፣ክላት፣ክብኝ፣ህብኝ፣ክልኝ፣ህልኝ፣ክብን፣ህብን፣ክልን፣ህልን፣ሸበት፣ሸለት፣ሸባት፣ሸላት፣ሸብኝ፣ሸልኝ፣ሸብን፣ሸልን፣ኩበት፣ሁበት፣ኩለት፣ሁለት፣ኩባት፣ኩላት፣ሁላት፣ሁባት፣ኩብህ፣ሁልህ፣ኩብሽ፣ሁልሽ፣ሁብህ፣ኩልህ፣ኩልሽ፣ሁብሽ፣ንባት፣ንላት፣ንብህ፣ንልህ፣ንብሽ፣ንልሽ፣ንለት፣ንበት፣ችኋት፣ችሁት፣ካቸው
four character	ችኋቸው፣ችባቸው፣ችላቸው፣ችላችሁ፣ችባችሁ፣ክባቸው፣ክላቸው፣ህባቸው፣ህላቸው፣ሸላቸው፣ሸባቸው፣ኩላቸው፣ኩባቸው፣ኩባችሁ፣ሁላችሁ፣ሁባችሁ፣ኩላችሁ፣ችሁብኝ፣ችሁልኝ፣ችሁልን፣ችሁብን፣ችሁባት፣ችሁላት፣ችሁበት፣ችሁለት፣ንባችሁ፣ንላችሁ፣ንባቸው፣ንላቸው
Five character	ችሁባቸው፣ችሁላቸው

The system has used the previous rules and tables to control gemination in Amharic verbs. However, it is known that this is not complete and without exceptions; instead it is recommend that further research is conducted in this area of determining gemination from orthographic

properties of Amharic words. *The developer of this system believes that the result of such researches can help the rule-based grapheme-to-phoneme conversion process*. It is also worth mentioning that many of the exceptions that may not work according to the aforementioned rules are also exceptions to gemination rules given in the literatures referenced in *Section 3.2.2* and 5.4. One of the exceptions that violate these rules is when bi-radicals that start with the vowel "አ" (*a*) are evaluated against *rule 8/b*. Let us use one example to make it clear. In the verb "አጠራ" (*at'rra*) the consonant "ር" (*r*) is geminated; and in the verb "አጠረ" (*at't'ere*) the consonant "ጥ" (*t'*) is geminated correctly according to *rules 3* and *4* respectively. When both are inflected, "አጠራቸው"(*at'errachew*) or "አጠራቸው"(*at't'erachew*) may result. However, the consonant "ር" (*r*) is geminated according to *rule 8/b*, leaving out the possibility of geminating "ጥ" (*t'*). This problem can be resolved by using semantic disambiguation which is out of the scope of this work.

5.5 Yes/No Interrogative Prosody Representation Technique

This project work deals with modeling the interrogative and declarative prosodies. However, the modeling is focused on modeling the last word of the sentences, and even from the last word, it is the last sound unit that is modeled. The scope is this limited because the conspicuous difference between declarative and interrogative sentences is concentrated on the sentence-final word as the experiment from [32] conclude by saying "Judgment Requests were characterized by a smaller standard deviation. This suggests that the distribution of the amount of rise for Judgment Requests is relatively concentrated and not scattered". Other prosodic features other than the rising pitch contour of interrogative sentences are not included because of scope and time limitation. *The developer of this system recommends further research in this area for*

Amharic language which is largely missing in much of the literature reviewed while doing this project. Other details on modeling interrogative sentences' prosody are given in *Section 3.2.3*.

Some pragmatic demonstrations are given for Amharic in order to show the applicability of other language conclusions given in *Section 3.2.3* on the difference of declarative and interrogative intonations. The easiest way to do so is to use waveforms. The statements "አበበ በሶ በላ።" (*abebe beso bella.*) and "አበበ በሶ በላ?" (*abebe beso bella?*) are used for comparison purpose. The figures below (*Figure 5.4* and *5.5*) show the difference of their waveforms.

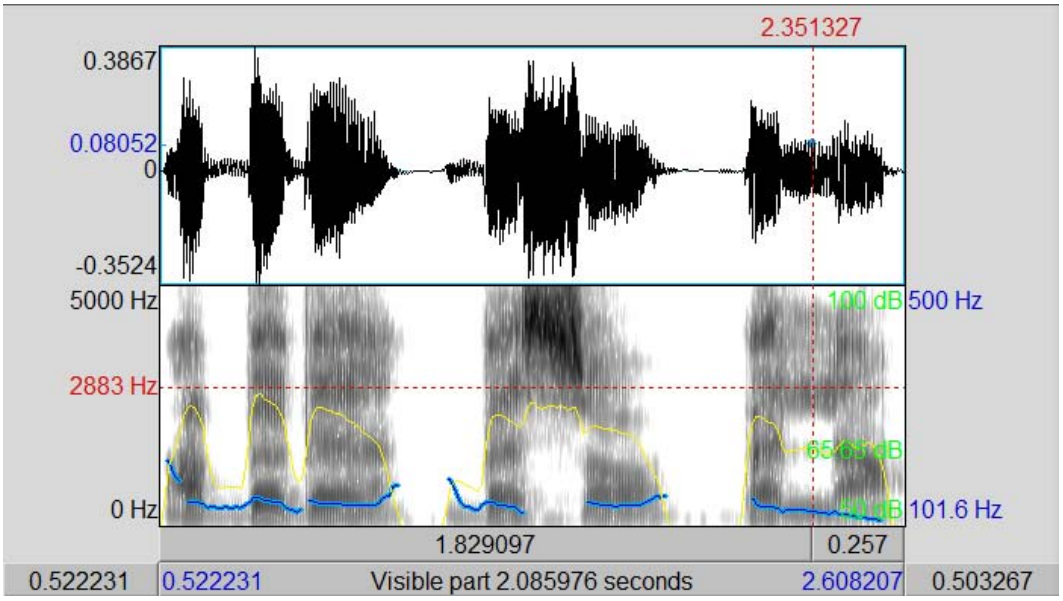


Figure 5.4 waveform of the declarative statement አበበ በሶ በላ። The light/yellow thin lines show the amplitude (intensity) and the black/blue bold lines on the spectrum show the pitch contour of the sound.

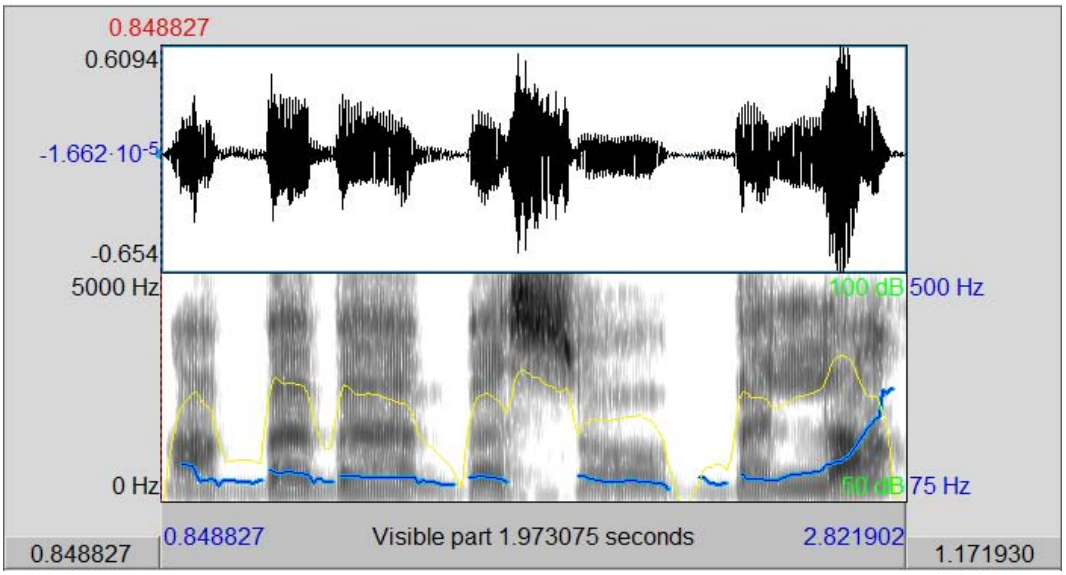


Figure 5.5 waveform of the interrogative statement አበበ በሶ በላ? The light/yellow thin lines show the amplitude (intensity) and the black/blue bold lines on the spectrum show the pitch contour of the sound.

It is evident from the *Figure 5.4* and *5.5* that there is a conspicuous difference among the sentences' final words. In the figures, it is clearly seen that the amplitude (intensity) and the pitch contour of the interrogative's sentence-final word is greater than that of the declarative. Though there seems to be also a little difference on other words of the statements, focus is given on the sentence finals. This is because reflecting all those changes requires more time, and research work which is enormously difficult to be handled by this project alone.

In the figures below (*Figure 5.6* and *5.7*) the focus is narrowed from a full sentence to a word in order to explore the real difference lying between the intonation of sentence-final words of declarative and interrogative statements.

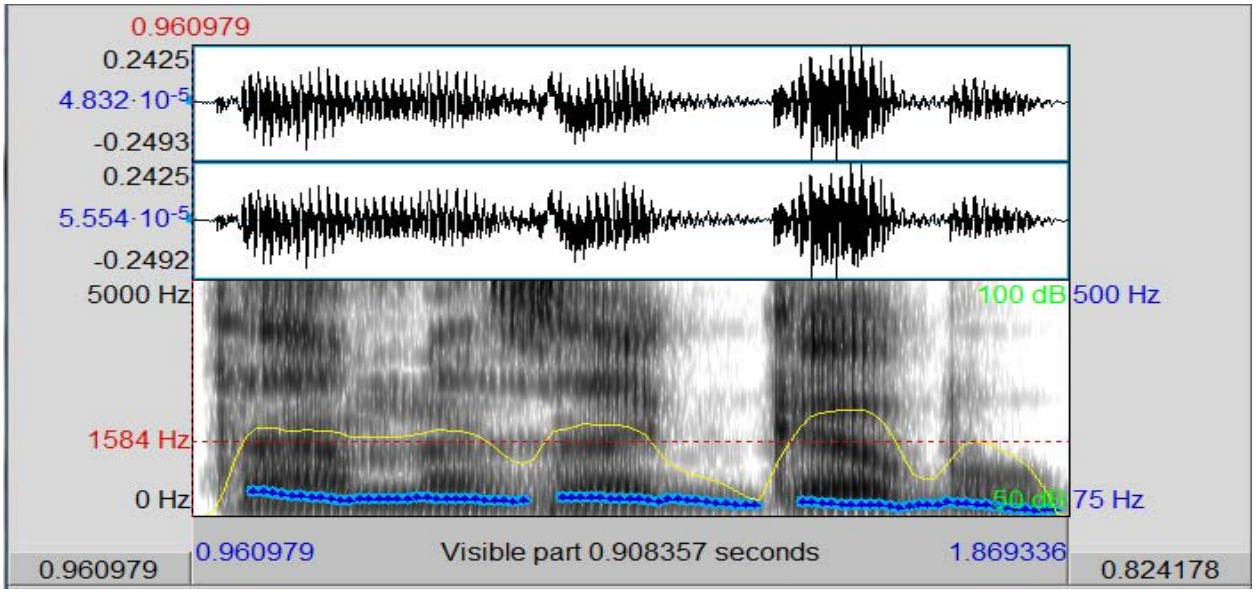


Figure 5.6 waveform of the word ተመዘገቡ (*temezeggebu*) in its declarative form. The light/yellow thin lines show the amplitude (intensity) and the black/blue bold lines on the spectrum show the pitch contour of the sound.

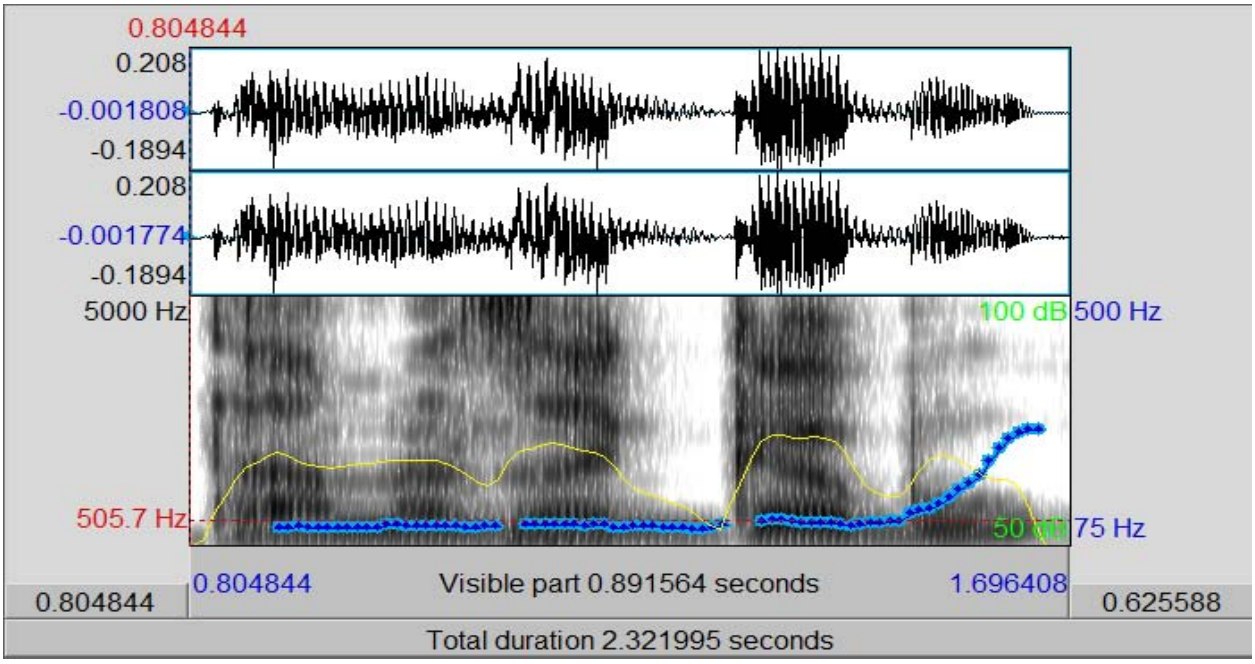
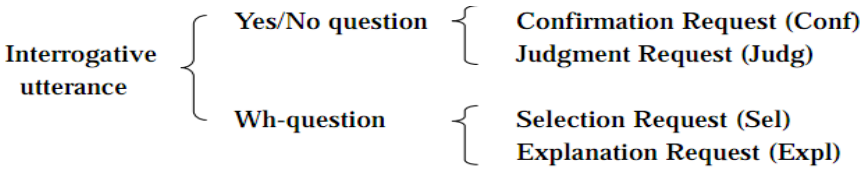


Figure 5.7 waveform of the word ተመዘገቡ (*temezeggebu*) in its interrogative form. The light/yellow thin lines show the amplitude (intensity) and the black/blue bold lines on the spectrum show the pitch contour of the sound.

Figures 5.6 and 5.7 can help us see that the difference is significantly focused on the word's final syllable/character, which is "ቡ" (*bu*) in this case. From this it can be said that the intonation of interrogative statements is reflected in their sentence-final words and especially in the last

syllable of the word. This is done by making those words render higher amplitude and pitch than their declarative counterparts.

Judgment Requests are the type of interrogative sentences that took the focus of this project. This work follows the division of interrogative utterances given in [32]. This category is chosen because their prosody can be easily represented by using the final phone of the sentence. Even though this same principle can be applied to both kinds of Yes/No sentences, the Confirmation Requests category is ignored because deciding between these two types of sentences requires knowledge on discourse of the text. The following division is adopted from [32].



The difference between judgment request and confirmation request sentences is that the speaker of the former does not have a prediction about the answer while the latter's has.

The following rules are the rules used in the project for interrogative sentence modeling:

1. Check if the word is a final word in a given interrogative sentence
2. If it is the sentence-final word and it ends with one of the suffixes listed in *Table 5.6*, select the interrogative sound unit version of the *penultimate* phoneme as the sound of the penultimate character in the word.
3. But if the word does not end with one of the suffixes listed in *Table 5.6*, select the interrogative sound unit version of the *final* phoneme as the sound of the final character in the word.

The only exception found so far for the rules given above is when the verb that comes at the end of the interrogative sentence ends with the consonant "ħ" (*k*). One example can be given to

clarify it, when the word “ሰበኩ” (*sebbeku*) comes as inflection of the verb “ሰ·በከ”. The word “ሰበኩ” (*sebbeku*) has the suffix “ኡ” (*u*) which makes the verb to be differently interpreted by supplying the subject “They” or “I”. In dealing with this ambiguity, the rules above represent only the prosody model of the sentence having the subject “I”. This problem requires disambiguation which actually is out of the scope of this work.

5.8 Integration with Microsoft Word 2007

It is customary that TTS tools are provided as plug-ins. Simply having its own form does not make it easy to use since many individuals do not write for synthesizing purpose but they want to synthesize what is already written. That is why add-in for Ms-Word 2007 is developed. In order to facilitate the option of synthesizing already written documents, options that allow the standalone application to read from text and word document (both in “.doc” and “.docx” extensions) files are also included. Microsoft Office is the application software used by much of the computer users in Ethiopia. Microsoft Word, in particular is the number one word processing application used by many PC users in Ethiopia. Many tools are developed that allow creating and editing Geez scripts in Microsoft Word application. Because of the aforementioned reasons, the Amharic TTS tool that came out from this project is integrated with Microsoft Word 2007 to ease its use.

5.9. Testing

This section presents the features tested, the test cases used and the testing approaches employed while testing the system. All testable functional and non-functional requirements specified in *Chapter 5* are tested. Different test cases were selected so that promised functionalities were proved to be working. The following test cases are the major test cases targeted at the functional requirements of the system:

1. Ability to pronounce words with declarative prosodic features
2. Ability to pronounce words with epenthetic cases
3. Ability to pronounce words with gemination cases
4. Ability to pronounce words with interrogative prosodic features

Integration with Ms-Word 2007, compatibility with any font style of the given Amharic text, and exception handling are some of the test cases geared towards the non-functional requirements.

The *bottom-up* approach was selected so that the test cases specified above are tested separately and then those features are joined step by step for testing until a test case is produced that combines all the sub test cases.

5.9.1 Test Results

All the test cases used to verify the non-functional requirements were successful. The tests performed on *epenthesis* were successful on all kinds of words except words with gemination cases. The test shows that the produced sound appears less intelligible when the consonant "ሀ" (*h*) comes at the end of a word and when two consonants (without the epenthetic vowel) come consecutively at the end of a word.

The tests made on *gemination* also proved that the rules given in *Section 5.4.2* are working. Most of the cases that these gemination rules were not successful are gemination patterns that also violate the general rules obtained from published materials referred in *Section 3.2.2*. However, the gemination rules happened to be successful on verbs with three or more radicals that geminate their penultimate radical, bi-radical verbs that geminate their final radical, and mono-radicals that start with the vowel "አ" (*a*). Such verbs were successfully geminated even if they have long suffixes as long as they are in their perfect form.

Tests performed to verify the effectiveness of prosody models of interrogative and declarative utterances were also successful. Even though one has to wait till the end of the sentence to identify it as interrogative or declarative, the intonation information found at the end really makes the sentence appear interrogative or declarative.

The test cases that were made on the combination of these test cases showed no decline or improvement of performance apart from those cases observed while doing the test cases separately.

5.10 Evaluation

Even though tests were performed as explained in *Section 5.9*, such systems require the evaluation of users by using techniques such as Mean Opinion Score (MOS) to evaluate intelligibility and naturalness of the system.

Ten individuals who are fluent speakers of the language were selected and were made to listen to sentences and words to evaluate their naturalness and intelligibility. The sentences and words were selected in such a way that they are appropriate for evaluating major features of the system. As it is described in the previous section, such features of the system include: its ability to model prosody of a Yes/No interrogative statements, declarative statements, its control of epenthesis and gemination. The way Arabic numerals and abbreviations are pronounced was also included in the evaluation process.

In doing so, eight sentences were prepared to evaluate whether prosodic features of interrogative and declarative utterances were properly represented. Four out of eight sentences are in declarative form, while the rest four are in their interrogative form. Nineteen words were selected to evaluate the system's control over gemination and epenthetic conditions. Two abbreviations

abbreviated using the dot "." symbol and the forward slash symbol "/" were also included in the evaluation. Finally, numerals, numerals preceded by one of the numeral prefix characters listed in *Section 5.3.2*, and numerals with decimal places were selected in the process.

A questionnaire was prepared and participants were made to evaluate the intelligibility and naturalness of the selected words/sentences. The questionnaire prepared is available in *Appendix B.* The participants were made to hear the utterances without first looking at the texts; but after hearing they are shown the texts so that they can compare what they have read against what they have heard and rank their opinion.

5.10.1 Result and Discussion

The following statements discuss the results obtained. They are divided into their appropriate category for analysis. *Table 5.7* shows the rating scales used for the MOS.

Table 5.7 – Rating scales

Rating	Very Good	Good	Fair	Poor	Bad
Value	5	4	3	2	1

Epenthesis

The results obtained from participants after making them hear words with epenthesis cases on their word initial, medial and final is shown in *Table 5.8.*

Table 5.8 – Evaluation on epenthesis

Naturalness	3.4
Intelligibility	3.3

Gemination

All the eight verb types described in *Section 3.2.2* were used in their stem form and by adding the suffixes to evaluate gemination. The result of the evaluation is shown in *Table 5.9.*

Table 5.9 – Evaluation on gemination

Naturalness	3.8
Intelligibility	3.6

Interrogative Modeling

Four interrogative sentences were given for the evaluation. Sentences that end with verbs with suffixes and without suffixes were evaluated and the result in *Table 5.10* is found.

Table 5.10 – Evaluation on interrogative sentences

Naturalness	3.8
Intelligibility	3.9

Declarative Modeling

Four declarative sentences were given for the evaluation. Like the interrogative, sentences with verbs that end with suffixes were included in the evaluation. *Table 5.11*shows the results.

Table 5.11 – Evaluation on declarative sentences

Naturalness	3.8
Intelligibility	3.5

Numerals

Table 5.12 summarizes the results obtained from evaluation of numerals with and without prefix and decimal places.

Table 5.12 – Evaluation on numerals

Naturalness	3.4
Intelligibility	3.2

Abbreviations

Table 5.13 summarizes the results obtained from evaluation of abbreviation abbreviated either with the dot or the forward slash symbols.

Table 5.13 – Evaluation on abbreviations

Naturalness	3.6
Intelligibility	3.7

Generally, the naturalness of the system was 3.63 and its intelligibility 3.53 which places the quality of the system just above fair and below good. The least scores are seen in epenthetic and numeral evaluations. The low scores of epenthesis are related to the low quality of the vowel-less consonants recorded whereas low scores on numerals is because of inappropriate duration of pauses between numbers.

6. Challenges, Conclusion and Recommendation

There are many challenges encountered during the design and development of the system. One of the major challenges was lack of literature in the area. There are very few researches that deal with many of the problems this project tried to solve. Grapheme-to-Phoneme conversion might be easy if it is only phonemes that developers are concerned about. However, this notion will be proved to be wrong when one engages in the activity. The big problem lies in the phoneme-to-phone conversion, since a phoneme has multiple competing phones (allophones) to give the proper pronunciation. The other challenge was building the sound units; starting from the electromagnetic noise of the PC that interferes during sound recording to deciding the number of allophones required for correctly representing a phoneme in different syntactic contexts was very challenging. The challenge of working with Amharic characters should also be mentioned here. Windows 7 and Microsoft Visual Studio 2010 have made available some of the flexibility needed to deal with this challenge.

One of the methods to easily produce intelligible and natural-like synthetic sound is using the unit selection concatenative approach. Two factors that determine the quality of a TTS system developed by this approach are: having as many allophones as possible and identifying contexts that determine allophonic variations. In this project, effort is shown to address these two factors. At least two allophonic variations for each phoneme in the Amharic alphabet are built; and epenthesis, gemination and interrogative prosody modeling rules to make appropriate selection of those variations from context are also used.

There are many missing functions that can be addressed by extending the functionalities already covered in this system and/or by adding new ones that are not covered in this system. Epenthesis, gemination and interrogative prosody modeling are some of the functionalities that require

extension. Epenthesis rules that deal with geminated consonant clusters, gemination rules that deal with Amharic words other than verbs in their perfect form and interrogative prosody modeling of the WH-Questions can and should be further researched and developed.

Some important features that determine the intelligibility and naturalness of the TTS system are not covered in this work. This is especially true with assimilation, which is the process of modifying a sound in order to make it more similar to some other sound in its neighbor. Further work is remaining to deal with assimilation processes such as labialization, palatalisation and nasalization.

One of the tools that ease the work of grapheme-to-phoneme process of a TTS system is the use of lexica which are very helpful to easily determine the pronunciation of a given word. In addition to dictionary lookups, since a dictionary cannot have all possible words of a given language, morphological analyzer is required especially for languages like Amharic which is morphologically so complex. Interested individuals may follow this way and help produce these tools that will definitely be used by many researchers and developers. Further research works can also be pursued in the area of rule-based grapheme-to-phoneme conversion apart from the use of lexica.

Some literatures reviewed during this undertaking show that further research is important to determine allophonic variations of phonemes according to their positions in a syllable; this is especially true for stops which take different allophonic forms when placed as onset and rhyme.

References

[1] Solomon Teferra Abate, Martha Yifiru Tachbelie, Wolfgang Menzel , “Amharic Speech Recognition: Past, Present and Future”, University of Hamburg, Department of Informatics, natural language systems division.

[2] Wikipedia. Speech Synthesis. Retrieved from http://en.wikipedia.org/wiki/Speech_Synthesis on 25 March 2010.

[3] Tadesse Anberbir and Tomio Takara, “Development of an Amharic Text-to-Speech System Using Cepstral Method”, Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages AfLaT2009, pages 46–52, Athens, Greece, 31 March 2009

[4] Sebsibe H/Mariam, S P Kishore, Alan W Black, Rohit Kumar, and Rajeev Sangal. 2004. *Unit Selection Voice for Amharic Using Festvox*, 5th ISCA Speech Synthesis Workshop, Pittsburgh.

[5] Christian Benoît, “Speech Synthesis: Present and Future”, European Studies in Phonetics & Speech Communication, ICP, Grenoble, France, 1997.

[6] S.S.Agrawal, Workshop on Spoken Language Processing, TIFR, Mumbai, January 9-11, 2003.

[7] Ann K. Syrdal, Gregor Mohler, Kurt Dusterho, Alistair Conkie and Alan W. Black, “Three Methods of Intonation Modeling”, AT&T Labs – Research, Florham Park, NJ, USA, 1999

[8] Sami Lemmetty, “Review of Speech Synthesis Technology”, Helsinki University of Technology, March 30, 1999.

[9] Juergen Schroeter, “Text to-Speech (TTS) Synthesis”, AT&T Laboratories, 2005

[10] Demeke Asres, Automatic Prosody Manipulation Using LPC-Residual Technique, MSc Thesis, Faculty of Informatics, Addis Ababa University, Ethiopia, 2007.

[11] Bereket Kasaye, Developing A Speech Synthesizer For Amharic Language Using Hidden Markov Model, MSc Thesis, Faculty of Informatics, Addis Ababa University, Ethiopia, 2008.

[12] Mark Beutnagel and Alistair Conkie, “Interaction of Units in a Unit Selection Database”, AT&T Labs-Research, Florham Park, NJ, USA. Retrieved from: *http://www.research.att.com/projects/tts* on March 1, 2011.

[13] Matthias Jilka and Ann Syrdal, “The AT&T German Text-to-Speech System: Realistic Linguistic Description,” in *Proceedings of ICSLP*. 2002, Denver.

[14] Yeon-Jun Kim, Ann K. Syrdal, Alistair D. Conkie, Mark C. Beutnagel, “Phonetically Enriched Labeling in Unit Selection TTS Synthesis,” INTERSPEECH, Pittsburgh, Pennsylvania, 2006.

[15] Yeon-Jun Kim, Ann K. Syrdal, and Matthias Jilka, “Improving TTS by Higher Agreement between Predicted versus Observed Pronunciations,” in *Proceeding of The 5th ISCA ITRW on Speech Synthesis*, 2004.

[16] Edinburgh Speech Tools. Introduction to the Edinburgh Speech Tools. Retrieved from: *http://festvox.org/docs/speech_tools-1.2.0/c23.htm* on February 27, 2011.

[17] Black A., Taylor P. and Richard Caley, “Festival Speech Synthesis System: System Documentation (1.4)”, University of Edinburgh, 2002.

[18] Alan W Black and Kevin A. Lenzo, “Flite: a small, fast speech synthesis engine”, System documentation Edition 1.4, for Flite version 1.4, 4th January 2009.

[19] Sun Microsystems, “Java Speech API Programmer’s Guide: Introduction”. Retrieved from: *http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide/Introducion.html* on February 25, 2010.

[20] Redmond Pie. “Converting Text to Speech in a C# WPF Application”. Retrieved from: *http://www.redmondpie.com/Converting Text to Speech in a C%23 WPF Application _ Redmond Pie.htm* on March 2, 2011.

[21] Microsoft. “Speech API Overview”. Retrieved from: *http://msdn.microsoft.com/en-us/library/ms720151%28v=VS.85%29.aspx#SpeechAPIOverview* on March 10, 2011.

[22] Yibeltal Tefera (2008). “Formant-Based Speech Synthesis: A Case of Amharic Words”. MSc Project, Faculty of Informatics, Addis Ababa University, Ethiopia.

[23] Nadew Tademe (2008). “Formant-based speech synthesis for Amharic vowels”. MSc Thesis, Faculty of Informatics, Addis Ababa University, Ethiopia.

[24] Solomon Gizaw, “Multiple Pronunciation Model for Amharic Speech Recognition System”, MSc Thesis, Faculty of Informatics, Addis Ababa University, Ethiopia, 2008.

[25] Habtamu Taye, “Amharic Concatenative Text-to-Speech Synthesis System Using Syllabic unit”, MSc Project, Faculty of Informatics, Addis Ababa University, Ethiopia, 2007.

[26] Tadesse Anberber, Michael Gasser, Tomio Takara and Kim Dong Yoon, “Grapheme-to-Phoneme Conversion for Amharic Text-to-Speech System”, *Conference on Human Language Technology for Development*, Alexandria, Egypt, (2011).

[27] Michael Gasser, “HornMorpho: a system for morphological analysis and generation of Amharic, Oromo, and Tigrinya words”, *Conference on Human Language Technology for Development*, Alexandria, Egypt, (2011).

[28] Janet C. E. Watson, “The Phonology and Morphology of Arabic”, Oxford University Press Inc., New York, 2002/2007.

[29] Mulugeta Seyoum, “The syllable Structure and Syllabification in Amharic,” Masters of philosophy in general linguistic thesis, Department of Linguistics, Trondheim, Norway, 2001.

[30] Getahun Amare, “ዘመናዊ የአማርኛ ሰዋሰው በቀላል አቀራረብ”, Alpha publishers, Addis Ababa, 2010.

[31] Anne and Klaus Wedekind, “Amharic stress (beat) rules of linguists, poets and singers: Which beat rules beat which?”, the 11th International Conference of Ethiopian Studies Addis Ababa, 1991.

[32] Mieko Takada and Takako Ayusawa, “The Intonation of Interrogative Utterances in the Japanese Dialogs — Analysis of the TUFS Language Module”,

[33] Ralf Kompe, “Prosody Speech Understanding Systems”, Subseries of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1997.

[34] Yuji Kawaguchi, Selim Yilmaz and Arsun Uras Yilmaz, “Intonation Patterns of Turkish Interrogatives”,

[35] Yoshiko Fujisaki, “Sonority and Its Role for Syllabification”, Department of Humanities, Natural Language, 1995.

[36] Joyce McDonough, “The Prosody of Interrogative and Focus Constructions in Navajo”, Department of Linguistics and Center for the Language Sciences, University of Rochester, Amsterdam, 2002.

[37] Jun Xu, “The Prosody of Interrogatives at Transition-Relevance Places in Mandarin Chinese Conversation”, Master of Research Thesis, University of Nottingham, March 2008

[38] Amsalu Aklilu, “Amharic – English Dictionary”, Mank Usa publishing, Addis Ababa, 2010

[39] Jochen Trommer , “A Feature-Geometric Approach to Amharic Verb Classes”, Institute of Linguistics, University of Leipzig.

[40] Francesco D’Este, and Erwin Bakker, “Articulatory Speech Synthesis with Parallel Multi-Objective Genetic Algorithms”, LIACS Medialab, Leiden University.

Appendix A: Notation used for representing Amharic text in the document

Amharic Characters	Notation
ሀ/ሐ/ኀ	h
ለ	l
መ	m
ሰ/ሠ	s
ረ	r
ሸ	sh
ቀ	q
በ	b
ተ	t
ቸ	c
ነ	n
ኘ	n’
ዐ	?
ከ	k
ወ	w
ዘ	z
ዠ	z’
የ	j
ደ	d
ገ	g
ጀ	g’
ጠ	t’
ጨ	c’
ጸ/ፀ	s’
ጰ	p’
ፍ	f’
ፕ	p
ቨ	v
ኧ	e
አ/ኣ	a
ኡ	u
ኢ	i
ኤ	ie
እ	ix
ኦ	o
ኳ	kwa
ጓ	gwa
ቋ	qua

Appendix B: Questionnaire used during the MOS evaluation

Addis Ababa University

Schools of Graduate Studies

Department of Computer Science

The purpose of this questionnaire is to collect your opinion regarding the Amharic synthesizer system from which you will listen as it pronounces sentences, words, numbers and abbreviations. After listening to the system, you will be shown/given the written form of the utterances you have heard. You will give results that you think reflect your opinion on the naturalness and intelligibility of the system by comparing the utterances with their written forms and your perception. Since the results you give are very important for the conclusions we make on the system, you are kindly requested to rank your opinion carefully and honestly.

***Description**: 5 – Very Good, 4 – Good, 3 – Fair, 2 – Poor, and 1 – Bad)*

EVALUATION CASES		**RANK**				
DECLARATIVE	NATURALNESS	1	2	3	4	5
	INTELLIGIBILITY	1	2	3	4	5
INTERROGATIVE	NATURALNESS	1	2	3	4	5
	INTELLIGIBILITY	1	2	3	4	5
EPENTHESIS	NATURALNESS	1	2	3	4	5
	INTELLIGIBILITY	1	2	3	4	5
GEMINATION	NATURALNESS	1	2	3	4	5
	INTELLIGIBILITY	1	2	3	4	5
NUMBERS	NATURALNESS	1	2	3	4	5
	INTELLIGIBILITY	1	2	3	4	5
ABBREVIATION	NATURALNESS	1	2	3	4	5
	INTELLIGIBILITY	1	2	3	4	5

Declaration

I, the undersigned, declare that this project is my original work and has not been presented for a degree in any other university, and that all source of materials used for the project have been duly acknowledged.

Declared by:

Name: _____

Signature: _____

Date: _____

Confirmed by advisor:

Name: _____

Signature: _____

Date: _____

Place and date of submission: Addis Ababa, June 2011.