2009

WebVoice: Speech Access to Traditional Web Content for Blind Users

Shahriar Chandon
University of Windsor

WebVoice: Speech Access to Traditional Web Content for Blind Users

by

Shahriar Chandon

A Thesis

Submitted to the Faculty of Graduate Studies

through Computer Science

in Partial Fulfillment of the Requirements for

the Degree of Master of Science at the

University of Windsor.

Windsor, Ontario, Canada.

2009

WebVoice: Speech Access to Traditional Web Content for Blind Users

by

Shahriar Chandon

APPROVED BY:

Dr. R. Caron, External Reader

Department of Mathematics and Statistics

Dr. J. Morrissey, Internal Reader

School of Computer Science

Dr. R. Frost, Advisor

School of Computer Science

Dr. X. Yuan, Chair of Defense

School of Computer Science

April 28, 2009

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Traditional web content and navigation features are made available to blind users by converting a webpage into a speech enabled X+V application, which allows blind users to follow the links present in a web page via speech commands. Also the application can read the different paragraphs and search for a word. This X+V application runs on the Opera browser.

Dedication

I would like to dedicate this thesis to my family.

Acknowledgement

I acknowledge Dr. Richard Frost, my advisor, without whom this thesis would not have materialized. I also wish to express my gratitude to the other members of my Master's thesis committee.

Contents

List of Tables

List of Figures

Chapter

I. Thesis Statement

Section 1: The Thesis Statement

It is possible to dynamically generate recognition grammars to facilitate non-visual navigation of the conventional web.

Section 2: Why is the Thesis Important?

The thesis is important for several reasons. First of all blind users will be able to navigate the conventional web just as easily as the sighted users. Secondly, having speech only access to the traditional web can mean another advance in Internet technology.

Section 3: Why is the thesis not-trivial?

- No one has dynamically generated recognition grammar before.
- Grammar based speech recognition began recently.
- There is a need to decide on modular structure of grammar for good recognition accuracy.

Section 4: What have we done to prove the thesis?

- We have built a prototype.
- We have Analyzed the advantages and shortcomings of the prototype.

II. Background, Problem Definition and Notations

Section 1: Background

Speech has been the prime mode of interaction for humans from the beginning of our civilized existence. It has been through the use of speech that we have communicated information, expressed our emotions and most of all given all our thoughts a structure. I dare say, without speech the advancement of technology would not have been possible. It should, therefore, be recognized as a vital element of humanity.

Speech in general belongs to a certain language. A very broad definition of language can be a set of rules that allows the formation of coherent and independent unit of thought that can be expressed and understood by other speakers of the same language. That is to say our thoughts must follow a set of rules in order to be expressed. The basis of the rules began with visual elements and cues. In addition, language also integrated itself with the visual elements of our surrounding nature. Thus language itself became a tool for sighted persons to progress via social interaction and interchange of ideas. This integration of visual elements has two major disadvantages. First, blind people were always left behind in human society particularly because of their lack in comprehending visual references. Thus we are losing an important part of our society just because our communication medium is filled with visual references. Secondly, we need to explore a language that is free of visual references. We need to explore the new thought patterns that would develop if such a language were in existence. It is important because we need to explore the possibilities that such a language would open. If such kind of language were present then it might give us a new direction in thinking and expressing ideas. After all, the entire humanity heavily depends on our ability to think and to express ideas.

In this thesis we have tried to achieve a very small step toward that goal. In today's world, the Internet is very popular and has become a major mode of exploration of ideas and interchange of thoughts. It is however, again limited to sighted users. We shall try to make it possible that with the help of speech a blind person should be able to browse the Internet.

Section 2: Problem Definition

All computer systems consist of a visual interface. The visual interface has been the prime mode of interaction with a computer until very recently when speech or voice activated access also

gained some popularity. However, due to the fact that speech is slower than sight, speech access has never really gained a great deal of use as visual interfaces have.

The traditional web has gone far with respect to delivering information to sighted users. In addition to that it has also become a popular way of delivering multimedia content. The traditional web is very easy to use for a sighted user. It allows many different GUI elements and visual cues to be interacted upon. Information is often categorized in a very orderly fashion allowing users to navigate easily to a certain piece of information. Querying or searching is also another popular method of navigation.

None of the above, however, is accessible to blind users. There have been many different suggested methods of interaction for blind users even before the emergence of The Internet, such as speech-based interfaces, auditory interfaces, tactile interfaces etc. Some methods use a tactile interface as input and auditory interface as output. Other methods use speech as both input and output.

If we consider, speech as the primary medium of interaction between a user and a computer then we need natural language processors and then program the computer in such way that the application allows human-like conversation with the computer to a certain extent.

The Internet has a lot of content and we have highly-efficient search engines serving that content. Unfortunately almost all of the content is graphical. Traditional screen readers can read it off the screen but navigation through hyperlinks is quite difficult. The hyperlinks are a key element of the Internet. Generally, hyperlinks are displayed in a web page and through mouse or key board interaction a sighted user can follow a link. Another feature of visual presentation is the ability to find a word in a paragraph or a page. A sighted user can easily find a certain word or phrase in a web page through the search feature of the browser where he or she can type the word and depending upon the existence of the typed word the browser will position itself at the first occurrence of the typed word.

It is felt that it will be an important step towards bringing the Internet to blind users, if there is a way through which, a web page is converted into some speech based program, incorporating the two key features discussed above, where the user can interact through speech. For example, the user might want to know how many paragraphs there are in a certain web page, or how many links are present and, most importantly, the user might want to follow a certain link and be

presented with similar speech based program where all of the speech-enabled features are made available again. It is possible that through this approach a blind user could keep on browsing the Internet following link after link.

Section 3: Terminology

GUI: Graphical User Interface.

Hearcons: Auditory equivalent of GUI Icons.

Haptic Device: Touch sensory device.

Hyperbolic Browsers: (Smith, 2004) "Browsers that employ a fisheye technique, whereby the user can see details of specific parts of the hierarchy at the same time as viewing the general content of the full hierarchy."

Tiflo: Means "*Blind*" in *Greek*.

S.O.L.A.: (Arons, 1997) "Conceptually, the SOLA method consists of shifting the beginning of a new speech segment over the end of the preceding segment to find the point of highest cross correlation (Maximum Similarity). The overlapping frames are averaged, or smoothed together, as in the sampling method. SOLA can be considered a type of selective sampling that effectively removes entire pitch periods. S.O.L.A. produces the best-quality speech for a computationally efficient time domain technique."

III. Related Work

In order to determine the current capabilities of existing non-visual web browsers, we conducted a survey of relevant published material. We identified 56 relevant papers. A detailed discussion of these papers is given in Appendix I. In the following we summarize the results of our survey, by categorizing the papers according to a number of questions which we developed concerning the focus of the papers.

Section 1: Survey Categories and Questions

To determine what should be the categories we have focused on several attributes of each paper. These attributes are best expressed by the following questions.

1. Does the paper discuss computer interfaces for blind users?
2. Does it use (or suggest to use) speech for both input and output?
3. Does it allow (or suggest to allow) skimming through the output speech?
4. Does it allow (or suggest to allow) content search?
5. Does it use (or suggest to use) speech as output?
6. Is the system (or suggest to be) dedicated for blind users only?
7. Is the system (or suggest to be) multi-modal?
8. Is the user interface developed for Internet browsing (or browsing related activity)?
9. Does the system transform (or suggest to transform) an existing GUI for blind users?
10. Does it use (or suggest to use) non-speech sounds as output?
11. Is the user interface developed for generalized computer operations?
12. Is the user interface developed for operating a mechanical device?
13. Does the paper discuss about the interface of some mechanical device for blind users?
14. Does the system use (or suggest to use) tactile hardware as input devices?

15. Does the system use (or suggests to use) dynamic recognition grammar generation?

Table 1: Survey questions

Section 2: Survey Samples

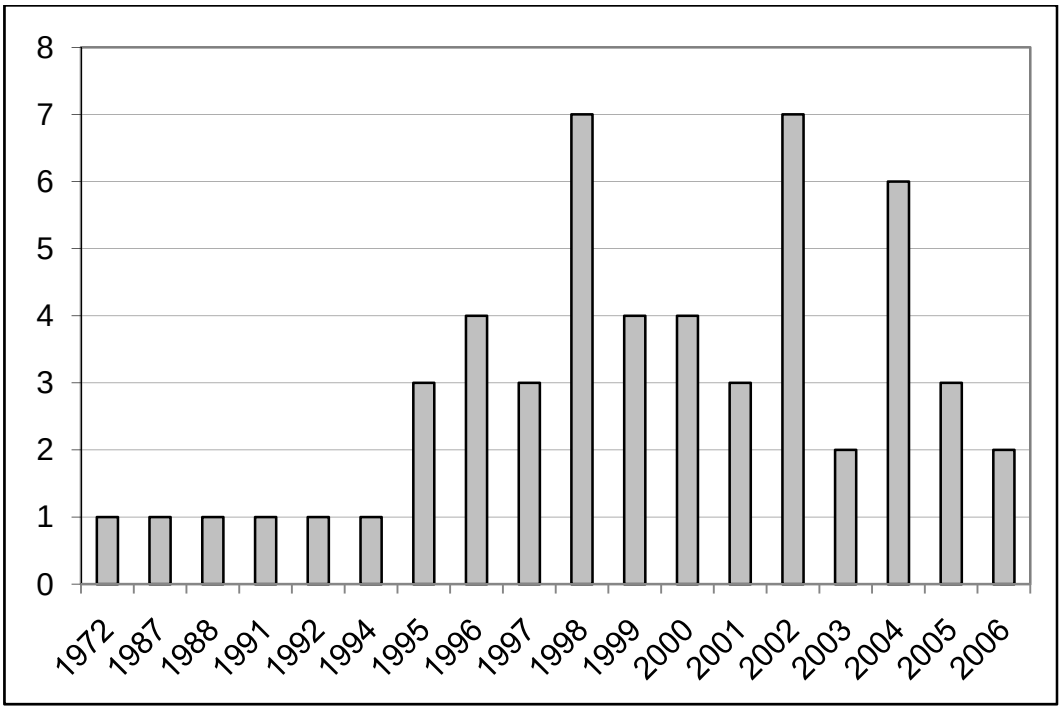

Figure 1: Dispersion of survey samples.

The Figure above shows the dispersion of survey samples. It is obvious that we have focused on papers that are published after 1994. The reason for that is the Internet evolved to the form that we know today after that year.

Section 3: Survey Result

In the next step all 15 questions were applied to the 50 papers and the following table was prepared. The table and graph following the table, summarize the research that has been done with respect to the questions.

40 **(80%)**	**1.** Does the paper discuss computer interfaces for blind users?
40 **(80%)**	**5.** Does it use (or suggest to use) speech as output?

27 **(54%)**	**9.** Does the system transform (or suggest to transform) an existing GUI for blind users?
26 **(52%)**	**10.** Does it use (or suggest to use) non-speech sounds as output?
26 **(52%)**	**11.** Is the user interface developed for generalized computer operations?
25 **(50%)**	**6.** Is the system (or suggest to be) dedicated for blind users only?
23 **(46%)**	**2.** Does it use (or suggest to use) speech for both input and output?
21 **(42%)**	**14.** Does the system use (or suggest to use) tactile hardware as input devices?
19 **(38%)**	**3.** Does it allow (or suggest to allow) skimming through the output speech?
18 **(36%)**	**7.** Is the system (or suggest to be) multi-modal?
18 **(36%)**	**8.** Is the user interface developed for Internet browsing (or browsing related activity)?
17 **(34%)**	**4.** Does it allow (or suggest to allow) content search?
7 **(14%)**	**13.** Does the paper discuss about the interface of some mechanical device for blind users?
6 **(12%)**	**12.** Does the paper focus on output interface only?
3 (6%)	**15.** Does the system use (or suggests to use) dynamic recognition grammar generation?

Table 2: Survey result

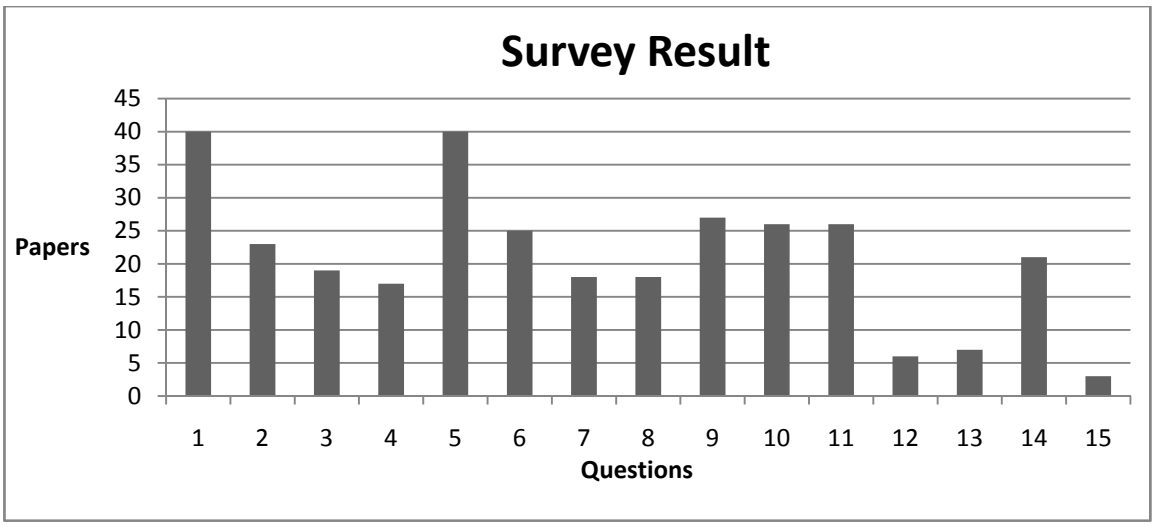

Figure 2: Summary of survey results.

The complete survey description is found in Appendix I. Very few of the papers that we have reviewed create dynamic grammars.

IV. The New Idea

The primary contribution of this thesis work is a new idea of dynamically-generating speech-recognition grammars to enable reference to html page **content** during the non-visual browsing of conventional web pages. For example, in addition to saying "read paragraph two" or "follow link three", our idea allows the user to say "read the paragraph containing the word "economy"", and "follow the link "History of Panama"".

Another contribution is that we have built a prototype to demonstrate the viability of our new idea, and have analyzed the prototype by comparing its capabilities with other non-visual web browsers, and have also determined the computational complexity of generating and using dynamic grammars.

V. The Prototype

We propose an architecture which will take a traditional web page and generate an X+V page and a Grammar allowing the user to browse through web content using speech commands. X+V stands for XHTML+Voice profile and it supports speech synthesis, speech dialogs, command and control and speech recognition grammars. It is based on an event-driven model and part of the W3C Speech Interface Framework.

Ideally the transformation process should transform all of the related html content in such a manner that the content is readily available as a speech output when a speech command is issued. However, in this thesis, we shall only focus on the transformation of the main features that are available to a sighted user. Namely, content and links. Now, when it comes to content, we have many types of content, ranging from basic text to multimedia and images. In our thesis we shall not concern ourselves with the transformation of multimedia and images rather we shall focus on transforming text content. In terms of the transformation of links, we shall only transform the text and image links. Graphical interfaces allow a certain word to be searched inside a web page, therefore, we shall try to provide the same feature in our speech-only access where a user will be able to search for a particular word in the transformed content. With the above approach we shall try to show that it is possible to allow speech-only interaction to the traditional web almost in the same interactive fashion as a graphical interface.

At the beginning, the Grammar will contain a basic command set. Through the basic command set a user may visit a link. The new page following the link will then be converted into an X+V page. The new Grammar will also be generated which will continue to support the basic command set. The following diagram explains the idea in more detail:

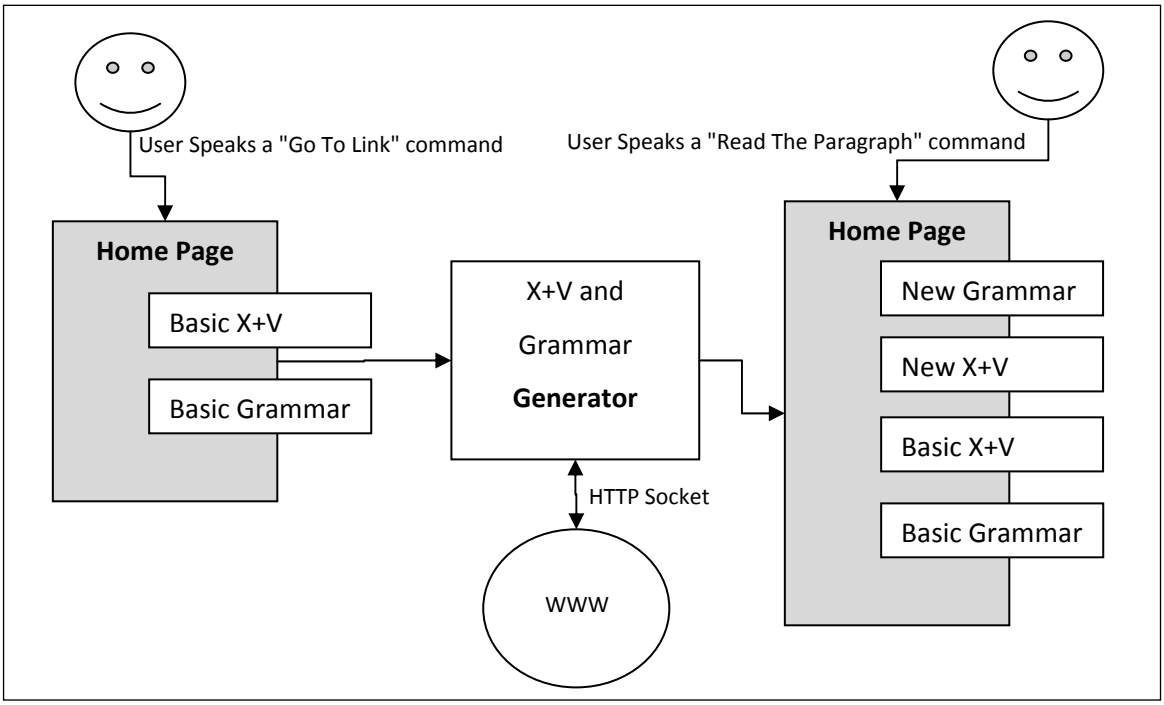

Figure 3: General WebVoice architecture.

We call the above architecture "WebVoice". Henceforth in this document we shall refer to this architecture as the general WebVoice architecture.

From the above diagram, we can see that a user may visit a link by using the "Go to Link" command found in the basic command set, which is composed of basic X+V and basic Grammar. Once the new link load request is sent to the Generator, it reads the remote content through an HTTP socket and creates a new X+V and Grammar that is based on the extracted content. At that point the user can ask the system to count how many paragraphs, read a paragraph, find a paragraph containing some word and so on. In addition to the commands involving the newly loaded page the user may still issue the basic voice commands, which enables continuous navigation through different pages.

Below is a general description of the architecture of the **"X+V and Grammar Generator"** as seen in the figure 14.

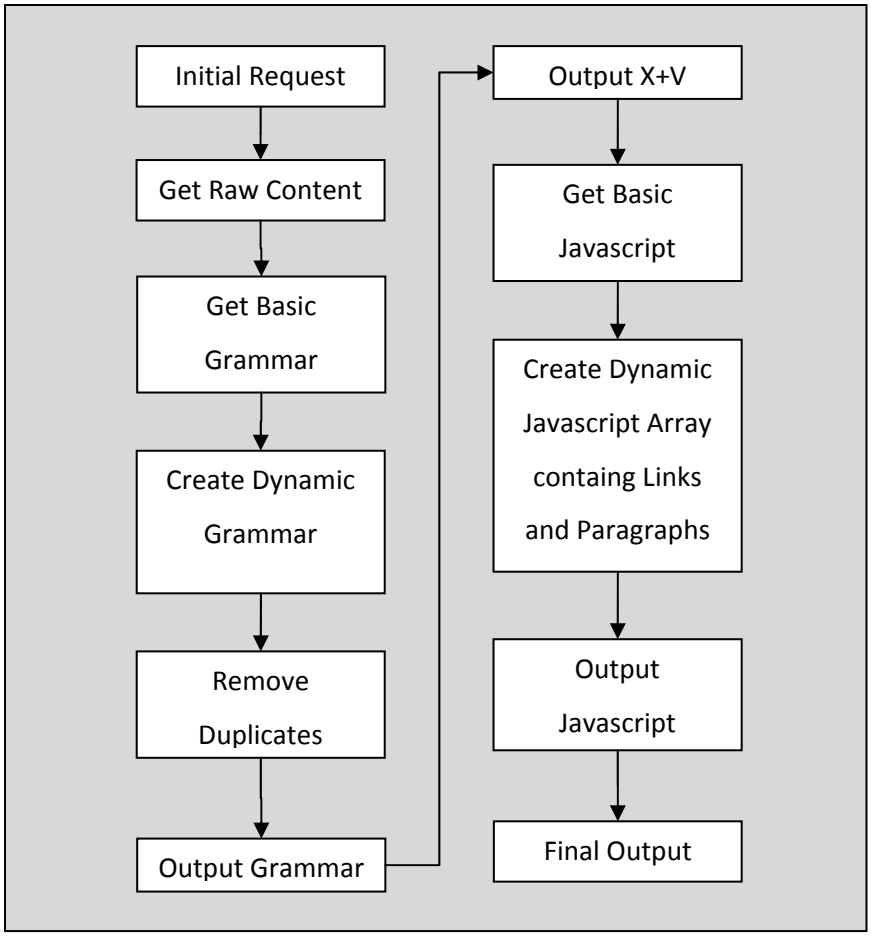

Figure 4: The architecture of the X+V And Grammar Generator module.

The above solution targets delivering content via speech commands. No GUI interaction is necessary. The output speech will provide information regarding how many paragraphs, links or other items that are present in the page. The user would be able to interact by issuing voice commands naming a target paragraph or link. This type of interaction does not involve transformation of GUI elements into auditory interfaces, rather the meta information of the GUI elements are spoken out loud so that they can be targeted later by the user in a speech command.

In this architecture, the problem that we faced first was to generate the client layer properly. In our client layer we have 3 major components:

1. The recognizer and recognition grammar.

2. The event handlers or X+V handlers.

3. The JavaScript helpers for the handler.

The recognizer and recognition grammar contains certain JSGF grammar rules that describe a set of commands that the user can utter. When a user command is recognized it is then processed by the X+V script.

The event handlers or X+V handlers fire different events based on different input types when they receive a recognized input. These events are then handled by the event handler portion of the X+V script. The event handlers with the help of the JavaScript Handlers prepare and send the actual output to the user.

The JavaScript helpers for the handler are the set of functions that help the X+V handlers to lookup a resource or execute a client command. For example, the transfer from one page to another is actually done by the JavaScript helpers.

All three of these components should interact properly when the user is looking for some web content. Our proposed architecture, in addition to parsing, also ensures proper generation of these components. Once they are properly generated the execution of a certain recognized command can be supported by the appropriate X+V event and, if required, by the JavaScript helpers. The X+V and JavaScript Helpers will access a generated resource in order to create the proper output. The resource in this case constitutes both JavaScript arrays and X+V resources that are created after parsing of the HTML page.

In the following two diagrams we shall try to describe what our parser does by showing the initial state and the extended state of the client layer.

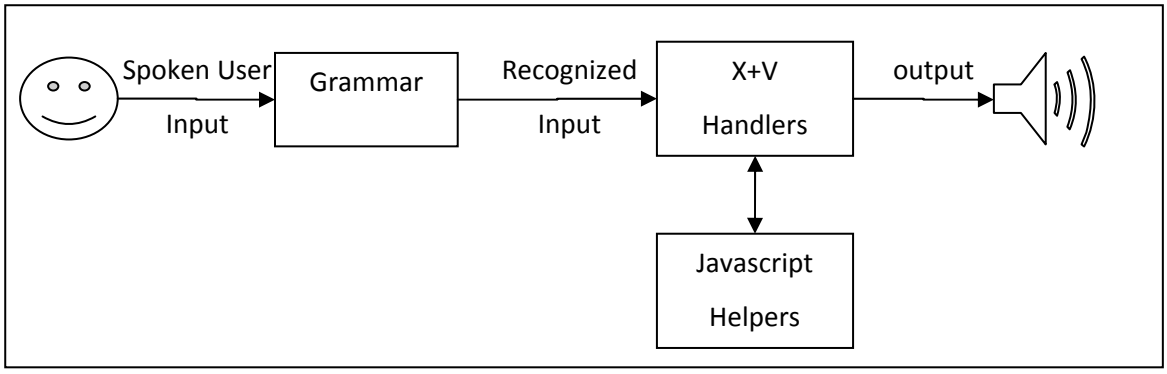

Figure 5: Initial State of the Client Layer

In the initial state the client layer consists of basic commands and support for the execution of the basic commands. When the user follows one of the default links or default urls and loads a new page, we reach what we call, "the extended state".

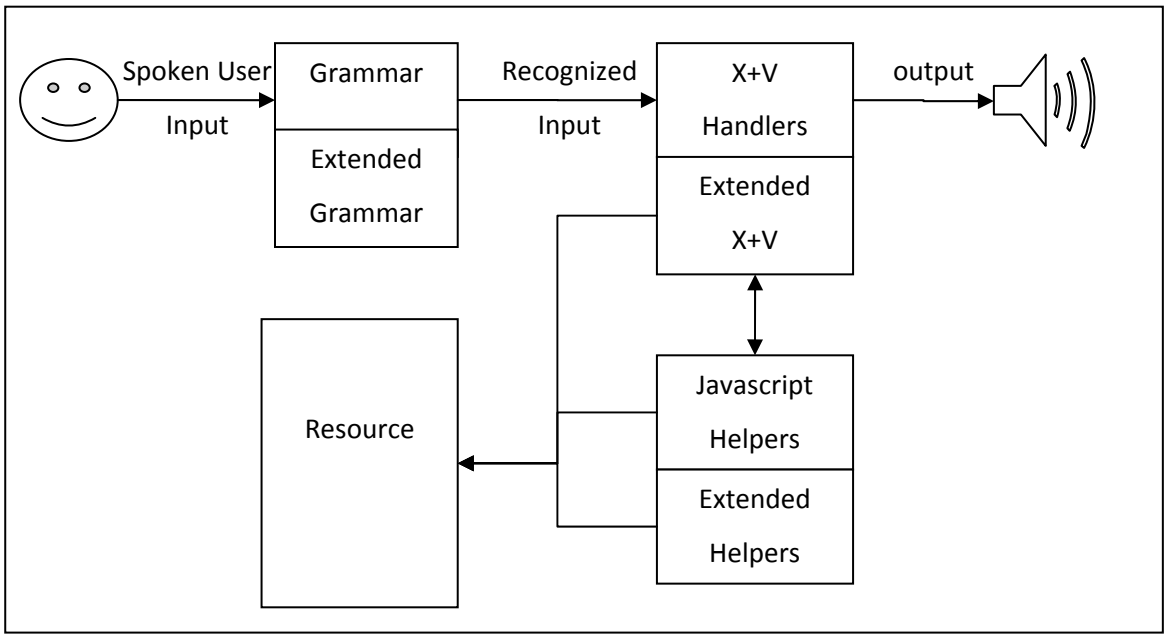

Figure 6: Extended State of the Client Layer

Once we arrive at the extended state, the user may choose to continue visiting other pages or may wish to stop the application by issuing a shut down command. If the user chooses to visit other pages, the client layer will transform itself with a different set of resource. So in a sense, the user will still be in the extended state of the client layer.

We feel that the proposed architecture addresses the main problem faced by speech-only interaction to the traditional web. It is therefore implemented using available technology, which we discuss in the next section.

Section 1: Requirements Analysis

We have used USE-CASE Diagrams to define the requirements specified in the next sub-section. To view the detailed USE-CASE analysis, please see Appendix II.

1.1. Requirement Specification

After observing the Use-Cases in the previous section, we have come up with the following set of requirements for our application.

1.1.1. Functional Requirements

1. Initially, once the home page is loaded, the user may request one of the following set of tasks to be performed:
    1.1. Visit one of a set of URLs.
        1.1.1.The command structure should be like the following:
            1.1.1.1. Go to “someurl.com”; where someurl.com would be a valid predefined URL.
    1.2. Visit a default link.
        1.2.1.The command structure should be like the following:
            1.2.1.1. Load Default Link “Some Number”. For example: Load Default Link 1 or Load Default Link 4.
    1.3. Shut down.
    1.4. Read the Help text.
    1.5. Reset to initial prompt if no shut down command is found.
2. When a new page is loaded the system will notify the user of the load completion event.
3. After the user is notified of the load event of a new page, the user may perform the following:
    3.1. Ask how many links in the page.
    3.2. Ask how many paragraphs in the page.
    3.3. Ask to read a single paragraph by issuing a command.
        3.3.1.The command structure should be like the following:
            3.3.1.1. Read Paragraph “Some N Number”. For example: Read Paragraph 1 or Read Paragraph 10.
        3.3.2.If the mentioned paragraph does not exist then the user should be notified.
        3.3.3.If the paragraph exists then the system should read the paragraph.
    3.4. Ask to read a single link by issuing a command.
        3.4.1.The command structure should be like the following:
            3.4.1.1. Read Link “Some N Number”. For example: Read Link 2 or Read Link 73.
        3.4.2.If the mentioned link does not exist then the user should be notified.
        3.4.3.If the link exists then the system should read the link label and the link URL.
    3.5. Ask to read all paragraphs.
    3.6. May wish to find a paragraph containing a certain word by issuing a command.
        3.6.1.The command structure should be like the following:

3.6.1.1. Find the Paragraph with the word XYZ; where XYZ stands for some word.

3.6.2. If a matching paragraph is found then the system will indicate the paragraph number.

3.6.3. If no such paragraph is found then the user should be notified.

3.7. May wish to find a link containing a certain word by issuing a command.

3.7.1. The command structure should be like the following:

3.7.1.1. Find the Link with the word XYZ; where XYZ stands for some word.

3.7.2. If a single link's link label matches the searched word then the system will indicate the link number.

3.7.3. If the searched word matches with less than 20 link's link label then all the numbers would be spoken out.

3.7.4. If the searched word matches with more than 20 link's link label then the system will not read out the matching link numbers, instead it will indicate the total number of links.

3.7.5. If no such link is found then the user should be notified.

3.8. May wish to visit a link by issuing a command.

3.8.1. The command structure should be like the following:

3.8.1.1. Load Link "Some Number". For example: Load Link 1 or Load Link 4.

3.9. Shut down the application.

3.10. Reset to initial prompt if no shut down command is found.

1.1.2. Non-Functional Requirement

1. Both input and output will be speech.

Section 2: Architectural Design

Please see Appendix III for the detailed assumptions and the architectural design.

Section 3: Coding

Please see Appendix V for all of the related codes.

VI. Example Scenarios/Demonstration of the Prototype

We shall now provide a brief demonstration of our developed prototype. For a detailed USE-CASE based scenario demonstration please see Appendix IV.

The user opens the program and says "Go To PHP.Net". The program loads PHP.net website and notifies the user via speech that it has finished loading.

Then the user says, "How Many Paragraphs?"

WebVoice: "10 Paragraphs".

USER: "Find the paragraphs with word 'ABOUT' "

WebVoice: I have found 3 paragraphs. Paragraph 1, 4 and 10 contains the word 'ABOUT'.

USER: "Read Paragraph 4".

WebVoice: "Reading- ...-Finished Reading Paragraph"

USER: "How many links?"

WebVoice: I have 20 links.

USER: Find the Link with the word 'Contact'

WebVoice: Link 3 and 32 contains the Word 'Contact'

USER: Read Link 32.

WebVoice: Reading ...finished reading.

USER: go to link 32.

VII. Analysis of the Prototype

Section 1: Brief Discussion of the Prototype Functionalities

The prototype we have developed, as its name suggests, is but a prototype. It shows us that it is possible to convert an HTML document into an X+V page where the grammar is created dynamically. This dynamic generation of grammar is the key to recognition accuracy. The prototype successfully shows that it can convert hyperlinks into speech based links, it can transform the content so that user can easily navigate in a page and finally the user can endlessly jump between links with only using speech only.

Section 2: Comparison with an existing system

We have compared our prototype with windows Vista's speech recognition software. In Windows Vista we had to say a certain command over and over, in our prototype it rarely had a unrecognized input. In addition, the Windows Vista's speech recognizer is for sighted users to have better access. Our system is targeted for blind users who are deprived of access to the web. Our system allows speech-only interaction. The Vista's speech recognizer needs GUI elements to be targeted after they are seen through the eye of the user. Our prototype gives access to the data on a level that HTML gives access to sighted users. For this reason blind users can easily access it with voice commands. It is as if the user is actually asking someone about a web page, its contents and its links.

Section 3: Complexity Analysis

Our prototype is of O(n) complexity where n is the length of the HTML page that is to be converted. According to our basic architecture displayed in figure 3, the grammar generator reads the entire HTML page taking O(n) amount of time. Then it parses the content in different phases each taking not more than O(n) amount of time one after another. The phases and their complexity are listed below:

1. Get Raw Content, O(n): When we are reading the source of a webpage, the reading is directly proportional to the number of characters present in the source. Therefore, the complexity of reading raw content is O(n).
2. Get Basic Grammar, O(1): A basic set of Grammars are already in place and it will take a constant amount of time to fetch. So the complexity is O(1).

3. Create the Dynamic Grammar, O(n): In this phase we are essentially increasing the vocabulary of our recognition grammar based on the number of words present in the source of the webpage. If the source contains n number of words, then n or less number of words to be added in the Grammar as duplicate words are removed. So it is also O(n).

4. Remove Duplicates, O(n): In order to increase recognition accuracy, duplicates should not be present in the grammar. So we had to remove duplicate words from the dynamic grammar, for it is very much possible that the source of the webpage might contain multiple occurrences of the same word. In order to remove multiple occurrences we looped through all the words present in the recognition grammar. In each iteration of the loop we pick a word from the source and create a hash value for the word. We then put the word in a hash table using the hash value as the key. But before we insert it, we check to see if the word already exists for the corresponding hash value. If it exists then we do not insert. We simply ignore it and move on to the next iteration. This loop is thus is of O(n) complexity. See Appendix VI for the actual code. We use a "near-perfect" hash function, and therefore do have to worry about collisions.

5. Get Basic JavaScript, O(1): The basic JavaScript functions and handlers that we need are already prepared and it will take a constant amount of time to fetch. So the complexity is O(1).

6. Create JavaScript Links and Paragraphs Array, O(n): The number of links and paragraphs are again a function of the total number of characters. The links were extracted in O(n) times and created into an array. The same thing was done for the paragraphs. The total complexity then is O(n) for this phase.

Therefore, the total complexity is O(n).

Section 4: Proof of Termination

The loops that are used inside the program all have a terminating condition.

The loop that reads in raw data, uses a counter with an initial integer value of 0 and increments of 1. The loop stops when the counter reaches the same value as the total character count of the file.

The loop that parses content also uses a character count to end the loop. When the counter, with an initial integer value of 0 and an increment of 1 reaches the character count(effectively indicating that there is nothing more to parse) then the parsing is stopped.

The loop that removes duplicate words uses the word count as the terminating integer and 0 as the initial value.

The loop that outputs parsed content uses a counter with an initial value of 0 and an increment of 1. The loop is stopped when the counter reaches the number of total parsed content. For example, when the content is parsed a number of paragraphs and links are parsed. They are then stored inside two arrays, one for the links and another for the paragraphs. When it comes to output these paragraphs and links we are use the length of the array as the terminating value.

The voice input loop terminates with the user command: "shut down" effectively stopping the entire program.

VIII. Conclusion

The prototype we have built shows clearly that it is possible to provide HTML content to blind users and allow them to browse forever using only speech as both input and output. Dynamical generation of the Grammar and the X+V page is something that we claim as new. Dynamically generating the Grammar increases the recognition accuracy for the downloaded and transformed page.

However, it has to be remembered our prototype is but a prototype. It cannot be denied that it has a few shortcomings. Our prototype uses grammar-based recognition so at the beginning when the user is trying to get to a website that is not listed as a basic website, the user must spell out all of the letters in the url.

Bibliography

1. Alonso, F., Frutos, S., Montes, C. and Others. (1998). A Generic Blind User Interface Model. 1998 *IEEE International Conference on Systems, Man, and Cybernetics.* (Cat. No.98CH36218). IEEE. Part vol.2, 1998. Pages: 1133-1138.

2. Alty, J. and Rigas, D. (1998). Communicating Graphical information to Blind users using Music: The Role of Context. *Proceedings of the SIGCHI conference on Human factors in computing systems 1998.* Pages: 574 – 581.

3. Arons, B. (1997). SpeechSkimmer: A System for Interactively Skimming Recorded Speech. *1997 ACM Transactions on Computer-human Interaction* Vol. 4, No.1 Pages 3-38 MIT Media Lab. U.S.A.

4. Arons, B. (1991). Hyperspeech: Navigating in SpeechOnly Hypermedia. *Proceedings of the third annual ACM conference on Hypertext*, San Antonio, Texas, United States 1991. Pages: 133 – 146.

5. Asakawa C., Takagi, H., Ino, S., and Ifukube, T. (2002). Auditory and Tactile Interfaces for Representing the Visual Effects on the Web. *ASSETS 2002, July 8-10, 2002 Edinburgh,*

Scotland. Pages 65-72. ISBN 1-58113-464-9-02/07. Tokyo Research Laboratory, IBM Japan & Research Institute for Electronic Science, Hokkaido University, Japan.

6. Bernsen, N. and Dybkjær, L. (1998). A Theory of Speech in Multimodal Systems. Proceedings of the ESCA Tutorial and Research Workshop on Interactive Dialogue in Multi-Modal Systems, Irsee, Germany, 1999, Pages. 105-108.

7. Berti, S. and Paternò, F. (2003). Model-based Design of Speech Interfaces. *DSV-IS* 2003: Pages: 231-244. Springer-Verlag GmbH.

8. Borodin, Y. (2006). A Flexible VXML Interpreter for Non-Visual Web Access. *ACM Assets 2006.*

9. Brewer, J., Dardailler, D. and Vanderheiden, D. (1998). Toolkit For Promoting WEB Accessibility. *Center On Disabilities Technology And Persons With Disabilities Conference 1998*. URL: http://www.csun.edu/cod/conf/1998/proceedings/csun98_057.htm

10. Brown, A., Pettifer, S. and Stevens, R. (2004). Evaluation of a NonVisual Molecule Browser. *ASSETS'04*, October 18–20, 2004, Atlanta. Pages: 40 –47.

11. Brunet, P., Feigenbaum, B., Harris, K. and Others. (2005). Accessibility requirements for systems design to accommodate users with vision impairments. *IBM Systems Journal*, Vol. 44, November 3, 2005. Pages: 445-466.

12. Capra, R., Pérez-Quiñones, M. and Ramakrishnan, N. (2001). WebContext: Remote Access to Shared Context. *ACM International Conference Proceeding Series; Vol. 15* ACM 1-58113-448-7-11/14/01. Pages: 1-9.

13. Donker, H., Klante, P. and Gorny, P. (2002). The Design of Auditory User Interfaces for Blind Users. *ACM International Conference Proceeding Series*; Vol. 31. Pages: 149-156.

14. Edwards, A. (1988). The Design of Auditory Interfaces for Visually Disabled Users. *Computer Human Interactions '88*. Pages 83-88. ACM ISBN-0-89791-265-9/88/0004/0083. Institute of Educational Technology, The Open University, Walton Hall, Milton Keynes, U.K.

15. Edwards, W., Mynatt, E. and Stockton, K. (1994). Providing Access to Graphical User Interfaces – Not Graphical Screens. *ASSETS '94 Marina del Rey, CA, USA, 1994*, ACM ISBN 0-89791-649-2. Pages 47-54.

16. Edwards, W., Mynatt, E. and Stockton, K. (1995). Access to graphical interfaces for blind users. *Interactions*. Volume 2, Issue 1. Pages: 54 - 67.

17. Frost, R. (2005). Call For a Public-Domain SPEECHWEB. *Communications of the ACM*. Volume 48, Issue 11 (November 2005). Pages: 45 - 49.

18. Hemphill, C. and Thrift, P. (1995). Surfing the Web by Voice. *ACM Multimedia 95 - Electronic Proceedings.* Pages 215 - 222.

19. Honkala, M. and Pohja, M. (2006). Multimodal Interaction with XForms. *Proceedings of the 6th international conference on Web engineering 2006*. Pages: 201-208.

20. Kane, R. and Yuschik, M. (1986). A Case Example of Human Factors in Product Definition: Needs Finding for a Voice Output Workstation for the Blind. *Conference on Human Factors in Computing Systems. Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface.* Toronto, Ontario, Canada. Pages: 69 – 73.

21. Kennel, A., Perrochon, L. and Darvishi, A. (1996). WAB: World Wide Web Access for Blind And Visually Impaired Computer Users. *ACM SIGCAPH Computers and the Physically Handicapped*. Pages: 10-15.

22. Klante, P. (2002). Visually Supported Design of Auditory User Interfaces. http://www-cg-hci-f.informatik.uni-oldenburg.de/ resources/ HCII2003-Klante-Visually%20Supported%20Design%20of%20AUI.pdf.

23. Klemmer, S., Sinha, A., Chen, J. and Others. (2000). Suede: a Wizard of Oz prototyping tool for speech user interfaces. *13th annual ACM symposium on User interface software and technology*, 1-58113-212-3. Pages: 1-10.

24. Kolias, C. , Kolias, V. , Anagnostopoulos, I. and others (2008). A pervasive Wiki application based on VoiceXML. *ACM Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*. Vol. 282, Article No. 58

25. Laux, L., McNally, P., Paciello, M. and Others. (1996). Designing the World Wide Web for People with Disabilities: A User Centered Design Approach. *ASSETS '96*, Vancouver, British Columbia, Canada. Pages: 94 – 101.

26. Leporini, B., Andronico, P. and Buzzi, M. (2004). Designing search engine user interfaces for the visually impaired. *Proceedings of the international cross-disciplinary workshop on Web accessibility*. Pages: 57-66.

27. McTear, M. (2002). Spoken Dialogue Technology: Enabling the Conversational User Interface. ACM *Computing Surveys (CSUR)*. Pages: 90-169.

28. Morley, S., Petrie, H., O'Neil, A. and Others. (1999). Auditory Navigation in Hyperspace: Design and Evaluation of a Non-Visual Hypermedia System for Blind Users. Behaviour & Information Technology, Taylor & Francis. e. Vol. 18, No. 1. Pages 18-26.

29. Mukherjee, S., Ramakrishnan, I. and Kifer, M. (2004). Semantic Bookmarking for Non-Visual Web Access. *ASSETS'04*, October 18–20, 2004. Pages: 185 – 192.

30. Munson, J. (1972). System study of a dial-up text reading service for the blind. ACM/CSC-ER. SESSION: SICCAPH 1 - Computers in the service of the handicapped - some issues and answers. iii. Pages: 228 - 239.

31. Murphy, E., McAllister, G., Strain, P. and Others. (2005). Audio For A multimodal Assistive Interface. *Proceedings ICAD 05 Eleventh Meeting of the International Conference on Auditory Display*. Pages 1–4.

32. Nass, C. and Gong, L. (2000). Speech interfaces from an evolutionary perspective. *Communications of the ACM* Vol. 43, Issue 9, Pages: 36 - 43.

33. Olsen, D. and Peachey, J. (2002). Query-by-critique: Spoken Language Access to Large Lists. *Symposium on User Interface Software and Technology*. Pages: 131 - 140

34. Paciello, M. (1999). Advanced Accessible Web Page Design*. Pre-Conference Session, CSUN's 1999 Conference Proceedings*. URL: http://www.dinf.ne.jp/doc/english/Us_Eu/conf/csun_99/session1001.html.

35. Pitt, I. and Edwards, A. (1996). Improving the Usability of Speech-Based Interfaces for Blind Users. *ASSETS 1996, Vancouver, British Columbia, Canada.* Pages 123-130. ISBN 0-89791-776-6/96/04. Department of Computer Science, University of York, Heslington, York, UK.

36. Raman, T. (1996). Emacspeak – A Speech Interface. CHI (Computer Human Interaction). p66ff, ACM Press, 1996 URL, http://emacspeak.sourceforge.net/.

37. Reece, G. (2002). Working to 508: Web Page and Interface Design for Compliance with the Americans with Disabilities Act (ADA). Web page assessment for compliance with the American with Disabilities Act*. In Proceedings of the 49th Annual Conference of the Society for Technical Communication* (STC): Vol. 1. Arlington, VA: STC. Pages: 1- 10.

38. Resnick, P. and Virzi, R. (1992). Skip and Scan: Cleaning Up Telephone Interfaces. *Conference on Human Factors in Computing Systems*. Pages: 419 – 426.

39. Rosenfeld, R., Olsen, D. and Rudnicky, A. (2001). *Universal Speech Interfaces. Interactions*: Volume 8, Issue 6, Pages: 34 - 44.

40. Roth, P., Petrucci, L., Pun, T. and Others. (1999). Auditory browser for blind and visually impaired users. *CHI '99* 15-20 May 1999. Pages: 218 – 219.

41. Savidis, A. and Stephanidis, C. (1995). Developing Dual User Interfaces for Integrating Blind and Sighted Users: The HOMER UIMS. *Conference on Human Factors in Computing Systems. Proceedings of the SIGCHI conference on Human factors in computing systems.* Pages: 106 – 113.

42. Schneider, J. and Strothotte, T. (2000). Constructive Exploration of Spatial Information by Blind Users. *ASSETS 2000, November 13-15, Arlington, Virginia, U.S.A.* Pages 188-192. ISBN 1-58113-314-8/00/0011. Department of Simulation and Graphics, Otto – von – Guericke University of Magdeburg, Magdeburg, Germany.

43. Smith, A., Francioni, J. and Matzek, S. (2000). A Java Programming Tool for Students with Visual Disabilities. *ACM SIGACCESS Conference on Assistive Technologies. Proceedings of the fourth international ACM conference on Assistive technologies.* Pages: 142-148.

44. Smith, A., Francioni, J., Anwar, M. and Others. (2004). Nonvisual Tool for Navigating Hierarchical Structures. *ACM SIGACCESS Accessibility and Computing*, Issue 77-78 Sept. 2003 - Jan. 2004. Pages: 133-139.

45. Stephanidis, C., Paramythis, A., Karagiannidis, C. and Others. (1997). Supporting Interface Adaptation: The AVANTI Web-Browser. 3rd ERCIM Workshop on User Interfaces for All, Alsace, France, 3-4 November, 1997.

46. Stephanidis, C. (1997). The AVANTI Web-Browser. ERCIM News No. 31 – October 1997. Source: http://www.ercim.org/publication/ Ercim_News/ enw31/stephanidis.html 1998-09-25 10:30:39 CEST.

47. Stephanidis, C., Paramythis, A., Sfyrakis, M. and Others. (1998). Adaptable and adaptive User Interfaces for Disabled Users in the AVANTI project. *Proceedings of the 5th International Conference on Intelligence and Services in Networks: Technology for Ubiquitous Telecom Services*, Pages: 153 - 166.

48. Stephanidis, C. (1998). Designing User Interfaces for All. *CSUN '98 Papers*. Web source: http://www.dinf.ne.jp/doc/english/Us_Eu/conf/csun_98/csun98_032.htm.

49. Walker, M., Fabbrizio, J., Mestel, C. and Others. (1998). What Can I Say? : Evaluating a spoken language interface to email. *CHI 98*. Pages: 582- 589.

50. Whittaker, S., Hirschberg, J., Amento, B. and Others. (2002). SCANMAIL: a voicemail interface that makes speech browsable, readable and searchable. *Conference on Human Factors in Computing Systems Proceedings of the SIGCHI conference on Human factors in computing systems 2002*. Pages: 275 – 282.

51. Whittaker, S. and Amento, B.(2004). *Semantic speech Editing*. CHI Volume 6, Number 1. Pages: 527 – 534.

52. Winberg, F. (2001). Auditory Direct Manipulation for Blind Computer Users. *Licentiate Thesis*. Royal Institute of Technology. KTH, Stockholm.

53. Winberg, F. and Bowers, J. (2004). Assembling the Senses: Towards the Design of Cooperative Interfaces for Visually Impaired Users. *Proceedings of the 2004 ACM conference on Computer supported cooperative work*. Pages: 332-341.

54. Winberg, F. and Hellström, S. (2003). Designing Accessible Auditory Drag and Drop. *CUU'03*, November 10-11, 2003, Vancouver, British Columbia, Canada. Pages: 152 – 153.

55. Yankelovich, N. and Lai, J. (1999). Designing Speech User Interfaces. *CHI '99 Extended Abstracts on Human Factors in Computing Systems.* ISBN: 1-58113-158-5. Pages: 124-125.

Appendix I

Section 4: General Discussion

Following the results from section 3 we shall now discuss the following aspects of research:

1. Computer interfaces for blind users.
2. Speech as a standard I/O medium.
3. Transformation of GUI's.
4. Internet access for blind users.
5. Multi-Modal applications.

4.1. Computer Interfaces for blind users

Any computer interface is highly dependent on the kind of input and output method. For our particular case we have reviewed the following input and output methods:

Input methods:

1. Speech.
2. Haptic device input.
3. Mouse/keyboard input.
4. Touch tablet.

Output methods:

1. Speech.
2. Non-Speech sound.
3. Braille or other tactile output.
4. Sensory output.

Any combination of these input and output methods can be included in the construction of an interface for blind users. For example, aiming at the possibility of adapting graphical interfaces for blind users, in the interface model described in (Edwards, 1988), a haptic device is used as input and speech and non-speech sound as output. Now that we have a general idea about the input and output methods, we can shift our focus towards the actual architecture of different interface design principles.

4.1.1. Auditory-Screen-Based Interfaces

The primary reason behind the proposal of this type of model is to convert GUI into something accessible by non-sighted users (Edwards, 1988). In addition, some other needs, such as the inefficiency of Braille tapes and audiotapes in terms of interaction with a computer (Morley, 1999) have also prompted the need for auditory-screen-based interface.

With the goal to make human-computer interaction easier it was proposed by (Edwards, 1988) that an auditory-screen-based system would easily be able to convey important information about the control screen to a blind user. From there the evolution of screen readers took place. But in (Edwards, 1994) it was noted that screen readers are a very rudimentary way of providing access to a certain graphical user interface. They do not provide access to the GUI elements, a feature whose absence only allows nothing more than awareness of the presence of a certain GUI element with which a sighted user might easily interact, where a blind user desires more in an ideal situation. To counter this problem the researchers (Edwards, 1994) suggested the use of auditory-screen-based interface.

This model proposes to convert regular GUI elements into an auditory interface. The most important characteristic that GUI enables a sighted user is that it exposes a number of interactive options in parallel. This cannot be transformed into auditory browser in the same extent as GUI due to the serial nature of audio and speech. The idea of transforming a GUI into an auditory interface is to provide the same set of actions a GUI exposes, regardless of their actual presentation on the screen (Edwards, 1994). For example, buttons provides the means to execute some command in the application. Menus on the other hand provide a list of commands. Therefore it is quite possible to provide the same semantic construct in an audio-based interface. The first step in that direction would be to convey information and the supported operation about the individual objects that constitute the interface. Since the presentation of the GUI elements is independent of its behaviour, tactile and auditory output can be used separately or in conjunction with each other. Tactile output can be used to convey different symbols and diagrams. Auditory interface on the other hand can be used to convey a broader range of information. Now we shall discuss a little more about the contents of such auditory interface. The interface contains:

1. Auditory objects.
2. Auditory Screen.

Auditory objects are defined by the conversion of a GUI object into a musical note or synthetic speech to which interaction is allowed through non-visual means. Auditory screen on the other hand contains different auditory objects and reflects a window like feature. The windows, however, are not allowed to overlap. They are placed in a grid like fashion.

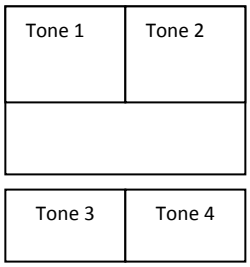

Figure 7: Auditory Screen.

Taken from page 84 of (Edwards, 1988)

The most common items that are contained in an auditory screen are: *documents, menus, dialogs* and *alerts.* The document window allows a user to perform modification to text-based documents. Now there are different approaches proposed for manipulation of a document via document window. One approach says when a user moves the cursor through each line of one document then different musical tones are played in addition to speaking out the words and characters that the line contains. Some researchers claimed that this approach is too slow and makes the user frustrated after a certain time. The alternative they propose is that the document window should not be designed as a direct analogue of its visual counterpart. Rather the user should be able to manipulate the text in terms of its syntactic structure. The document is divided into following syntactic units (Edwards, 1988):

1. The whole document.
2. The paragraph.
3. The Sentence.
4. The word.
5. The character.
6. The point between two characters.

The implementation of the *selection* concept is changed so that text can be selected in syntactic units. Selected text is also spoken whenever there is an alteration. One such adaptation of the above concept is “Mercator” (Edwards, 1994). It provides access to X-Win System. “DAHNI”

(Morley, 1999) is another auditory interface for navigating in hypermedia. Although, Its input methods are heavily depended on mouse, keyboard, joystick and touch tablets, the idea is similar to what we have described so far. The only difference is that the system also helps the user to visualize the navigational tree, through which the user may navigate by means of the input devices. The items in each node of the hypermedia system are either spoken out or displayed in a tactile device. For example, the experimenters in (Morley, 1999) have displayed a tactile picture of "House of Parliament", whereas in a different node the system describes a picture of Monet.

The following evaluation is released by the researcher (Edwards, 1994).

■**Response time:** Using Auditory interface a blind user is 15 times slower. However this could be reduced to 5 times if proper training were given.

■**Pointing Behaviour:**

■$T_{Position} = T_{Think} + dT_{Move}$ where

■d is the distance (number of windows crossed to the target)

■T_{Move} is the time taken to move the mouse;

■T_{Think} is a constant representing the time during which the mouse is not moved.

■$T_{Think} = 4.15$ S and $T_{Move} = 0.73$ S

■This suggests the time to locate an auditory object is quite high, resulting in a slower response from user. However, this could again be improved once the user gets used to it.

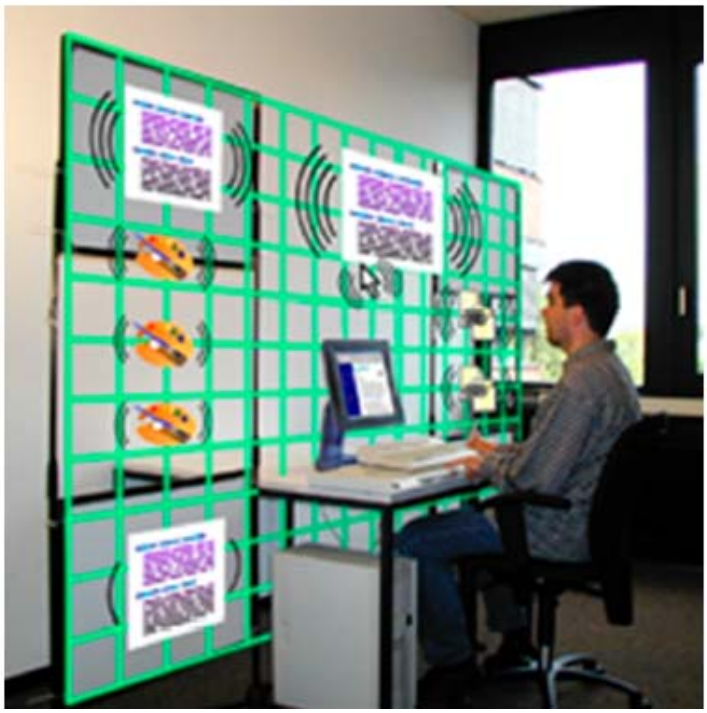

Figure 8: Auditory Interface in Action.

taken from Page 150 of (Donker, 2002).

A further expansion towards auditory interface is, instead of having a 2D auditory interface, it is suggested that (Donker, 2002) there should be a 3D audio space that would allow more items to be presented resulting in a higher level of interaction.

Now, we shift our focus towards the development of different interaction techniques that GUI allows and the auditory transformation of those techniques. The proposed auditory drag and drop (Morley, 1999) is one example. Another proposed scheme (Winberg, 2001) is heavily depended on non-speech sounds. For example, direct manipulation is a fundamental interaction technique of GUI. In addition it also suggests that we use non-speech sounds as an equivalent of GUI icons (earcons). The auditory drag and drop technique falls under that category and it was further enhanced (Winberg, 2003).

The auditory drag and drop includes three earcons. The first earcon indicates that the dragged auditory object is on top of another auditory object, the second one indicates that the object

was correctly dropped on the target window or auditory screen, and the third one indicates that the object was dropped on no target screen or item.

In general, all the research work mentioned above claimed the use of auditory interface is a success towards the solution of their research problems.

However, a number of problems were identified, most of which were related to the extra load imposed on the user's memory due to too much spoken content. As a solution to that it was proposed to have more information conveyed via the sounds used in the interface.

4.1.2. Speech-Dialog-Based Interface

The reason behind the proposal of this type interface is that the sequential nature (Arons, 1991) of audio interface, a nature which makes the transition of different GUI interaction techniques (such as, highlighting, to the right of etc.) very difficult to implement and use. Such research as (Arons, 1991), where the researchers goal was to create a speech only hypermedia system, the strong presence of speech-based-dialog based interface is absolutely vital. Also, aiming to solve the problem of creating a multi-modal system, speech-dialog-based system has been proposed (Yankelovich, 1999). In addition there is always the dream of the "conversational computer" (McTear, 2002) for which speech-dialog-based interface is also a must.

The idea behind this type of interface is quite simple. The user actually has a sort of conversation with the computer (McTear, 2002), thereby allowing him/her to execute different commands. This type of interface contains a speech synthesizer and a speech recognizer. The speech recognizer takes input and delegates the speech input to the application layer. In some systems, the application layer is independent of the method of input. If that is the case, then input could be a click of a button on a GUI or a verbal command in a speech interface. In both cases, the same operation could be executed if the application layer is independent of the input method. Once the execution of certain operation is performed the output is ready to be delivered. However, in the case of output, the application layer cannot stay independent from the output method. If the output is delivered in a GUI then the information can be presented in a parallel manner. In the case of a speech interface, it has been suggested by different researcher in (Arons, 1991), (Yankelovich, 1999), and (McTear, 2002) that the resultant output information should be precise, brief and due to the serial nature of sound it has to be sequential. Now we shall discuss a few examples of such speech interfaces.

The Hyperspeech system (Arons, 1991) is one example of the extensive dependence on speech-dialog-based interface. The aim of the researchers of this project was to create a hypermedia system with many nodes where different nodes will carry certain information but they all would be accessible via speech-based-interface. This system contains a hyperaudio database, a speech synthesizer and a speech recognizer. Conversation on different topics by different authors is recorded via telephone interview. These conversations are then manually translated into text. The transcripts for each question were then categorized into major themes (summary nodes) with supporting comments (detail nodes). In this manner about 20 minutes of speech was gathered and placed into the hyperaudio database. The prototype used X-Windows based tool to link the nodes in the database. The Hyperspeech system is implemented on a Sun SPARCstation using its own codecs to play the sound and speech segments. A serial controlled text-to-speech synthesizer is used for user output and for user input a board in a microcomputer provides the speech recognition capability.

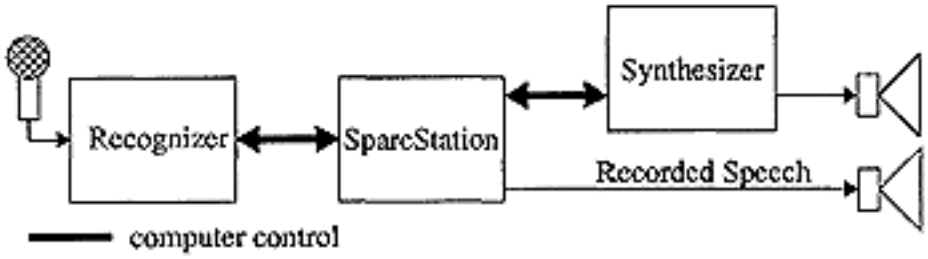

Figure 9: Hyperspeech, A Speech Dialog Based Interface from (Arons, 1991).

The above diagram from (Arons, 1991) explains the hardware architecture. The software architecture of the system constituted a program written in C and it was executed in standard Unix operating system. This program allowed a user to browse, listen and navigate through a certain conceptual navigation tree by means of executing a set of command words. Algorithm consisted a simple stack model that keeps track of all nodes that have been visited and permit the user to return to previously heard node at any time.

The problems exposed by the makers of this project are: There are too many menus and links at each node and sometimes the test subjects were actually lost in the navigation tree. They did not know where they were. To resolve this problem a uniform set of links were introduced in all nodes. It was claimed that after the introduction the system became very efficient and easy to navigate.

Another example of dialog based speech interface is an email speech interface (Walker, 1998).

A:	Hi, Elvis here. You have 5 new and 0 unread messages in your inbox. Say Read, or Summarize, or say Help for more options.
U:	Read.
A:	Select by Content or Position?
U:	Content.
A:	Select by Sender or Subject?
U:	Sender
A:	Which Sender?
U:	Kim.
A:	The message from Kim is about Meeting Tomorrow. The meeting tomorrow is at 10:30 in 2D-516.

Table 3: Dialog based speech interface for email from (Walker, 1998).

Although in this section we are discussing about speech-dialog-based interface and the way they are constructed, the discussion of a tool, which allows the creation of a speech-based-interface can also be deemed relevant. SUEDE (Klemmer, 2000) is one tool that allows the developers to create dialog-based application. The aim of the researchers here was to create the means through which speech-dialog-based systems could be easily created.

In SUEDE, the developer creates the different steps involved in the application by drag-n-drop approach. For example, if the developer wanted to create a set of response for the question: “How is the weather outside?”; he/she can create the following group of responses: “Rainy, Sunny, Cloudy ” etc.

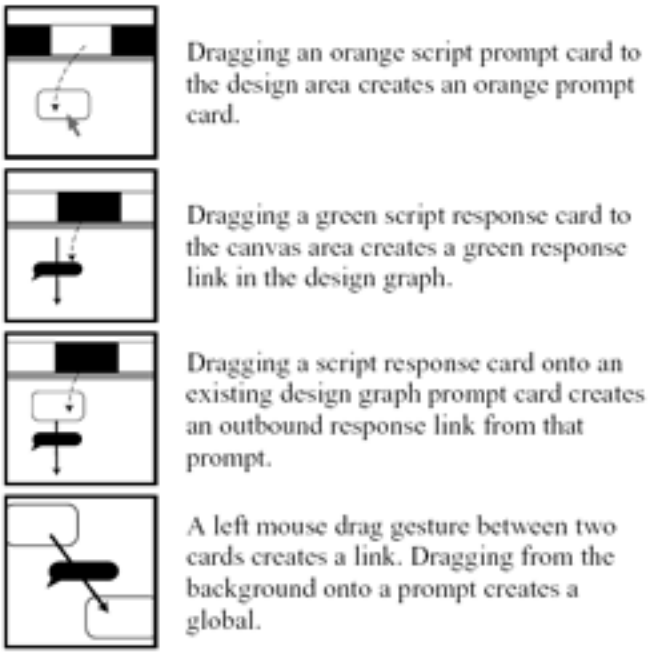

Figure 10: SUEDE, A speech dialog based system development kit.

Taken from Page 3 of (Klemmer, 2000).

Eventually the developer creates a graph like the following:

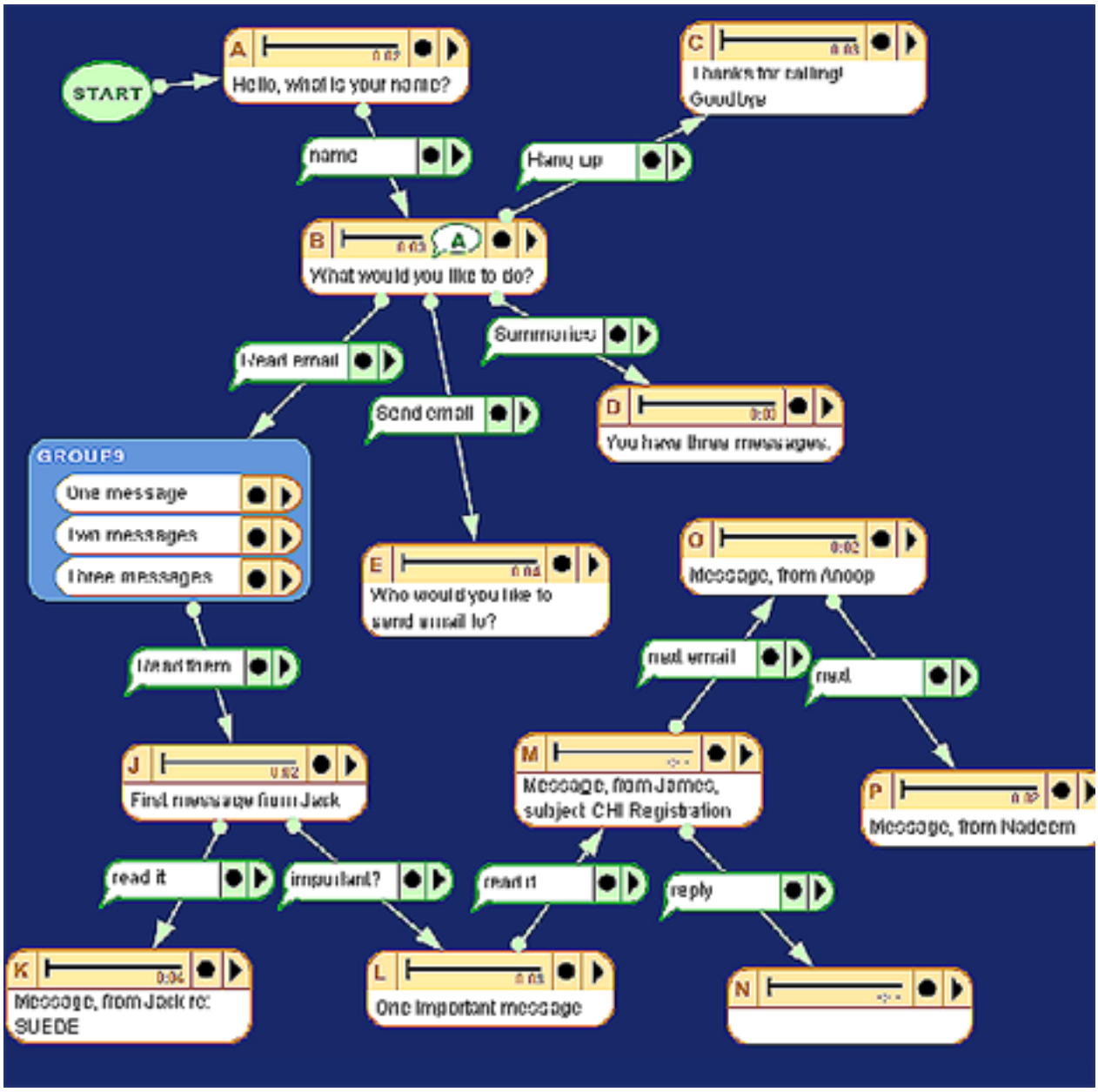

Figure 11: SUEDE Graph.

Taken from Page 2 of (Klemmer, 2000).

Once the design phase is finished the application can be tested in a visual environment by switching to “Test Mode”. SUEDE creates appropriate HTML file for every prompt in the above design graph. In “Test Mode”, SUEDE opens the first page in a browser like interface, except it can accept voice input and can provide synthesized speech output. The developer then performs testing and debugging operation by providing speech I/O. Any necessary change can be made by going back to design graph. SUEDE is used in the early phase of software design. It actually

allows creating sample prototypes very quickly. The completed prototype can then be moved to CSLU RAD (Klemmer, 2000) and/or Unisys' Natural Language Speech Assistant (NLSA) (Klemmer, 2000) for further refinement.

The problem with dialog-based interface is that it only allows a limited set of commands to be executed and the user must follow along a finite set of predefined path. With the addition of speech recognition the set of commands and the way they are uttered can be expanded to include a variety of styles and/or accents.

The Public Domain SpeechWeb (Frost, 2005) also employs a dialog-based speech interface. The aim of the researcher in this project was to make speech data browse-able, storable and transferable to different electronic media and data outlets. It uses a grammar to identify user input and provide appropriate response. It also allows linking between different nodes just by uttering a command like: "let me speak to solar man."

4.1.3. Other Non-Visual Interfaces

There are certain scenarios when dialog-based interfaces are not really suitable. Interfaces that deal with such cases are the focus of this section.

It was claimed by some researchers that currently-available screen readers are not efficient in reading hierarchical structures often found in computer interface. One such example of hierarchical structure would be the windows tree view component to view files and folders in a computer. It was reported by several researchers (Smith, 2000), (Smith, 2004) that a new tool called, JavaSpeak, has been created to resolve that problem. The tool is used as a plug-in with Eclipse IDE. The main feature of this product is that it uses sound tones of different frequency to distinguish between tree nodes at different depth. It accepts input via a keyboard and speech synthesizer provides output.

4.2. Speech as a standard I/O mediums

In this section we shall focus mostly on the I/O aspect of different research works and the resultant projects (or applications/tools) that addresses the issues faced by a blind user. In other words, we shall explore what are the different input and output methods of different blind-user-friendly systems.

4.2.1. Speech as Both the Input and Output

If we want to consider speech as the only input and output medium then the premise of our discussion is limited in what we had termed in the earlier section as “Dialog Based System”. Our focus in this section would be to explore how many different types of dialog-based systems are available at present, regardless of their relationship with a computer interface.

Now we shall look at some short examples of dialog-based systems.

In the exploration of the existing speech dialog based technology for the ultimate dream of a “conversational computer”, several types natural-language dialog-based systems (McTear, 2002) has been defined:

- State-Based Dialog Control
- Frame-Based Dialog Control
- Agent-Based Dialog Control

State-Based Dialog Control systems do not allow the user to deviate. It always guides the user along a certain predefined path. An example of that is the Nuance Automatic Banking System (McTear, 2002).

A sample conversation (McTear, 2002):

“

System: What would you like to do?

User: Pay a bill.

System: what Company would you like to pay?

User: Midland Bank.

System: How much would you like to pay?

User: One hundred and twenty-five pounds fifty-seven pence.

System: What date would you like the payment to be made on?

User: Monday.

System: You want to pay HSBC one hundred fifty-five pounds and fifty seven

pence on April 5th? Is this correct?

User: Yes.

System: Your payment will be sent. Would you like to pay another bill?

User: No.

”

Table 4: State based dialog control from (McTear, 2002).

An example of Frame-Based dialog system:

”

System: What is your destination?

User: London.

System: What day do you want to travel?

User: Friday.

System: What is your destination?

User: London on Friday around 10 in the morning.

System: I have the following connection .. ..

”

Table 5: Frame based dialog control from (McTear, 2002).

Example of Agent Based Dialog System:

”

User: I’m looking for a job in the Calais area. Are there any servers?

System: No, there aren’t any employment servers for Calais. However, there is an employment server for Pasde-Calais and an employment server for Lille. Are you interested in one of these?

”

Table 6: Agent based dialog control from (McTear, 2002).

The above three sample conversations tries to establish that the frame-based system is a bit more flexible (in terms of recognizing users utterance of certain sentence and providing proper response) than state-based system. The agent-based system on the other hand provides a whole new level of flexibility, for it is entirely possible to design a system so that the user can get help from different types of agent each equipped with AI algorithms.

From the need of hands-free, eyes-free interaction with information speech system is also claimed to be efficient (Olsen, 2002). The speech system is applied on a large list to gather information. The main component of the proposed speech system is a query language. This query language is very close to what we generally say in natural language. However, the query language includes a certain set of English language words and the verbal command that is uttered by the user must include some words from those set. The system does not support synonyms of the command words that consists the query language. A proposed set of command words for such a query language is given below (Olsen, 2002):

- Like <attribute name>
- Don't like <attribute name>
- Like [this | <object type>]
- Don't like [this | <object type>]
- Higher <attribute name>
- Lower <attribute name>

It also proposes some value specific command format:

- <Attribute name> less than <value>
- <Attribute name> greater than <value>
- <Attribute name> is <value>

The systems resultant sample dialog (Olsen, 2002)

```
"

U4: "rent less than $500"

C4: "apartment rent is $500, city is Salt Lake, bedrooms is 1"

U5: "city is Spanish Fork"
```

C5: “apartment city is Spanish Fork, rent is $600” “

Table 7: Value specific command format

4.2.2. Speech as Output

Screen-readers are the most popular tool used in the current market by blind users. They provide a very basic level of access to all kind of computer interface. It is not always sufficient but it has its use. One such product is JAWS by Freedom Scientific Inc.

Whereas traditional screen-readers have little or no contextual information about the contents of the display, the design of the speech-enabling approach as implemented in Emacspeak (Raman, 1996) is to overcome many of the shortcomings. The approach used by Emacspeak is very different from that of traditional screen-readers. Emacspeak does not speak the screen. Instead, applications provide both visual and speech feedback, and the speech feedback is designed to be sufficient by itself. While reducing cognitive load on the user this approach is relevant to providing general spoken access to information. Producing spoken output from within the application, rather than speaking the visually displayed information, vastly improves the quality of the spoken feedback. Thus, an application can display its results in a visually effective manner; the speech-enabling component renders the same in an aurally effective way.

When speech is used as the output medium it is very important that user be able to skim through the speech so that the entire process of accessing does not become inefficient. One such example that allows skimming through recorded speech is SpeechSkimmer (Arons, 1997). SpeechSkimmer allows a user to what we know as skim or browse through recorded audio by employing simple speech-processing techniques. A tactile input device provides the user with continues real time control over the speed and detail level of audio presentation.

The Concepts:

The concepts/ideas that are involved in such a system includes the following:

- Sampling Method.
- Time Compression: The length of time should be compressed in such a way that the speech remains intelligible.

- S.O.L.A. (Synchronized Overlap Add Method).
- Dichotic Presentation.
- Segmentation: This refers to creating segments or chunks of speech based upon either pause or pitch.
- Pauses – generally refer to beginning of a new sentence.
- Pitch – refers to beginning of a new paragraph.
- Energy – Emphasis detection.
- Speaker Identification.
- Intelligibility – Refers to the ability to identify isolated words.
- Comprehension – refers to understanding the content of the material.

" Studies have showed that isolated words that are carefully selected and trained can remain intelligible up to 10 times normal speed, while continuous speech remains comprehensible up to about twice(2x) normal speed. Time compression decreases comprehension because of a degradation of the speech signal and a processing overload of short-term memory." (Arons, 1997).

The Specification:

The following are specification of a speech skimmer.

- Skim **forward** through a recorded speech.
- Skim **backward** through a recorded speech.
- Control the skim **speed**.
- **Jump** to a different parts of a speech.
- In all the above cases, the intelligibility of speech must be maintained.

Post-Processing and Segmentation

- Each recording is post-processed with the speech detection and emphasis detection algorithms.
- Segments are created based upon – pause, emphasis, energy etc.
- These segment's information are kept in a single file.

4.3. Transformation of GUI's and Multi-Modal Applications

GUI's contain a lot of visual elements each of which encapsulates certain functionality. In most cases GUI's functional elements are available in a parallel fashion. Some applications that are focused towards serving the blind community most often tries to transform a certain GUI into an auditory or speech interface. Some examples of such applications are the lexical analysis tools (i.e. screen readers). Although it can be argued that the screen readers do transform a certain GUI into something audibly controllable, it is clear that no screen reader is capable of providing equal access as GUI does. It has been suggested by researchers (Edwards, 1994) that to have equal access as GUI we need to have a syntactic analysis of the application thereby not limiting ourselves only to its lexical analysis. When researcher focused on the syntactic analysis it seemed quite clear to them that this approach also opens the opportunity of turning one system into a multi-modal system. This approach allows the system to keep its existing visual representation and by adding another interface this approach is essentially turning the system into a dual-interface system. If this approach is extended a bit more it is apparent that it also allows single interface systems to be turned into multi-interface or multi-modal system just by adding different types of interface on top of the application core. So in this section the transformation of GUI's and Multi-Modal Systems will be discussed together.

HOMER UIMS (User Interface Management System) (Savidis, 1995) is a good example of a dual user interface. The aim of the researchers of this project is to create a certain system where both visually and non-visually supported access will be permitted. This system supports both the sighted users and the blind users. This system does not disregard the lexical analysis altogether, but it incorporates syntactic information with lexical analysis. Dual Objects- a term used for implying that a functional command can either be represented by a certain GUI button or by a speech dialog, encapsulates all the functional commands of the system.

Researchers (Berti, 2003), while working on a model-based solution to help designers in the development of voice applications, also identified the need for such design environments that are able to support development of user interfaces exploiting interaction modalities other than the traditional Graphical interfaces. Their particular solution to the problem is focused towards the speech interface. They propose a model-based solution to help designers in the development of Voice Applications. To be more specific, they explore the options of how to derive a speech interface implemented in VXML from an abstract user interface description. At

the end of the paper (Berti, 2003) they claim that they have achieved success in completing the above.

Another approach was implemented while working on a system, which aims to provide spatial and navigational information to visually impaired internet users through speech and non-speech audio with Haptic feedback (Murphy, 2005). In this approach, the software architecture is based on an AI agent, installed in the user's computer as a web plug-in. The way this plug-in works is that it presents the user with feedback to the nearest graphic or link object, from the users' relative position on the page. A force feedback mouse is used and its cursor position is constantly monitored by the plug-in that detects the surrounding objects. When the plug-in identifies an object in the vicinity of the cursor, the cursor position, relative to the object, is mapped to audio and haptic feedback. It provides the user with a sense of space and the formation of space through vibration and sound.

The diagram below explains the software architecture (Murphy, 2005):

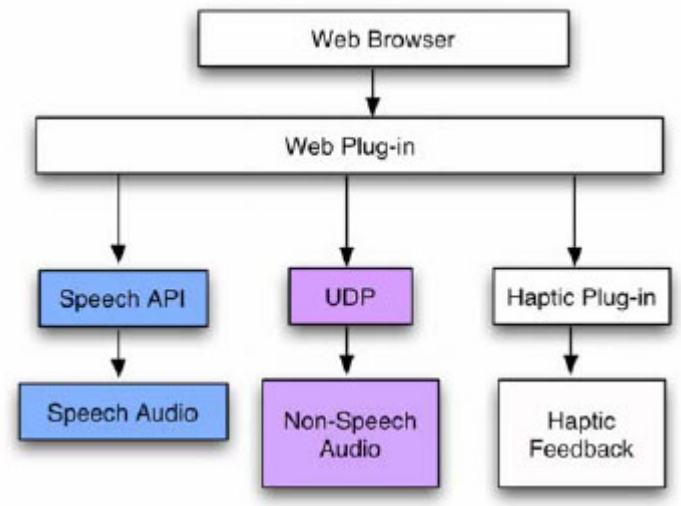

Figure 12: Audio for a multimodal assistive interface.

Taken from page ICAD05-2 of (Murphy, 2005).

This system (Murphy, 2005) also employs an auditory screen based interface.

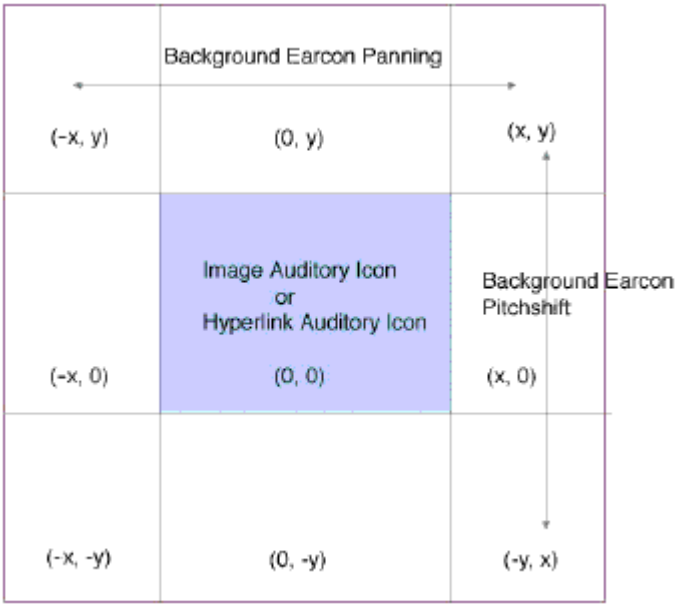

Figure 13: Auditory Icons or Earcons.

Taken from page ICAD05-2 of (Murphy, 2005).

The system (Murphy, 2005) has been tested. The test subjects were asked to draw the structure of a web page after they access it via the web plug-in. The published test result is given next:

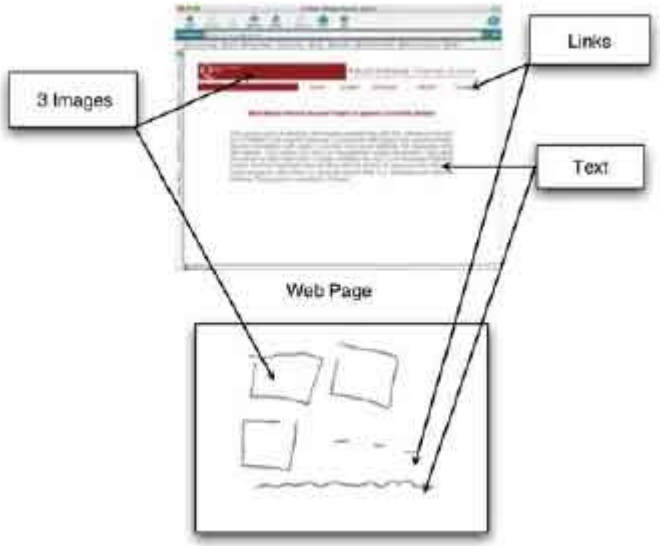

Figure 14: Transformation of a web page.

Taken from page ICAD05-3 of (Murphy, 2005).

Smart Pages, (Hemphill, 1995), were another idea to transform the webpage. The webpage has a meta-file attached where the meta-file contains voice navigational data. Through the use of this meta-file the user would be able to get speech access to the content. However, the draw back in that idea is that someone has to create the meta-file manually following a pre-defined transformation rule.

More recently, XFORM's, (Honkala, 2006), discusses the idea of developing an integrated environment where the transformation of a webpage to speech based application becomes easier.

4.4. Internet access for blind users

4.4.1. Internet Browser related technology and proposals

The researchers involved in the WAB (Kennel, 1996) project says, WAB transforms website into a form that could prove to useful for blind users to surf the internet where the titles, links and form elements are described textually and by means of special navigation aids it also allows hierarchical navigation. They also claim that the implemented system works with any WWW browser and screen reader and no client side software need not to be installed.

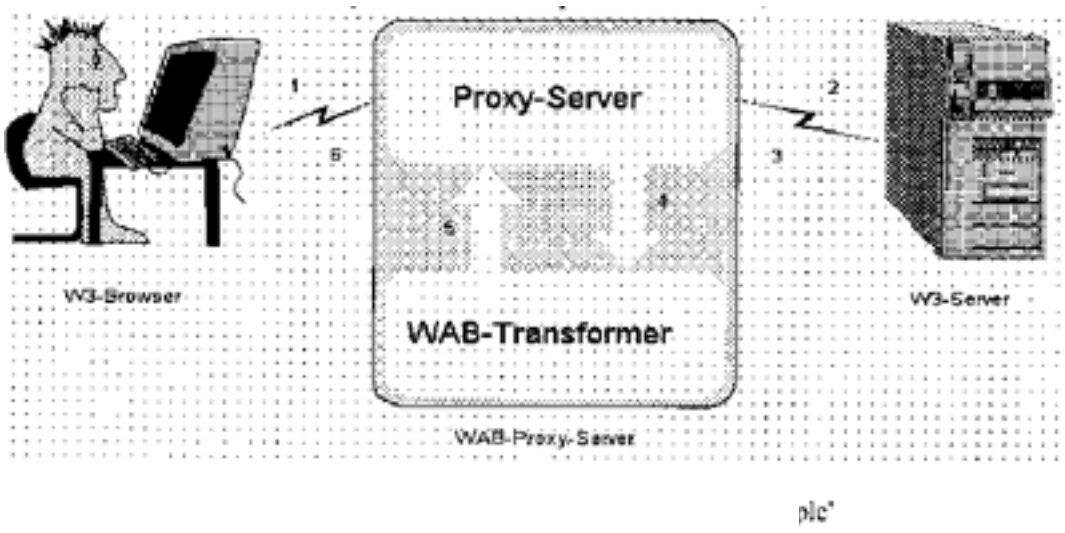

Figure 15: WAB Transformer.

Taken from page 14 of (Kennel, 1996).

Internet browser is the primary application in today's Internet. From the beginning of the invention of Internet and up until recently no Internet browser provided any mechanism for blind user's access in an easy accessible manner. With the release of Opera ™ 1 8.1 it has changed. This new version of Opera™ 8.1 provides mechanism to have auditory and speech access to web sites to a certain extent. The extent however, could be dramatically increased by the use of X+V application. X+V is a mixture of XHTML and VXML language. It provides client side scripting feature in such a manner that VXML applications could be embedded inside an HTML page and different functions could be tied together with different HTML events. That allows the development of auditory menus, dialogs etc. in a very flexible manner. In the example[2] below the programming style has been briefly discussed:

```
"<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:ev="http://www.w3.org/2001/xml-events">
<head>
  <title>Example 3: Drink dispenser</title>
  <vxml xmlns="http://www.w3.org/2001/vxml" id="drinkform">
    <field name="drink">
      <prompt>Would you like coffee, tea, or milk?</prompt>
      <grammar><![CDATA[
        #JSGF V1.0;
        grammar drinks;
        public <drinks> = coffee | tea | milk
       ]]>
      </grammar>
      <filled>
        <block>Sorry, I'm out of <value expr="drink"/>.</block>
      </filled>
    </field>
  </vxml:form></head>
<body ev:event="load" ev:handler="#drinkform">
<h1>Example 3: Drink dispenser</h1>
<p>Our drink dispenser can offer you a wide choice of refreshing drinks.</p>
</body>
</html>"
```

Table 8: X+V drink dispenser example.

[1] http://www.opera.com/products/desktop/
[2] http://my.opera.com/community/dev/voice/xv-intro/

The load event of the body tag initiates the VXML block of code. The VXML block consists of a field name Drink. This field also contains the grammar for the speech recognizer. In this particular example the grammar limits the possible input into 3 choices. However, depending on the complexity of the grammar many more choices could be incorporated. The "prompt" tag reads out the content that is enclosed. Then it waits for the user input. If the user input is any of the 3 choices it recognizes the choice and then speaks out the content inside the tag named "filled".

4.4.2. Server-side Transformation of Content

Server transformation is a very useful technique to provide content in such a manner that in can be accessed by blind users quite readily. One such example is the "AVANTI" (Stephanidis, 1997), (Stephanidis, 1997), (Stephanidis, 1998) project. This system is actually a multi-modal system, meaning it adapts to the need of the user. If the user is blind it tries to provide content in such a manner that the blind user could easily access the content. The makers of this project says:

"Aimed to address the interaction of individuals with diverse abilities, skills, requirements and preferences, using web-based information systems.

System Architecture".

The AVANTI framework comprises five main components:

- A collection of multimedia databases, which contain the actual information and are accessed through a common communication interface (Multimedia Database Interface-**MDI**)
- The User Modeling Server (**UMS**), which maintains and updates individual user profiles.
- The content Model (**CM**), which retains a meta-description of the information available in the system.
- The Hyper-Structure Adaptor (**HSA**), which adapts the information content, according to user characteristics, preferences and interests.
- The User Interface (**UI**) component, which is also capable of adapting itself to the users' abilities, skills and preferences, as well as to the current context of use.

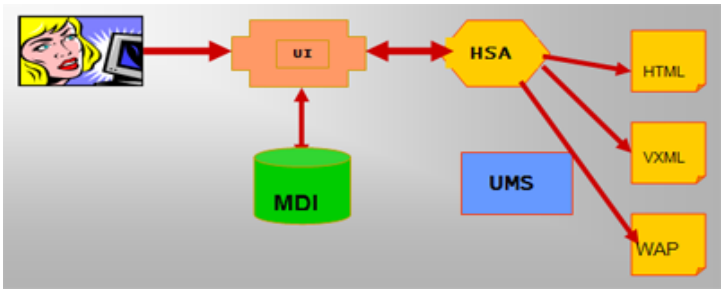

Figure 16: Reproduction of the AVANTI concept model.

4.4.3. Other types of Internet access tools

To make WebPages accessible in general, a researcher (Paciello, 1999), has come up with several specific instructions and guidelines.

Another researcher in (Reece, 2002) proposed a set of guidelines that are to be followed if a developer wanted to make the web page complaint with the ADA 508.

Search engines are one of the most common tools in today's Internet. Several researchers have focused on the access method to a search engine for blind user. Once such example is the work done in the evaluation of popular search engine accessibility (Leporini, 2004).

"JAWS" was used to test the usability of currently available search engines.

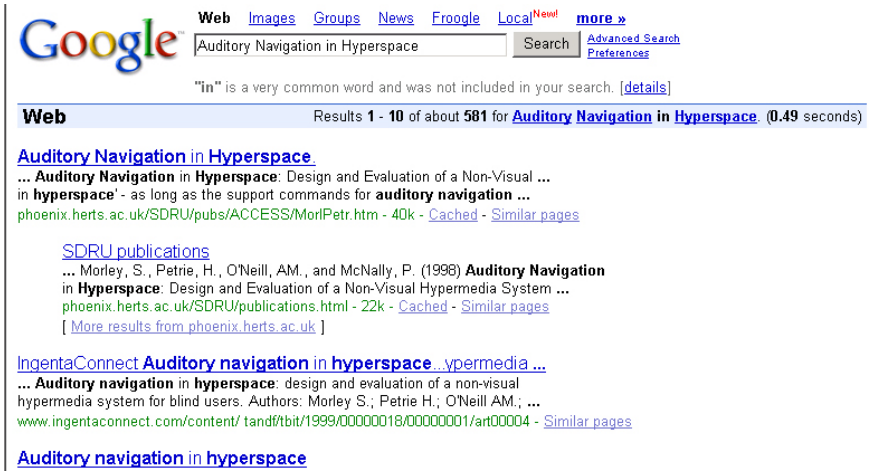

Figure 17: Testing with JAWS and currently available search engines.

Visual Layout vs. Aural Perception

- Google
- Link Web
- Link Images
- Link Groups

- Link News

"The Screen reader serializes each element – link, button, field, cell and so on – so that even a simple user interface becomes particularly long and tedious." (Leporini, 2004)

In addition all the search results have several common links. Those links will be repeated over and over.

Thus the researchers emphasized 3 major problems (Leporini, 2004):

Page Content serialization. The JAWS screen reader takes the page source and serializes its content.

Navigation by Tab key and special commands. Blind user usually prefers to visit the page link-by-link (By using the Tab key) or using special commands in order to move quickly around the pager. Many special screen reader commands operate well only if the developer has applied specific tags or attributes, or appropriate criteria have been followed.

What is offered by a visual layout differs from one provided for aural perception. Often when developers design a web page they provide some useful information by means of visual features, such as position, color, separating blank spaces, formatting features, and so forth. For instance some secondary information is put on the side so that users can recognize it immediately. It is important to provide the same message to a blind user by another means.

The following principles are proposed that should be considered in the design process of a search engine UI (Leporini, 2004):

1. Easy locations and labelling of edit field and search options.
2. Highlighting the search result.
3. Arranging the results.
4. Recognizing sponsored links.
5. Adding navigation and help links.
6. Navigating more quickly.
7. Alerting by sound.

CSS2 aural style sheets. – Aural style sheet proved by CSS 2 spec.

Appendix II

In our general WebVoice architecture, we have established that speech alone would be the method of interaction both as input and output. We shall now go into further detail of how to develop a system based on the proposed general architecture. We have chosen object oriented paradigm to develop our system.

Section 1: Detailed Requirement Analysis

1.1. Identifying User groups

In our target application there will be only one kind of user group, the general internet users. We shall refer to this group as users. However, according to our general architecture of the system (Figure 3) we notice that at the beginning only basic command set is available to the user. Once the user loads a new page only then the other commands become available. We can imagine two states of the same user. If we consider the first page user as a separate actor, namely "Main_page_user". Once a new page is loaded then that page and subsequent page users would be classified as "Newly_loaded_page_user". We hope that it will be easier to design our system with this approach. So to summarize for the purpose of design only we have classified two user groups:

1. Main Page User.
2. Newly Loaded Page User.

We shall now move on to describe a list of requirements that we think is sufficient enough for a speech only web content transformer. At first we shall list the requirements in a very broad fashion and later after analyzing the different case scenarios brought about by Use Case specification we shall produce the detailed final requirements.

1.2. List of Functionalities

We now present a general list of functionalities that is desired to fulfill our target.

1. The user should be able to visit a link.
2. The user should be able to query how many links present in a page.
3. The user should be able to query how many paragraphs present in a page.
4. The user may listen to a paragraph being read by the system.
5. The user may listen to a link label and link URL being read by the system.

6. The user may wish to find a paragraph containing a certain word.
7. The user may wish to find a link label containing a certain word.
8. The user may wish to shut down the system.
9. The user may ask for help.

We shall now refine the above list further by using the two user groups we had defined in the previous sub-section, namely "Main Page User" and "Newly Loaded Page User".

1.2.1. List of functionalities for "Main Page User"

1. Visit a default link.
2. Visit a default URL.
3. Shutdown the application.
4. Ask for help.

1.2.2. List of functionalities for "Newly Loaded Page User"

In addition to all the functionalities mentioned in 1.2.1, the "Newly Loaded Page User" should be able to perform the following:

1. Visit a link present in the web page.
2. Read a certain paragraph found in the web page.
3. Read the link label and link URL of a certain link found in the web page.
4. Find a paragraph or link label containing a certain word.
5. Count the number of links or paragraphs found in the web page.
6. Read All paragraphs found in the webpage.
7. Read the Title of the webpage.
8. Return to the Web Voice home page.

1.3. Use Cases

In this sub-section we shall discuss the different user interaction based case scenarios. This type of interaction model is called Use Cases. At the end of this procedure we should have clear understanding of our requirements for the system.

1.3.1. Use Case Specification:

Use case 1.1		Visit a default link.
Brief description		The user says “Load Default link 1” or “Load Default link 4” etc. and then the application, loads the URL for the link. Alternatively the user says “Go To yahoo.com” or “Go To PHP.net” etc. and then the application, loads the URL.
Actors		Main Page User
Preconditions		--
Main flow	A1	“Load Default Link link_number”
Alternative flows	A2	“Go To some_url”
Post-conditions		A new page is loaded with all the basic and extended commands.

Table 9: Use-Case "Load Default link"

Use case1. 2		Shutdown the application.
Brief description		The user says “Stop It” or “Shut Down” and then the application shuts itself down.
Actors		Main Page User
Preconditions		--
Main flow	A1	“Stop It”
Alternative flows	A2	“Shut Down”
Post-conditions		The application Shuts down.

Table 10: Use-Case "Shut Down"

Use case 1.3		Ask For Help.
Brief description		The user says “Help” and the voice prompt lists all the available commands.
Actors		Main Page User
Preconditions		--
Main flow	A1	“Help”
Alternative flows	--	
Post-conditions		The voice prompt lists all the available commands. If the Command is not recognized then the application should notify the user.

Table 11: Use-Case "Help"

Use case 2.1		Visit a link present in the web page.
Brief description		The user says “Load link 1” or “Load link 4” etc. and then the application, loads the URL for the link.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Load Link Link_number”
Alternative flows	--	--

Post-conditions	A new page is loaded with all the basic commands and extended commands. If the mentioned link is not found then the user is notified.

Table 12: Use-Case "Load Link"

Use case 2.2		Read a certain paragraph found in the web page.
Brief description		The user says “Read Paragraph 1” or “Read Paragraph 4” etc. and then the application reads the paragraph.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Read Paragraph paragraph_number”
Alternative flows	--	--
Post-conditions		The desired paragraph is read. If the mentioned paragraph number is not found then the application should notify the user.

Table 13: Use-Case "Read Paragraph"

Use case 2.3	Read the link label and link URL of a certain link found in the web page.
Brief description	The user says “Read Link 1” or “Read Link 43” etc. and then the application reads the link

		label and link URL.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Read Link link_number”
Alternative flows	--	
Post-conditions		The link label and link URL of the desired link is read. If the mentioned link number is not found then the application should notify the user.

Table 14: Use-Case "Read link"

Use case 2.4		Find a paragraph containing a certain word.
Brief description		The user says “Find the paragraph with the word some_word” and all the paragraphs in the page are searched for the word. It returns the paragraph number of all the matching paragraphs.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Find the paragraph with the word some_word”
Alternative flows	--	

Post-conditions	The matching paragraph numbers are read. If the word was not found in any paragraph then the user is notified.

Table 15: Use-Case " Find the paragraph with the word "

Use case 2.5		Find a link label containing a certain word.
Brief description		The user says “Find the link with the word some_word” and all the link labels in the page are searched for the word. It returns the link number of all the matching links.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Find the link with the word some_word”
Alternative flows	--	
Post-conditions		The matching link numbers are read. If the word was not found in any link label then the user is notified.

Table 16: Use-Case "Find the link"

Use case 2.6	Count the number of links found in the web page.
Brief description	The user says “How Many Links” and the system replies with the total number of links found in the page.

Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“How Many Links?”
Alternative flows	--	
Post-conditions		The total number of links found in the page.

Table 17: Use-Case "How many links"

Use case 2.7		Count the number of links found in the web page.
Brief description		The user says “How Many Paragraphs” and the system replies with the total number of paragraphs found in the page.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“How Many Paragraphs?”
Alternative flows	--	
Post-conditions		The total number of paragraphs found in the page.

Table 18: Use-Case "How many paragraphs"

Next, we have created two new use cases as a generalization of all the basic commands and extended commands. In other words, we have added two new use cases by grouping the basic commands as one and extended one as another.

Use case 3		The user may issue any basic command.
Brief description		The user may say any of the commands listed in use case 1.1 to 1.3.
Actors		Main Page User
Preconditions		--
Main flow	A1	“Load Default Link link_number”
Alternative flows	A2	“Go To some_url”
Alternative flows	A3	“Stop It”
Alternative flows	A4	“Shut Down”
Alternative flows	A5	“Help”
Post-conditions		Appropriate use case post-conditions to be applied as described in use cases 1.1 to 1.3

Table 19: Use-Case "Basic Commands"

Use case 4		The user may issue any extended command.
Brief description		The user may say any of the commands listed in use case 2.1 to 2.7.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Load link link_number”
Alternative flows	A2	“Read Paragraph paragraph_number”
Alternative flows	A3	“Read Link link_number”

Alternative flows	A4	"Find the paragraph with the word some_word"
Alternative flows	A5	"Find the link with the word some_word"
Alternative flows	A6	"How Many Links?"
Alternative flows	A7	"How Many Paragraphs?"
Post-conditions		Appropriate use case post-conditions to be applied as described in use cases 2.1 to 2.7

Table 20: Use-Case "Extended Commands"

In the following sub-section we present a Use Case diagram created in Rational Rose.

1.3.2. Use Case Diagrams:

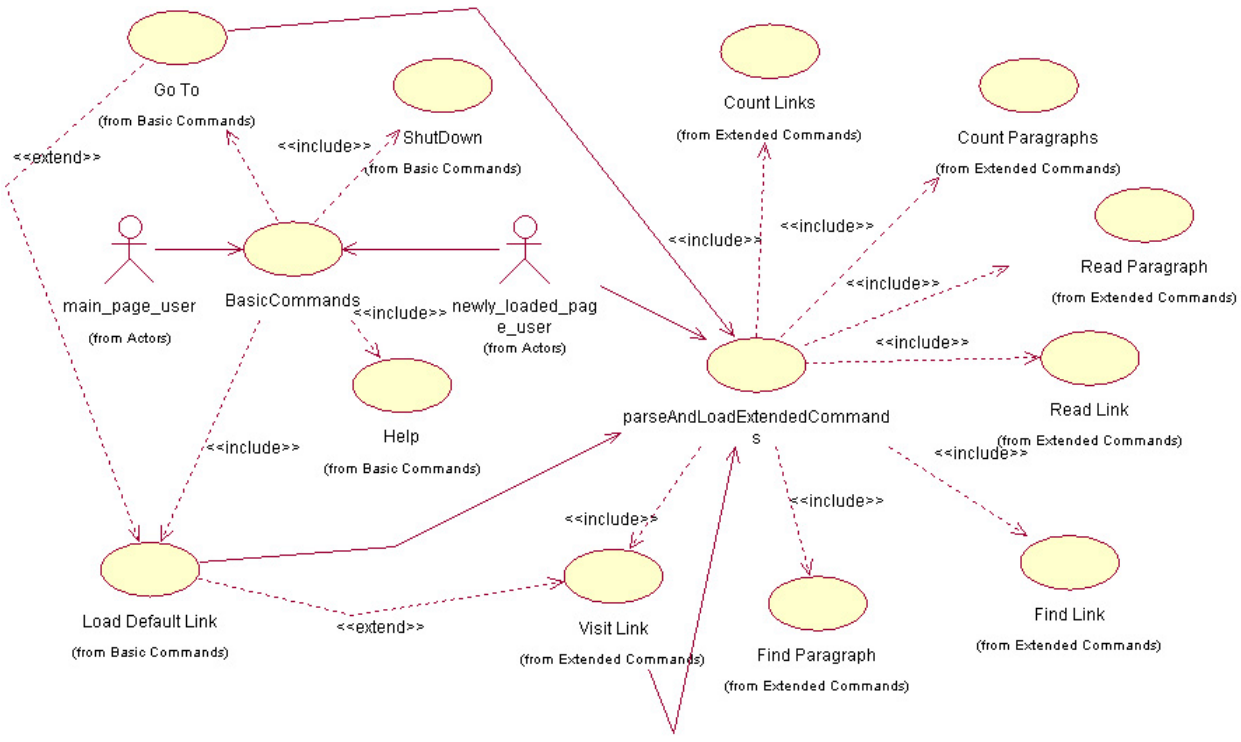

Figure 18: Complete Use-Case Overview

Now we shall describe each of these use cases as a combination of some more detail oriented use cases.

We have produced the following diagram for the basic command “Load Default Link”.

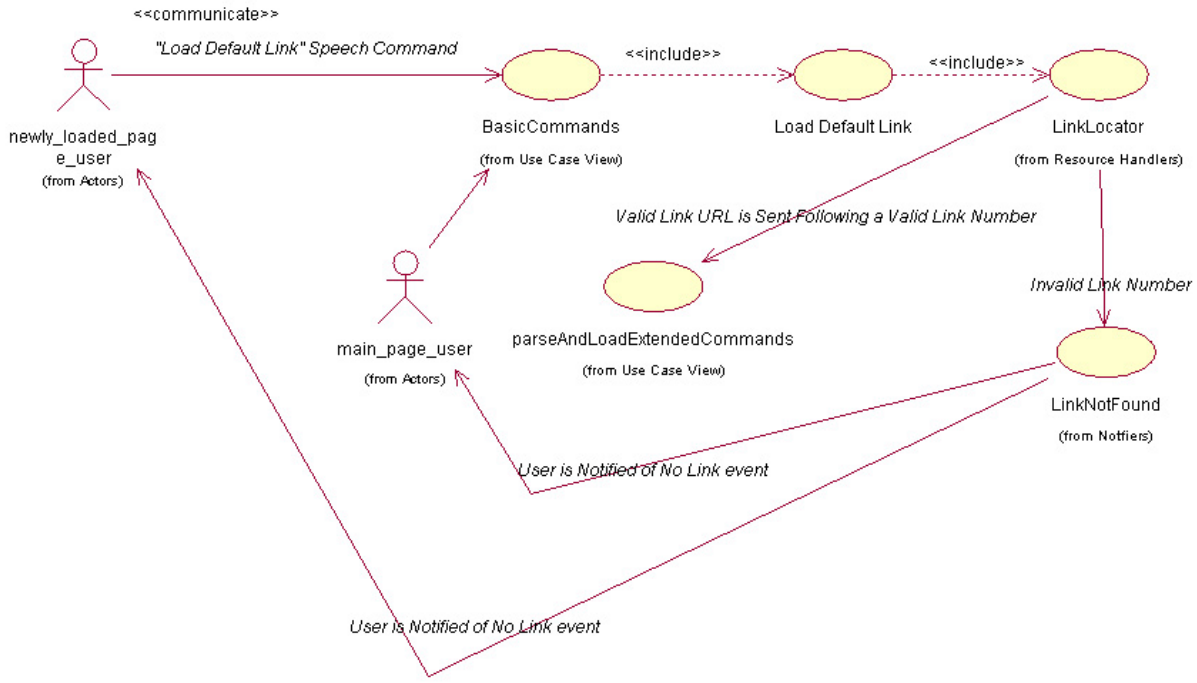

Figure 19: Use-Case Diagram for "Load Default Link"

In the above diagram, we have introduced two new use cases. They are, “LinkLocator” and “LinkNotFound”. Below is the specification of the two.

Use case 5.1: LinkLocator	Finds the link label and link URL of the given link number.
Brief description	The user says “Find the link with the word some_word” and all the link labels in the page are searched for the word. It returns the link number of all the matching links.

Actors		Newly Loaded Page User, Main Page User
Preconditions		Use Case 1.1
Main flow	A1	Default Link number is successfully parsed from the issued command in use case 1.1
Alternative flows	--	
Post-conditions		The matching link numbers are read. If the word was not found in any link label then the user is notified.

Table 21: Use-Case " LinkLocator "

Use case 5.2: LinkNotFound		Notifies the user that link is not found.
Brief description		The user says “Find the link with the word some_word” and the word is not found in the link. Then the system notifies the user of the failure.
Actors		Newly Loaded Page User, Main Page User
Preconditions		Use Case 1.1
Main flow	A1	Default Link number is successfully parsed from the issued command in use case 1.1
Alternative flows	--	
Post-conditions		The matching link numbers are read. If the word was not found in any link label then the user is notified.

Table 22: Use-Case " LinkNotFound "

We have produced the following use case diagram for the extended command "Load link link#"

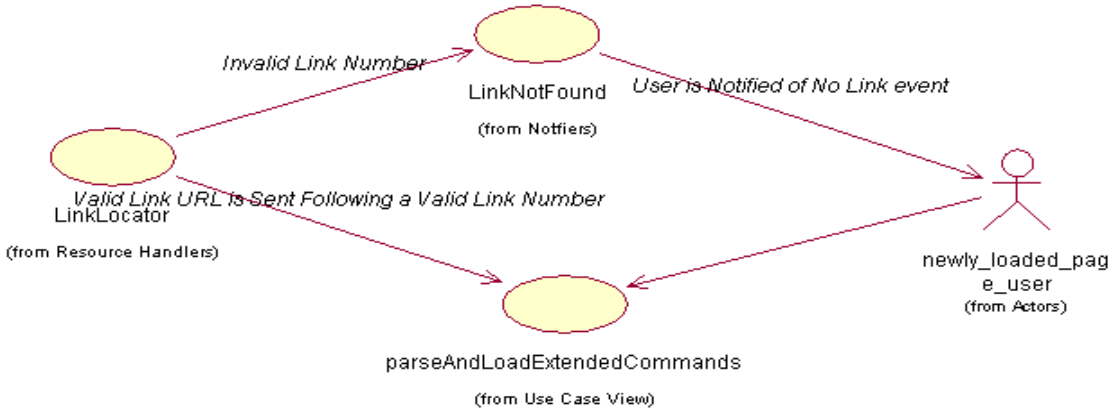

Figure 20: Use-Case Diagram for "LoadLink"

Following is the diagram of the use-case "Find Paragraph With the word 'XYZ'". At first the word will be parsed from the whole sentence that the user uttered. If that word is in the vocabulary of our recognition grammar that was generated dynamically from the source webpage, then the word will be recognized. Then we shall find the paragraph(s) that contains the word. It is possible that the word is a part of the recognition grammars vocabulary but no paragraphs contain it. Finally, if we find the matching paragraphs we return the paragraph numbers.

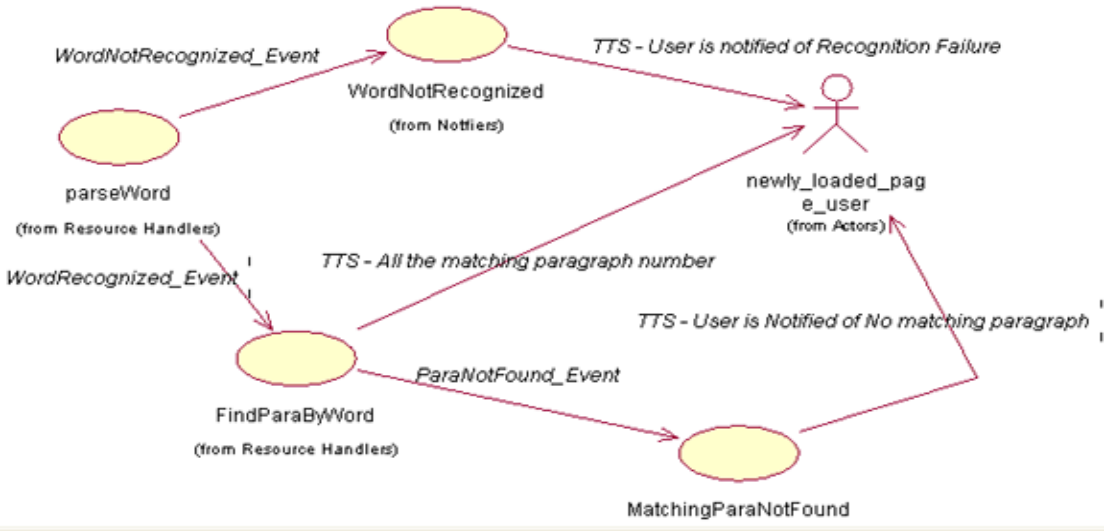

Figure 21: Use-Case Diagram for "Find Paragraph"

The diagram below shows find link use case diagram

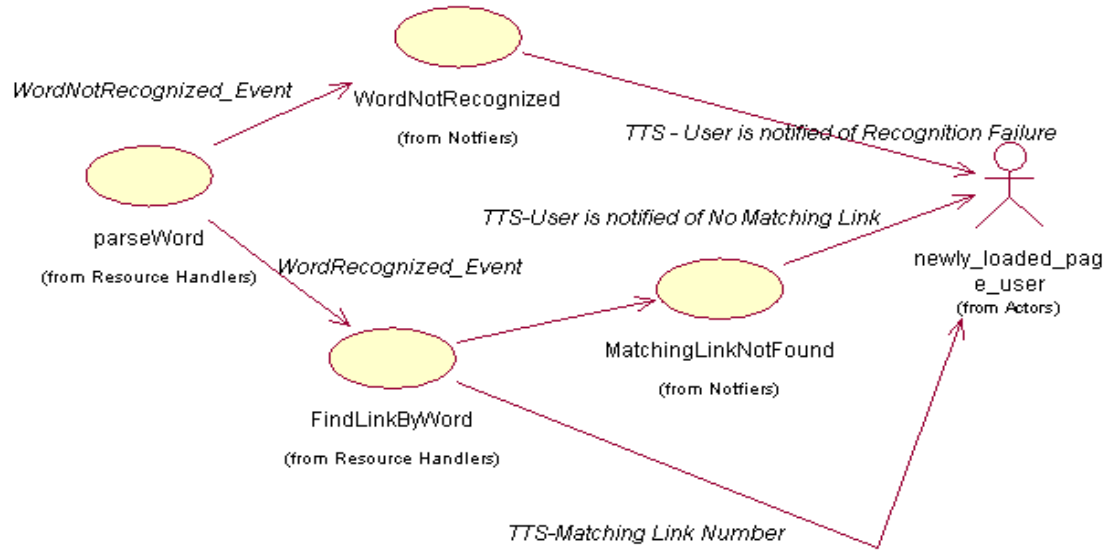

Figure 22: Use-Case Diagram for "Find Link"

The following is an use-case diagram of the read paragraph.

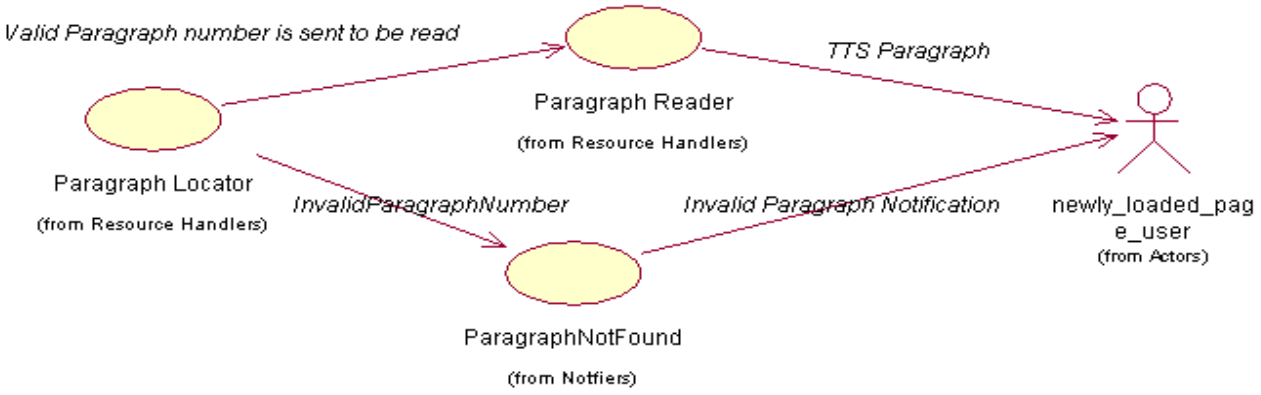

Figure 23: Use-Case Diagram for "Read Paragraph"

Appendix III - Detailed System Architecture

Section 1: Model based architectural Description

1.1. State Model

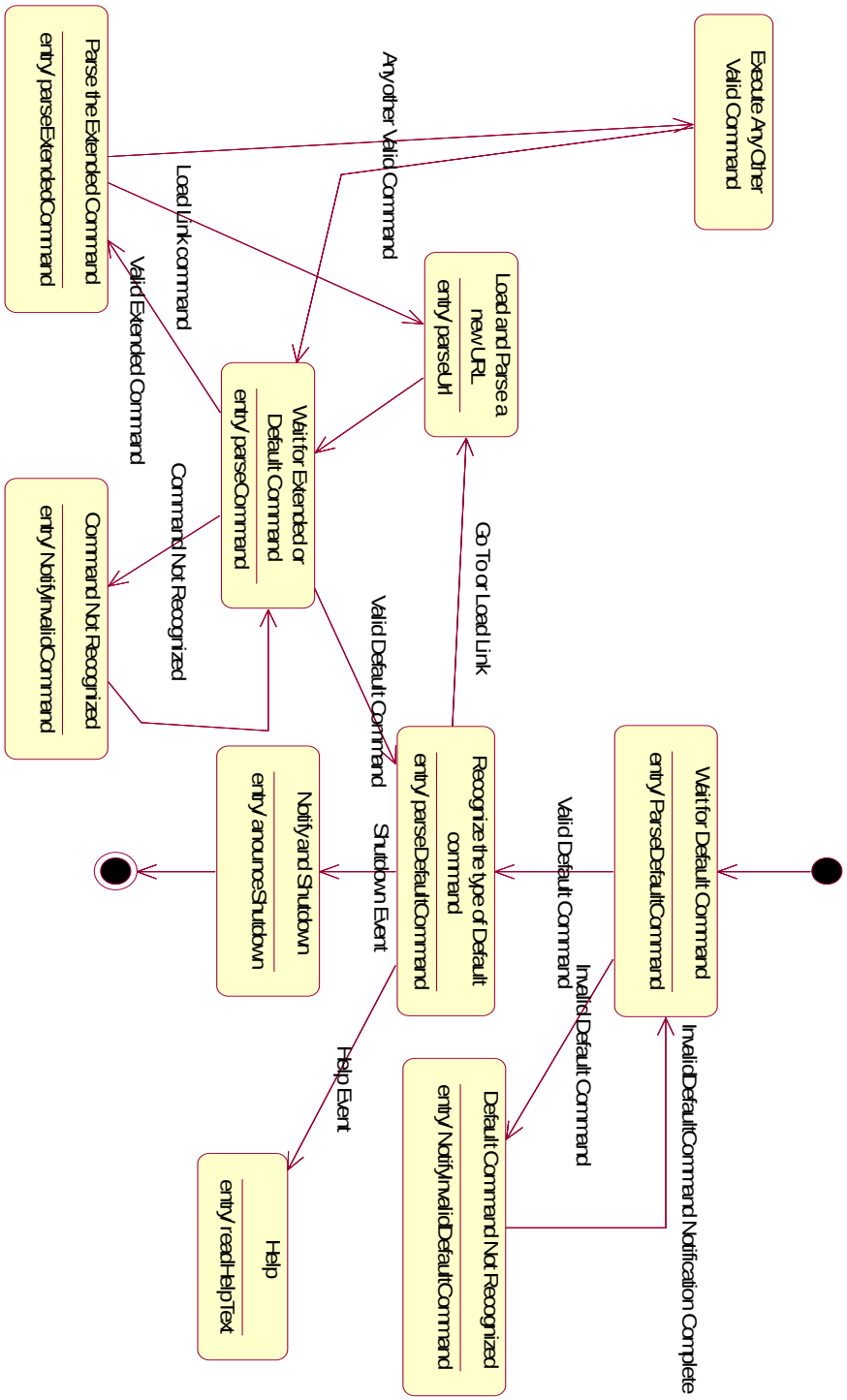

Figure 24: General System States

1.2. Logical Model

In our application we have used X+V, Grammar and JavaScript for the client side interaction. In addition, we have used PHP for the server side processing. PHP is object oriented and JavaScript also supports objection oriented notions. X+V pages are basically XML that supports event based calls to external script such as JavaScript. Grammars contain rules and semantics, where each rule may contain some other rule.

1.2.1. Assumption

The two later items X+V and Grammar, are not fully Object Oriented for they do not support inheritance, encapsulation, polymorphism etc. However, we can apply the following transformation and consider them as classes. They are:

1. Consider both X+V Page and Grammar as a final class.
2. In the case of X+V, consider each event handler as a method call. Inside each event handler we would either find a call to some JavaScript function or it will fire some other event. In either case, we can safely view them as methods with speech as return value.
3. In the case of Grammar, consider each rule as a method and each semantic value as an attribute. Methods can call other methods or itself which is just as same as the grammar rules. So again, we can safely view the whole grammar as one class and each rule as methods.

Now, we shall move on to the logical view of our application.

1.2.2. Basic Architecture

Since application is a web application, it is therefore falls under client-server architectural model. The server prepares the content in X+V format and the client interact with the X+V page. As a part of the interaction process the client may also trigger a server side request resulting in a newly prepared X+V page with similar capabilities.

First, we shall show the client side design.

Client Side Class Diagram:

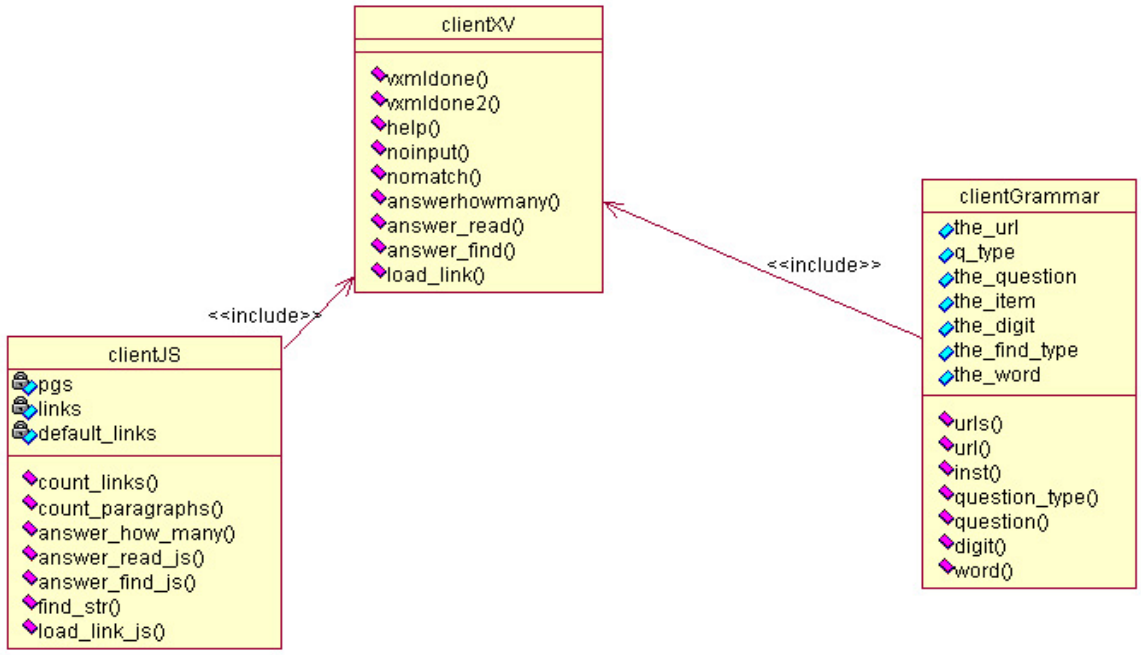

Figure 25: Client side class diagram

In the above diagram, we have 3 classes:

1. clientXV: The actual X+V page with event handlers referred as methods.
2. clientJS: JSON class containing the JavaScript handlers for the X+V events.
3. clientGrammar: The recognition Grammar with rules referred as methods and semantic variables referred as attributes.

The clientXV class contains the clientJS class and clientGrammar. With the help of those two classes the clientXV class handles the user commands.

Now, we shall show sequence diagrams of the processing of several user commands.

The next diagram explains what happens when the user says “How many links?”.

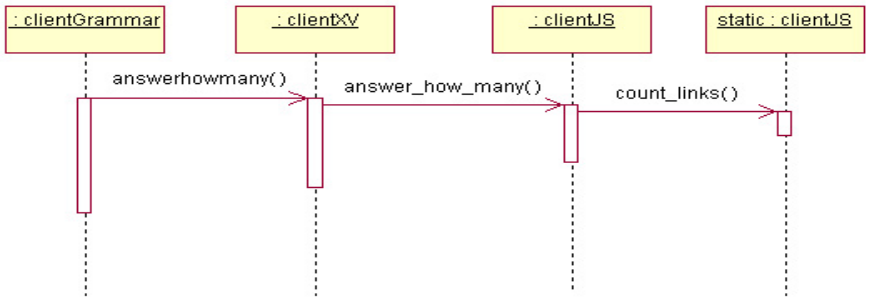

Figure 26: Sequence Diagram for the "How many links?" command

The diagram below explains what happens when the user says “Find the paragraph with the word ‘about’ ”.

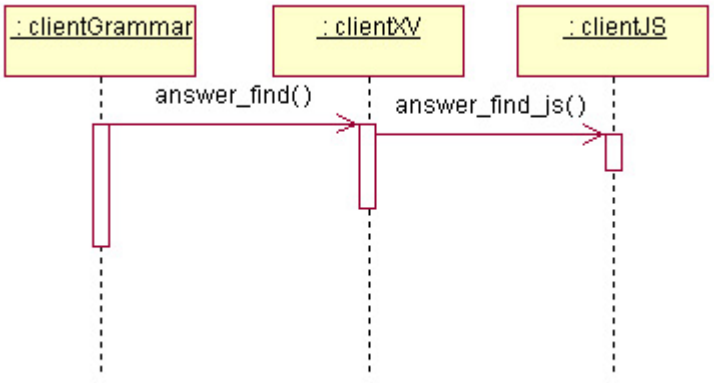

Figure 27: Sequence Diagram for the "Find the paragraph with the word 'about'"

In order to show the sequence diagram for load link we have to show the class diagram of the server side first.

Server Side class diagram:

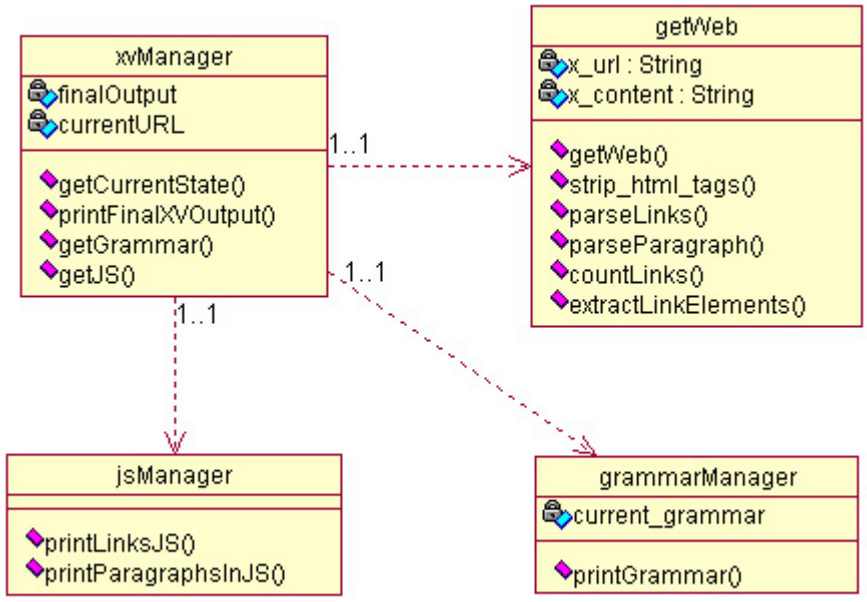

Figure 28: Server side class diagram

In the diagram above, we have described 4 classes:

1. xvManager: the main class that will output the X+V, Grammar and JavaScript.
2. jsManager: the class responsible for properly generating the required JavaScript.
3. grammarManager: the class responsible for generating the required Grammar.
4. getWeb: the class responsible for creating a socket to newly requested page and then parse the content.

The sequence diagram below shows how the Load Link command is executed:

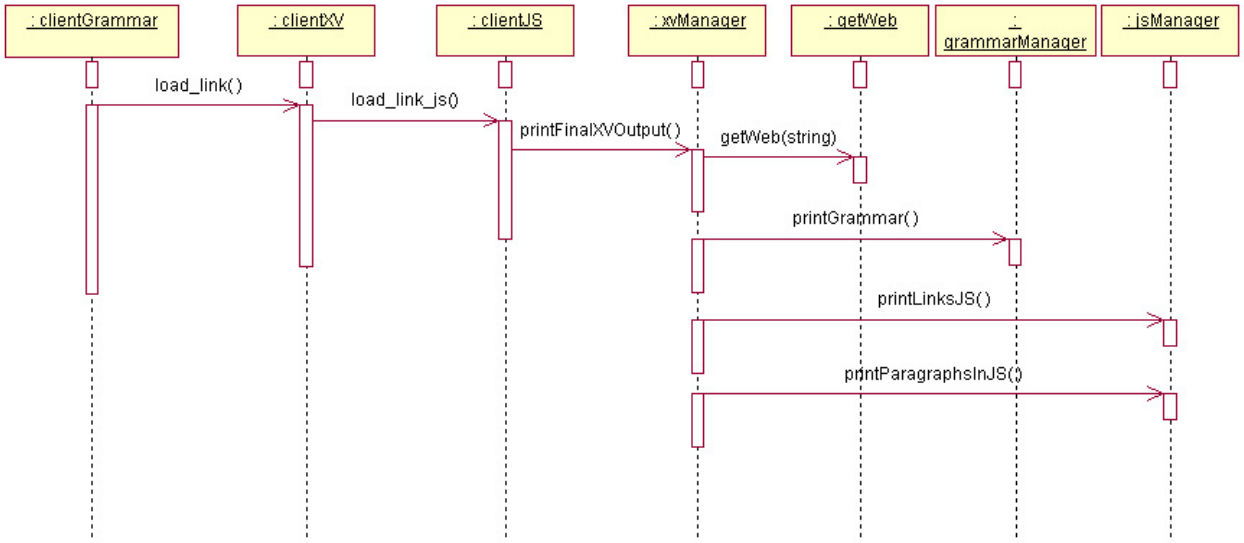

Figure 29: Execution of "Load Link " Command

Appendix IV - Example Scenarios/Demonstration of the prototype

We shall use USE-CASES to describe our result from the demonstration.

Use case 1.1		Visit a default link.
Brief description		The user says “Load Default link 1” or “Load Default link 4” etc. and then the application should load the URL for the link.
Actors		Main Page User
Preconditions		--
Main flow	A1	“Load Default Link 1”
Result		http://en.wikipedia.org/wiki/SpeechWeb is loaded and WebVoice is ready to explore links of that page

Table 23: Example "Load Default Link1"

Use case 1.1 Alternative flow		Visit a default link.
Brief description		Alternatively the user says “Go To yahoo.com” or “Go To PHP.net” etc. and then the application should load the URL.
Actors		Main Page User
Preconditions		--
Alternative flows	A2	“Go To php.net”
Result		http://www.php.net is loaded and WebVoice is ready to explore links of that page

Table 24: Example "Go to Php.net"

Use case1. 2		Shutdown the application.
Brief description		The user says “Stop It” or “Shut Down” and then the application shuts itself down.
Actors		Main Page User
Preconditions		--
Main flow	A1	“Stop It”
Alternative flows	A2	“Shut Down”
Result		The application Shuts down.

Table 25: Example "Shut Down"

Use case 1.3		Ask For Help.
Brief description		The user says “Help” and the voice prompt lists all the available commands.
Actors		Main Page User
Preconditions		--
Main flow	A1	“Help”
Alternative flows	--	
Result		The voice prompt lists all the available commands. If the Command is not recognized then the application should notify the user.

Table 26: Example "Help"

Use case 2.1		Visit a link present in the web page.
Brief description		The user says “Load link 1” or “Load link 4” etc. and then the application, loads the URL for the link.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Load link 10”
Alternative flows	--	--
Result		A new page is loaded with all the basic commands and extended commands. If the mentioned link is not found then the user is notified.

Table 27: Example "Load link 10"

Use case 2.2		Read a certain paragraph found in the web page.
Brief description		The user says “Read Paragraph 1” or “Read Paragraph 4” etc. and then the application reads the paragraph.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Read Paragraph paragraph_number”
Alternative flows	--	--

Result	The desired paragraph is read. If the mentioned paragraph number is not found then the application should notify the user.

Table 28: Example "Read Paragraph 2"

Use case 2.3		Read the link label and link URL of a certain link found in the web page.
Brief description		The user says “Read Link 1” or “Read Link 43” etc. and then the application reads the link label and link URL.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Read Link link_number”
Alternative flows	--	
Result		The link label and link URL of the desired link is read. If the mentioned link number is not found then the application should notify the user.

Table 29: Example "Read Link1"

Use case 2.4	Find a paragraph containing a certain word.
Brief description	The user says “Find the paragraph with the word some_word” and all the paragraphs in the page are searched for the word. It returns

		the paragraph number of all the matching paragraphs.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Find the paragraph with the word some_word”
Alternative flows	--	
Result		The matching paragraph numbers are read. If the word was not found in any paragraph then the user is notified.

Table 30: Example "Find The paragraph with the word 'XYZ'."

Use case 2.5		Find a link label containing a certain word.
Brief description		The user says “Find the link with the word some_word” and all the link labels in the page are searched for the word. It returns the link number of all the matching links.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Find the link with the word some_word”
Alternative flows	--	

Result	The matching link numbers are read. If the word was not found in any link label then the user is notified.

Table 31: Example "Find the link with word 'XYZ'. "

Use case 2.6		Count the number of links found in the web page.
Brief description		The user says “How Many Links” and the system replies with the total number of links found in the page.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“How Many Links?”
Alternative flows	--	
Result		The total number of links found in the page.

Table 32: Example "How many links"

Use case 2.7		Count the number of links found in the web page.
Brief description		The user says “How Many Paragraphs” and the system replies with the total number of paragraphs found in the page.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“How Many Paragraphs?”
Alternative flows	--	

Result	The total number of paragraphs found in the page.

Table 33: Example "How many paragraphs"

Next, we have created two new use cases as a generalization of all the basic commands and extended commands. In other words, we have added two new use cases by grouping the basic commands as one and extended one as another.

Use case 3		The user may issue any basic command.
Brief description		The user may say any of the commands listed in use case 1.1 to 1.3.
Actors		Main Page User
Preconditions		--
Main flow	A1	"Load Default Link link_number"
Alternative flows	A2	"Go To some_url"
Alternative flows	A3	"Stop It"
Alternative flows	A4	"Shut Down"
Alternative flows	A5	"Help"
Result		Appropriate use case post-conditions to be applied as described in use cases 1.1 to 1.3

Table 34: Example - Issue a basic command

Use case 4	The user may issue any extended command.
Brief description	The user may say any of the commands listed

		in use case 2.1 to 2.7.
Actors		Newly Loaded Page User
Preconditions		Use Case 1.1
Main flow	A1	“Load link link_number”
Alternative flows	A2	“Read Paragraph paragraph_number”
Alternative flows	A3	“Read Link link_number”
Alternative flows	A4	“Find the paragraph with the word some_word”
Alternative flows	A5	“Find the link with the word some_word”
Alternative flows	A6	“How Many Links?”
Alternative flows	A7	“How Many Paragraphs?”
Result		Appropriate use case post-conditions to be applied as described in use cases 2.1 to 2.7

Table 35: Example - Issue an Extended Command

Appendix V - Prototype Codes

Section 1. Grammar

```
<?php

/*-------------------------------------------------------------

        THIS IS THE GRAMMAR GENERATOR CLASS grammarManager.

        THE FUNCTION: printGrammar() IS CALLED BY xvManager class

-------------------------------------------------------------------*/


class GrammarManager

{

        private $current_grammar="";

        public function __construct()

        {

                $this->current_grammar = " <grammar><![CDATA[

                                #JSGF V1.0;

                                grammar urls;

                                public <urls> = <url>{\$.the_url=\$url;q_type=''} | (stop it
| shut down|shut up|be quiet){\$='stop it'} | [Go to] web voice home{\$='web voice home'}
| Go Back| Something Else;

                                <url> = yahoo.com | google.com | msn.com | bdwindsor.org |
php.net |bbc.com|cnn.com|windsorstar.com|blogs.com|wikipedia.org;

                                public   <inst>   =   <question_type>{q_type=\$question_type}
<question>{\$.the_question = \$question} ;

                                <question_type> = How Many | Read   | (Find | Is There |
Search){\$='Find'} | Go To | Load | Load Default;

                                <question>    =        title    |links    |    [    All    ]
paragraphs{\$='paragraphs'}  |  Paragraph{\$.the_item='Paragraph'}  <digit>{\$.the_digit  =
\$digit}|   Link   {\$.the_item='Link'}   <digit>{\$.the_digit   =   \$digit}   |   [the|a]
paragraph{\$.the_find_type='para'}     [with     the     |     that     has     the]     word
<word>{\$.the_word=\$word}|[the]      link{\$.the_find_type='link'}      [with]      [the]
word<word>{\$.the_word=\$word}|[the] Paragraph{\$.the_item='paragraph_read'} with the word
<word>{\$.the_word = \$word};

                                <digit> = 1|2|3|4|5|6|7|8|9|10  ;

                                <word> = remain | silence | working by| believe ;
```

```
                        ]]>
                    </grammar>";
        }



        public function __toString()
        {
                return $this->current_grammar;
        }
        public function printGrammar()
        {
                return $this->current_grammar;
        }
}
?>
```

Section 2. Javascript

```
<?php
/*-------------------------------------------------------------
        THIS IS THE JAVASCRIPT GENERATOR CLASS JSManager.
        THE FUNCTION: printLinkJs() and printParagraphJS() IS CALLED BY xvManager class
-------------------------------------------------------------------*/
class JSManager
{
        public function printJS(&$webx, $urlx)
        {
                ?>
 <script>
  <![CDATA[
                var links = new Array();
                 <?php
                        if(!empty($urlx))
                        {
                                $webx->printLinksJS();
                        }
                ?>
                var default_links = new Array();

                //print the default links
                //this is part of the basic commands
                default_links[1] = new Array();
                default_links[1][0]="http://en.wikipedia.org/wiki/SpeechWeb";
                default_links[1][1]="speechweb Wikipedia web page";

                default_links[2] = new Array();
                default_links[2][0]="http://www.cnn.com";
                default_links[2][1]="CNN";
                default_links[3] = new Array();
```

```
default_links[3][0]="http://dev.opera.com/articles/view/xhtml-voice-in-
style/";
default_links[3][1]="X+V Styles";
default_links[4] = new Array();
default_links[4][0]="http://www.windsorstar.com/";
default_links[4][1]="The Windsor Star";

 var pgs = new Array();
 pgs[0]="";
 <?php

       if(!empty($urlx))
       {
              $webx->printParagraphsInJS();
       }
       else
       {
              //print the default links
              //this is just for example purpose
              ?>
               pgs[1]="this is the first paragraph.";
               pgs[2]="this is the second paragraph.";
               pgs[3]=" this is the third paragraph.";
               pgs[4]="this is the fourth paragraph."

              <?php
       }
?>
//--------------------------------
function count_links()
{
       return links.length-1;
}
//--------------------------------
```

```
function count_paragraphs()
{
	return pgs.length-1;
}
//--------------------------------
function answer_how_many(a)
{
	var si = a.interpretation;
	if(si.the_question=='links')
		return " "+count_links()+" links.";
	else if (si.the_question=='paragraphs')
		return " "+count_paragraphs()+" paragraphs.";
}
//-----------------------------------------
function answer_read_js(a)
{
	var si = a.interpretation;
	if(si.the_question.the_item)
	{
		if(si.the_question.the_item=='Paragraph')
		{
			if(si.the_question.the_digit<pgs.length)
				return " Paragraph "+si.the_question.the_digit+": "+pgs[si.the_question.the_digit]+": Finished Reading Paragraph "+si.the_question.the_digit;
			else
				return "No Such Paragraph";

		}
		else if(si.the_question.the_item=='Link')
		{
			if(si.the_question.the_digit<links.length)
```

```
                                        return " Link "+si.the_question.the_digit+": Link Label: "+links[si.the_question.the_digit][1]+": Link U R L: "+links[si.the_question.the_digit][0]+" Finished Reading Link "+si.the_question.the_digit;
                                    else
                                        return "No Such Link!";
                                }
                            }
                            else if (si.the_question.the_word)//if the word is present then it means the user requested "read the paragraph with the word 'about'"
                            {

                                a.interpretation.the_question.the_find_type='para';
                                return answer_find_js(a);
                            }
                            else if(si.the_question)
                            {
                                if(si.the_question == 'links')
                                {
                                    return " It can take long time to read all the links. It is suggested that you command me to read one link at a time. For example, say: read link 1";
                                }
                                else if(si.the_question == 'paragraphs')
                                {
                                    if(<?php if(!empty($urlx)) { print $webx->count_paragraph_words();}else{print "0";}?>  >7000)
                                    {
                                        return " It can take long time to read all the paragraphs. It is suggested that you command me to read one paragraph at a time. For example, say: read paragraph 1";
                                    }
                                    else
                                    {
                                        //read all the paragraphs
                                        var paragraphs_all="Reading All "+(pgs.length-1)+" Paragraphs.  ";
```

```
                                        for(i=1; i<pgs.length;i++)
                                        {
                                                paragraphs_all += "Reading Paragraph
"+i+":  "+pgs[i]+": Finished Reading Paragraph "+i+": ";
                                        }
                                        paragraphs_all    +=    "Finished    Reading
"+(pgs.length-1)+" Paragraphs.";
                                        return paragraphs_all;
                                }
                        }
                        else if(si.the_question == 'title')
                        {
                                 <?php
                                        if(!empty($urlx))
                                        {
                                                echo        "return        \"".$webx-
>printTitle()."\";";
                                        }
                                        else
                                        {
                                                echo "return 'Webvoice Home';";
                                        }
                                ?>
                        }
                }
                else
                {
                        return " Command Not Recognized!";
                }
        }
        //-----------------------------------------
        function answer_find_js(a)
        {
                var si = a.interpretation;
```

```
var i=0;
var temp_str="";
var temp_arr=new Array();
var temp_arr_count=0;

if(si.the_question.the_find_type=='para')
{
	for(i=1; i<=count_paragraphs();i++)
	{
		temp_str="I have found the following: paragraph ";
		pg_txt="";
		pg_txt = pgs[i];
		pg_split=pg_txt.split(" ");

		if(find_str(pg_split,
si.the_question.the_word)==true)
		{
			temp_arr[temp_arr_count] = i;
			temp_arr_count++;
		}
	}

	if(temp_arr_count>1)
	{
		for(i=0;i<temp_arr_count;i++)
		{
			temp_str +=""+temp_arr[i];
			if(i+2==temp_arr_count)
				temp_str += " and ";
			else
				temp_str += ", ";
		}
	}
	else
```

```
                        {
                                if(temp_arr_count==1)
                                        temp_str += ""+temp_arr[0];
                        }
                        temp_str +=" contains the word: "+si.the_question.the_word;


                        if(temp_arr_count==0)
                                temp_str =" I did not find a matching paragraph for
the word: "+si.the_question.the_word;
                }
                else if(si.the_question.the_find_type=='link')
                {
                        for(i=1; i<=count_links();i++)
                        {
                                temp_str="I have found ";
                                pg_txt="";
                                if(links[i])
                                {
                                        if(typeof(links[i][1])!='undefined')
                                        {
                                                pg_txt = links[i][1];


                                        }
                                }
                                pg_split=pg_txt.split(" ");
                if(find_str(pg_split, si.the_question.the_word)==true)
                                {
                                        temp_arr[temp_arr_count] = i;
                                        temp_arr_count++;
                                }
                        }
        link_dialog_mid = " containing the word: "+si.the_question.the_word;
if(temp_arr_count>1 && temp_arr_count<=20) //i'll return only if the number of resulted
links is less than 20
```

```
{
temp_str += temp_arr_count+ " links "+link_dialog_mid+". They are: link ";
                                        for(i=0;i<temp_arr_count;i++)
                                        {
                                                temp_str +=""+temp_arr[i];
                                                if(i+2==temp_arr_count)
                                                        temp_str += " and ";
                                                else
                                                        temp_str += ", ";
                                        }
                                }
                                else if(temp_arr_count>20)
                                {
                                        temp_str     +=     temp_arr_count+     "     links
"+link_dialog_mid+". I am reading the first 20 links: link ";
                                        for(i=0;i<20;i++)
                                        {
                                                temp_str +=""+temp_arr[i];
                                                if(i+2==20)
                                                        temp_str += " and ";
                                                else
                                                        temp_str += ", ";
                                        }
                                }
                                else
                                {
                                        if(temp_arr_count==1)
                                        {
                                                temp_str   +=     "   link:   "+temp_arr[0]+"
"+link_dialog_mid+".";
                                        }
                                }
                                if(temp_arr_count==0)
```

```
                                temp_str =" I did not find a matching links for the
word: "+si.the_question.the_word;
                        }
                        return temp_str;
                }
                //-------------------------------------------------------
                function find_str(arrayObj, myStr)
                {
                        var i=0;
                        for(i=0;i<arrayObj.length;i++)
                        {
                                if (arrayObj[i].toUpperCase() == myStr.toUpperCase())
                                {
                                        return true;
                                }
                        }
                        return false;
                }
                //-------------------------------------------------------------
                function load_default_link_js(a)
                {
                        var si = a.interpretation;
                        if(si.the_question.the_item)
                        {
                                if(si.the_question.the_item=='Link')
                                {
                                        document.location                                =
"xvManager.php?url="+default_links[si.the_question.the_digit][0];
                                }
                        }
                }
                //-------------------------------------------------------------
                function load_link_js(a)
                {
```

```
                var si = a.interpretation;
                //if an url was said
                if(si.the_url)
                {
                        temp_url = si.the_url;
                        //remove all the spaces
                        while(temp_url.indexOf(' ')>0)
                        {
                                temp_url = temp_url.replace(' ', '');
                        }
                        document.location="xvManager.php?url="+temp_url;
                        //alert(si.the_url);
                }
                else
                {
                        if(si.the_question.the_item)
                        {
                                if(si.the_question.the_item=='Link')
                                {
                                        document.location                               =
"xvManager.php?url="+links[si.the_question.the_digit][0];
                                }
                        }
                }
        }
        //-------------------------------------------------------------
        function load_home_js()
        {
                document.location = "xvManager.php";
        }
        //-------------------------------------------------------------
        function load_last_js()
        {
                history.back();
```

```
                }
                //-----------------------------------------------------
                function mydebug(a)
                {
                        var si = a.interpretation;
                        alert(_event);
                        return si;
                }
           ]]>
 </script>
                <?php
        }
}
?>
```

Section 3. X+V

```
<?php
        /*-------------------------------------------------
                THIS IS THE MAIN CLASS xvManager.
                THE MAIN FUNCTION IS: printFinalXVOutput()
        -------------------------------------------------*/
$out = new xvManager();
$out->printFinalXVOutput();


class xvManager
{
        private $finalOutput="";
        private $currentURL="";
        //++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        public function getCurrentState()
        {
                /*-----------------------------------------------------------
                |:     this function checks the $this->currentURL variable.
                |:     If it is empty, then it means the user is in main page.
```

```
		|:	otherwise, the user has loaded a new page.
		|:	1: indicates the user has loaded a new page.
		|:	0: indicates the user is in the main page.
		-----------------------------------------------------------*/
		if(!empty($this->currentURL))
			return 1;
		else
			return 0;
	}
	//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
	public function printFinalXVOutput()
	{
	/*-------------------------------------------------------------------
		|:	this function combines the output from JS generator and
		|:	Grammar Generator then the $this->finalOutput variable is
populated.
	-------------------------------------------------------------------------*/
		header('content-type: text/xml');
		header("Cache-Control: no-cache, must-revalidate");
		 // Date in the past
		header("Expires: Mon, 28 Jul 1997 05:00:00 GMT");
		error_reporting(E_ALL);
		require_once("getWeb.php");
		require_once("GrammarManager.php");
		require_once("JSManager.php");

		$gram = new GrammarManager();
		$jss = new JSManager();

		$urlx = "";
		if(isset($_REQUEST["url"]))
		{
			$urlx = $_REQUEST["url"];
		}
```

```
print "<html xmlns=\"http://www.w3.org/1999/xhtml\" xmlns:ev=\"http://www.w3.org/2001/xml-events\">";
print "<head>";
print "  <title>WebVoice: Voice access to web</title>";
print "  <form xmlns=\"http://www.w3.org/2001/vxml\" id=\"weburlform\">";
?>
        <field name="weburls">
        <?php
                if(!empty($urlx))
                {
                        $webx = new getWeb();
                        $webx->getWeb($urlx);
                }

                $strGram = $gram->printGrammar();
                if(!empty($urlx))
                {
                        $strGram = str_replace("remain | silence | working by| believe", $webx->printGrammar(),$strGram);
                }
                if (isset($webx))
                {
                        $link_count="";
                        for($i=1; $i<=$webx->countLinks(); $i++)
                        {
                                $link_count .= "| $i";
                        }
                        if(!empty($link_count))
                        {
                                $strGram = str_replace("<digit> = 1|2|3|4|5|6|7|8|9|10", "<digit> = 0".$link_count,$strGram);
                        }
                }
```

```
                print $strGram;
            ?>
              <prompt timeout="40s">
              <?php
                if(empty($urlx))
                {
                ?>
                        Welcome to Web Voice. Please tell me an U R L?
                <?php
                }
                else
                {
                        if($webx->urlFailedToOpen())
                        {}
                        else
                        {
                                $title = $webx->printTitle();
                                print "Web Voice has loaded: $title. For a
list of speech commands please say HELP.";
                        }
                }
                ?>
              </prompt>
              <catch event="vxmldone">
                        <value   expr="load_link_js(application.lastresult$)"
/>
                        <clear namelist="weburls" />
              </catch>

              <catch event="vxmldone2">
                        Shutting down.
              </catch>

              <catch event="help">
```

```
                        You may say:
                        How many Links?
                        How Many paragraphs?
                        Find the paragraph with the word: x.y.z.
                        Read All Paragraphs:
                        Read Paragraph 1:
                        Load Link 1:
                        Load Default Link 1:
                        Go To Php.net:
                        Webvoice Home:
                        Shut Down:
                        Read Title:
                        <clear namelist="weburls" />
                    </catch>

                    <catch event="noinput">
                        I am not detecting any input from your microphone.
                    </catch>

                    <catch event="nomatch">
                        I did not understand what you said. Sorry. Please try again.
                        <clear namelist="weburls" />
                    </catch>
                    <catch event="someother">
                            <clear namelist="weburls" />
                            <prompt>How about orange juice? or simple
water?</prompt>
                    </catch>
                    <catch event="answerhowmany">
                        <clear namelist="weburls" />
                        <!-- We send the lastresult$ variable to the voice_done() JS
function. -->
                        <assign                          name="how_many_count"
expr="answer_how_many(application.lastresult$);" />
```

```
                        <prompt>There are <value expr="how_many_count" /></prompt>
                    </catch>
                    <catch event="answer_read">
                        <clear namelist="weburls" />
                        <!-- We send the lastresult$ variable to the voice_done() JS
function. -->
                        <assign                                    name="read_txt"
expr="answer_read_js(application.lastresult$);" />
                        <prompt><value expr="read_txt" /></prompt>
                    </catch>

                    <catch event="answer_find">
                        <clear namelist="weburls" />
                        <!-- We send the lastresult$ variable to the voice_done() JS
function. -->
                        <assign                                    name="read_para"
expr="answer_find_js(application.lastresult$);" />
                        <prompt><value expr="read_para" /></prompt>
                    </catch>
                    <catch event="load_default_link">
                        <clear namelist="weburls" />
                        <!-- We send the lastresult$ variable to the voice_done() JS
function. -->
                        <assign                              name="no_need_to_count"
expr="load_default_link_js(application.lastresult$);" />
                    </catch>
                    <catch event="load_link">
                        <clear namelist="weburls" />
                        <!-- We send the lastresult$ variable to the voice_done() JS
function. -->
                        <assign                              name="no_need_to_count"
expr="load_link_js(application.lastresult$);" />
                    </catch>
                    <catch event="load_home">
                        <clear namelist="weburls" />
                        <assign name="no_need_to_count" expr="load_home_js();" />
```

```
  </catch>
  <catch event="load_last">
        <clear namelist="weburls" />
        <assign name="no_need_to_count" expr="load_last_js();" />
  </catch>
<filled>
        <if cond="weburls == 'Nothing' || weburls == 'stop it' ">
                <throw event="vxmldone2" />
        <elseif cond=" weburls == 'web voice home' " />
                <throw event="load_home" />
        <elseif cond=" weburls == 'Go Back' " />
                <throw event="load_last" />

        <elseif cond="weburls.the_url !='' &amp;&amp; q_type==''" />
                <throw event="vxmldone" />
        <elseif cond="weburls == 'Something Else' " />
                <throw event="someother" />
        <elseif cond=" q_type=='How Many' " />
                <throw event="answerhowmany" />
        <elseif cond=" q_type=='Read' " />
                <throw event="answer_read" />
        <elseif cond=" q_type=='Find' " />
                <throw event="answer_find" />
        <elseif cond=" q_type=='Load Default' " />
                <throw event="load_default_link" />
        <elseif cond=" q_type=='Go To' || q_type=='Load' " />
                <throw event="load_link" />

        <else />
                <!--Do Nothing-->
        </if>
</filled>

 </field>
```

```
  </form>
<?php
	$jss->printJS($webx, $urlx);
?>
</head>
<body ev:event="load" ev:handler="#weburlform">
<h1>Chandon's WebVoice</h1><small>V0.00063</small>
<p>say an url like: php.net, bbc.com or wikipedia.org</p>
<p>You May Say :
	<ol>
		<li>Help</li>
		<li>How many Links?</li>
		<li>Load link 2.</li>
		<li>How many paragraphs?</li>
		<li>Read Paragraph 2. </li>
		<li>Read All Paragraphs. </li>
		<li>Read Link 3.</li>
		<li>Find the paragraph with the word "believe".</li>
		<li>Find the link with the word "believe".</li>
		<li>Load default link 1 (There are 4 default links)</li>
		<li>Go to php.net</li>
		<li>Webvoice Home.</li>
		<li>Shutdown.</li>
		<li>Read Title.</li>
	</ol>
</p>

<p>
Please not this page requires the following browser
<a
```
href="http://arc.opera.com/pub/opera/win/923/en/Opera_9.23_Eng_Setup.exe">Opera 9.23 with voice pack enabled.

The voice pack is installed after the Opera browser has been installed. To install the voice pack, open the Opera browser.

```
		Then go to Tools>Preferences and select the "Advanced" tab.

		Then click the option called voice and put a check mark on the checkbox labeled "Enable Voice-Controlled Browsing".

		It will ask you for downloads and then download it.

		</p>

		</body>

		</html>

		<?php

	}

}

?>
```

Section 4. Transformer

```
<?php

	class getWeb
	{
		var $x_url;
		var $x_content;
		var $base_url;

		private $failedToOpen;

		function __construct()
		{
			$this->x_url="";
			$this->x_content="";
			$this->base_url="";
			$this->failedToOpen=false;
		}
		//-------------------------------------------------
		//		gets the url and puts it in x_content var
		//-------------------------------------------------
		function getWeb($ax)
		{
			$options = array
			(	'http' => array
				(
				 'user_agent'    => 'spider',    // who am i
				 'max_redirects' => 10,          // stop after 10 redirects
				 'timeout'       => 120,         // timeout on response
```

```
                )
        );
                $context = stream_context_create($options);
                $this->setUrl($ax);
                $handle = fopen($this->x_url, "r", false, $context);
                $this->x_content="";
                if ($handle)
                {
                        while (!feof($handle))
                        {
                                $this->x_content .= fgets($handle, 4096);
                        }
                        fclose($handle);
                        //extract the base url
                        preg_match("#http\://([a-zA-Z0-9\-\.]+)(\.)([a-zA-
Z]{2,3})(/.)*#", $this->x_url, $matches);
                        $this->base_url                                         =
"http://".$matches[1].$matches[2].$matches[3]; //because $matches[0] contains /php.net


                        if(stripos($this->x_content, "failed to open stream:
HTTP request failed! HTTP/1.0 403 Forbidden")!==false)
                        {
                                //we have encountered a 403 error
                                print "<prompt>The URL is Forbidden. Failed
to open: ".$this->x_url.". Please try again.</prompt>";
                                $this->failedToOpen=true;
                        }
                }
                else
                {
                        print "<prompt>Failed to open: ".$this->x_url.". Please try
again.</prompt>";
                        $this->failedToOpen=true;
                }
        }
        //-------------------------------------------------------
        //              (END OF)gets the url and puts it in x_content var
        //-------------------------------------------------------
        function urlFailedToOpen()
        {
                return $this->failedToOpen;
        }
        //-------------------------------------------------
        //              returns x_content var after stripping the tags
        //-------------------------------------------------
```

```
		function printContent()
		{
			return $this->strip_html_tags($this->x_content);
		}
		//------------------------------------------------
		//			(END OF)returns x_content var after stripping the tags
		//------------------------------------------------
		//------------------------------------------------
		//			returns x_content var
		//------------------------------------------------
		function printRawContent()
		{
			return $this->x_content;
		}
		//------------------------------------------------
		//			(END OF)returns x_content var
		//------------------------------------------------
		//------------------------------------------------
		//			Sets the x_url var
		//------------------------------------------------
		function setUrl($a)
		{
			$this->x_url = $a;
			$pos = stripos($this->x_url, "http");
			if($pos===false)
				$this->x_url = "http://".$this->x_url;
		}
		//------------------------------------------------
		//			(END OF)Sets the x_url var
		//------------------------------------------------
/**
* Remove HTML tags, including invisible text such as style and
* script code, and embedded objects.  Add line breaks around
* block-level tags to prevent word joining after tag removal.
* function taken from:
* http://nadeausoftware.com/articles/2007/09/php_tip_how_strip_html_tags_web_page
*/
		function strip_html_tags( $text )
		{
		    $text = preg_replace(
		        array(
		          // Remove invisible content
```

```
                        '@<head[^>]*?>.*?</head>@siu',
                        '@<style[^>]*?>.*?</style>@siu',
                        '@<script[^>]*?.*?</script>@siu',
                        '@<object[^>]*?.*?</object>@siu',
                        '@<embed[^>]*?.*?</embed>@siu',
                        '@<applet[^>]*?.*?</applet>@siu',
                        '@<noframes[^>]*?.*?</noframes>@siu',
                        '@<noscript[^>]*?.*?</noscript>@siu',
                        '@<noembed[^>]*?.*?</noembed>@siu',
                      // Add line breaks before and after blocks
                        '@</?((address)|(blockquote)|(center)|(del))@iu',
                        '@</?((div)|(h[1-9])|(ins)|(isindex)|(p)|(pre))@iu',
                        '@</?((dir)|(dl)|(dt)|(dd)|(li)|(menu)|(ol)|(ul))@iu',
                        '@</?((table)|(th)|(td)|(caption))@iu',
                        '@</?((form)|(button)|(fieldset)|(legend)|(input))@iu',
                        '@</?((label)|(select)|(optgroup)|(option)|(textarea))@iu',
                        '@</?((frameset)|(frame)|(iframe))@iu',
                    ),
                    array(
                        ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
                        "\n\$0", "\n\$0", "\n\$0", "\n\$0", "\n\$0", "\n\$0",
                        "\n\$0", "\n\$0",
                    ),
                    $text );
            return strip_tags( $text );
        }
        //----------------------------------------
        function printGrammar()
        {
            //-------------------------------------
            //     FIRST GET ALL THE PARAGRAPHS TEXT
            //-------------------------------------
            $alltext = $this->strip_html_tags($this->x_content);

//-------------------------------------------------------------------------
//     NOW GET ALL THE LINKS TEXT AND ADD THEM AT THE END OF PARAGRAPHS TEXT
//-------------------------------------------------------------------------
            $incFiles=$this->parseLinks();
            if (count($incFiles)>0)
            {
                for ($i=0;$i<count($incFiles);$i++)
                {
        //lets split up the links attributes and content
```

```
                $one_link = $incFiles[$i];
                $link_label="";
                $link_href="";
                $this->extractLinkElements($one_link, $link_label, $link_href);
                $alltext .= " ".$this->strip_html_tags($link_label);
                        }
                }

                //print $alltext;
                //-----------------------------------------------------
                //      NOW ITERATE THRU COMBINED TEXT
                //-----------------------------------------------------
                $mwords = explode(" ", $alltext);

                //Remove Duplicate from the array
                //-------------------------------
                $this->rem_duplicate($mwords);
                //-------------------------------
                $the_final_word="begin ";
                for($i=0;$i<count($mwords);$i++)
                {
                        $the_word = trim($mwords[$i]);

                        $the_word = preg_replace
                                                (
                                                        array('#\#|\!|^|\|&|@|/|,|-
|\(|\)|"|&|:|_|\[|\]|{|}|\.|\?|;|\*|\||\%|\=|\n|\$|\'|\+#si'),
                                                        array(''),
                                                        $the_word
                                                );
                        if(preg_match('/^[a-z0-9]+$/i', $the_word))
                        {
                                $the_word = trim($the_word);
                                if(!empty($the_word) && strlen($the_word)<20)
                                {
                                        $the_final_word.= "|".$the_word ;
                                }
                        }
                        else
                        {
                                $the_final_word.="";
                        }
```

```
		}
		return $the_final_word;
	}
	//----------------------------------------
	// Remove duplicate function
	//----------------------------------------
	function rem_duplicate(&$arrWords)
	{
		$arrOut = array();
		$arrHash = array();

		$new_array_counter=0;
		for($i=0;$i<count($arrWords);$i++)
		{
			$my_hash = hash('sha256', $arrWords[$i]);
			if(!isset($arrHash[$my_hash]))
			{
				$arrHash[$my_hash]=$arrWords[$i];//hash   result   is
used as index
				$arrOut[$new_array_counter]=$arrWords[$i];
				$new_array_counter++;
			}
		}
		//now change the pointer to point at new array
		$arrWords = $arrOut;
	}
	//----------------------------------------
	// (END OF)Remove duplicate function
	//----------------------------------------

	//----------------------------------------
	// Remove duplicate function 2
	//----------------------------------------
	function rem_duplicate2(&$arrWords)
	{
		$arrOut = array();
		$arrHash = array();

		$new_array_counter=0;
		for($i=0;$i<count($arrWords);$i++)
		{
			$strItem = $arrWords[$i];
			$arrWords[$i]="";
```

```
                                if(!in_array($strItem, $arrWords))
                                {
                                        $arrOut[$new_array_counter]=$strItem;
                                        $new_array_counter++;
                                }
                        }
                        //now change the pointer to point at new array
                        $arrWords = $arrOut;
                }
                //----------------------------------------
                // (END OF)Remove duplicate function 2
                //----------------------------------------
                function printLinks()
                {
                        $incFiles=$this->parseLinks();
                        if (count($incFiles)>0)
                        {
        print "<table border=0 width='100%' cellspacing=1 cellpadding=4 bgcolor=aaaaaa>";


for ($i=0;$i<count($incFiles);$i++)
{
print "<tr>";
print   "<td bgcolor=eeeeee align=left>$i";
print htmlentities ( $incFiles[$i] )."<hr/>";

//lets split up the links attributes and content
$one_link = $incFiles[$i];
$link_label="";
$link_href="";
$this->extractLinkElements($one_link, $link_label, $link_href);
print "[".$link_label."] -> [".$link_href."]";
print   "</td>";
print "</tr>";
}
print "</table>";
}
                        else
                        {
                                print "No Links!!<br/>";
                        }
                }
                //----------------------------------------
```

```
		function printLinksJS()
		{
			$incFiles=$this->parseLinks();
			if (count($incFiles)>0)
			{
				$j=1;
				for ($i=0;$i<count($incFiles);$i++)
				{
//lets split up the links attributes and content
$one_link = $incFiles[$i];
$link_label="";
$link_href="";
$this->extractLinkElements($one_link, $link_label, $link_href);
$link_label = $this->strip_html_tags($link_label);
$link_label = trim($link_label);
/*-----------------------------------------------------
**	links[1] = new Array();
**	links[1][0]="http://en.wikipedia.org/wiki/SpeechWeb";
**	links[1][1]="speechweb wikipedia web page";
-----------------------------------------------------*/
				if(!empty($link_label)&&!empty($link_href))
				{
				print "links[".($j)."]=new Array();\n";
				print "links[".($j)."][0]=\"$link_href\";\n";
				print "links[".($j)."][1]=\"$link_label\";\n";

				$j++;
				}
				}
			}
		}
		function extractLinkElements($one_link, &$link_label, &$link_href)
		{
	preg_match_all("/<a.*href=\"(.*)\"\s(.*?)>(.*)<\/a>/i",$one_link, $out);

			if(isset($out[1][0]))
			{
				$link_href = $out[1][0];
			}
			else
				$link_href="";
			if(isset($out[count($out)-1][0]))
			{
				$link_label = $out[count($out)-1][0];
```

```
                        }
                        else
                                $link_label="";
                        //if the content(link label) is only an image then extract the alt.
if no alt is found then dont include it.

        if(preg_match_all("/<a[^>]*?><img(.*)alt=(.*)><\/a>/si",$one_link,$out))
                        {
                                $link_label = $out[2][0];
                        }
                        $link_label = str_replace("\"","",$link_label);


                        //if the href does not contain http then its a relative link. so
resolve it by adding the base url infront of it
                        if(strpos($link_href, "http://")===false)
                        {
                                //if it contains enclosing double quotes then we remove the
double quotes
                                if(preg_match_all("/\"(.*)\"/i", $link_href, $out))
                                {

                                        $link_href                                      =
str_replace("\"","",substr($link_href,1,strpos($out[1][0],"\"")));
                                }
                                else
                                {
                                        //if it does not contains enclosing double quotes
then we do nothing
                                }

                                //if the relative link contains a back slash(/) at the
beginning then we are fine
                                //else add a back slash
                                if(strpos($link_href, "/")===false)
                                {
                                        $link_href = "/".$link_href;
                                }
                                else if(strpos($link_href,"/")>0)
                                {
                                        $link_href = "/".$link_href;
                                }
                                else
                                {
                                        //do nothing
                                }
```

```
                        //now add the refined link to the base url to make it the
complete link
                        $link_href = $this->base_url.$link_href;
                }
                else
                {
                        if(preg_match_all("/\"(.*)\"/i", $link_href, $out))
                        {                                       $link_href          =
str_replace("\"","",substr($link_href,1,strpos($out[1][0],"\"")));
                        }
                        else
                        {
                                //do nothing
                        }
                }
        }
        //----------------------------------------
        function printParagraphs()
        {
                $incFiles=$this->parseParagraph();
                if (count($incFiles)>0)
                {
                        print    "<table    border=0    width='100%'    cellspacing=1
cellpadding=4 bgcolor=aaaaaa>";
                        for ($i=0;$i<count($incFiles);$i++)
                        {
                                $cnt = "";
                                $cnt = trim($this->strip_html_tags($incFiles[$i]));
                                if(!empty($cnt))
                                {
                                        print "<tr>";
                                        print   "<td bgcolor=eeeeee align=left>";
                                        print   $cnt."<hr/>";
                                        //parseLinks ( $incFiles[$i] );
                                        print   "</td>";
                                        print "</tr>";
                                }
                        }
                        print "</table>";
                }
                else
                        print "No Paragarphs!!<br/>";
        }
        //----------------------------------------
```

```
		function printParagraphsInJS()
		{
			$incFiles=$this->parseParagraph();
			if (count($incFiles)>0)
			{
				for ($i=0;$i<count($incFiles);$i++)
				{
					$cnt = "";
					$incFiles[$i]	=	$this->remove_html_entites
($incFiles[$i] );
					$cnt = trim($this->strip_html_tags($incFiles[$i]));
					if(!empty($cnt))
					{
						print	"pgs[".($i+1)	."]
=\"".str_replace("\n"," ",htmlentities($cnt))."\";\n";
					}
				}
			}
		}
		//-----------------------------------------------
		function remove_html_entites($line)
		{
			//this function removes any html entity
		    $line = preg_replace(
		        array('@\&(.*);@siu'), array(' '),  $line );
			return $line;
		}
		//-----------------------------------------
		function printHeadings()
		{
			$incFiles=$this->parseHeadings();
			if (count($incFiles)>0)
			{
				print	"<table	border=0	width='100%'	cellspacing=1
cellpadding=4 bgcolor=aaaaaa>";
				for ($i=0;$i<count($incFiles);$i++)
				{
					print "<tr>";
					print  "<td bgcolor=eeeeee align=left>";
					print		$this->strip_html_tags($incFiles[$i])
."<hr/>";
					print  "</td>";
					print "</tr>";
				}
				print "</table>";
```

```
                }
                else
                        print "No Headings!!<br/>";
        }
        function printTitle()
        {
                $incFiles=$this->parseTitle();
                if (count($incFiles)>0)
                {
                        $result =  $this->strip_html_tags($incFiles[0]) ;
                        $result = preg_replace
                                                        (
                array('#\#|\!|^|\|&|@|/|,|-
|\(|\)|"|&|:|_|\[|\]|{|}|\.|\?|;|\*|\||\%|\=|\n|\$|\'|\+#si'),
                                                                array(''),
                                                                $result
                                                        );
                }
                return $result;
        }
        function parseLinks()
        {
                $fileData = $this->x_content;
                preg_match_all("/<a[^>]*?>.*?<\/a>/si",$fileData, $out);
                return $out[0];
        }
        function parseParagraph()
        {
        $fileData = str_replace("\r\n", " ", $this->x_content);
        preg_match_all("/<p(\s*)(.*)>(.*?)(<\/p>)/i",$fileData, $out);
        $result = $out[0];
        return $result;
        }
        function parseHeadings()
        {
        $fileData = str_replace("\r\n", " ", $this->x_content);
        preg_match_all("/<h[1-9]>(.*?)(<\/h[1-9]>)/i",$fileData, $out);
        return $out[0];
        }
        function parseLists()
        {
        $fileData = str_replace("\r\n", " ", $this->x_content);
        preg_match_all("/<(ul|ol)>(.*?)(<\/(ul|ol)>)/i",$fileData, $out);
        return $out[0];
```

```
		}
		function parseListItem($myList)
		{
			preg_match_all("/<li>(.*?)(<\/li>)/i",$myList, $out);
			return $out[0];
		}
		function parseTitle()
		{
			$fileData = str_replace("\r\n", " ", $this->x_content);
		preg_match_all("/<title>(.*?)(<\/title>)/i",$fileData, $out);
		return $out[0];
		}
		//-------------------------------------------------------------------
		function countLinks()
		{
			$fileData = $this->x_content;
			preg_match_all("/<a[^>]*?>.*?<\/a>/si",$fileData, $out);
			return count($out[0]);
		}
		function count_paragraph_words()
		{
			if(!empty($this->x_content))
			{
				$alltext = $this->strip_html_tags($this->x_content);
				$arrWords = explode(' ', $alltext);
				return count($arrWords);
			}
			else
			{
				return 0;
			}
		}
	}//class ends

?>
```

Appendix VI - Remove Duplicate Function

```
function rem_duplicate(&$arrWords)
{
	$arrOut = array();
	$arrHash = array();

	$new_array_counter=0;
	for($i=0;$i<count($arrWords);$i++)
	{
		$my_hash = hash('sha256', $arrWords[$i]);

		if(!isset($arrHash[$my_hash]))
		{
			$arrHash[$my_hash]=$arrWords[$i];
			//hash result is used as index
			$arrOut[$new_array_counter]=$arrWords[$i];
			$new_array_counter++;
		}
	}

	//now change the pointer to point at new array
	$arrWords = $arrOut;
}
```

VITA AUCTORIS

Shahriar Chandon was born in 1979 in Khulna, Bangladesh. He graduated from Dhaka Residential Model High School and College in 1996. From there he went on to the North South University of Dhaka, Bangladesh. In 1999 He transferred to University of Windsor, Windsor, Ontario and obtained a B. Sc. (General) in Computer Science in 2000. He also obtained a B. Sc. (Hons) in Computer Science in 2004. Currently, he is a candidate for M.Sc. degree in Computer Science and hopes to graduate in Summer 2009.