A Voice-based Multimodal User Interface for VTQuest

THOMAS W. SCHNEIDER

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Dr. Osman Balci, Chair
Dr. Shawn A. Bohner
Dr. Manuel A. Pérez-Quiñones

May 6, 2005

Blacksburg, Virginia

Keywords and phrases: Client/server software, human computer interaction, multi modal user interface, voice user interface
A Voice-based Multimodal User Interface for VTQuest

THOMAS W. SCHNEIDER

Abstract

The original VTQuest web-based software system requires users to interact using a mouse or a keyboard, forcing the users’ hands and eyes to be constantly in use while communicating with the system. This prevents the user from being able to perform other tasks which require the user’s hands or eyes at the same time. This restriction on the user’s ability to multitask while using VTQuest is unnecessary and has been eliminated with the creation of the VTQuest Voice web-based software system. VTQuest Voice extends the original VTQuest functionality by providing the user with a voice interface to interact with the system using the Speech Application Language Tags (SALT) technology. The voice interface provides the user with the ability to navigate through the site, submit queries, browse query results, and receive helpful hints to better utilize the voice system. Individuals with a handicap that prevents them from using their arms or hands, users who are not familiar with the mouse and keyboard style of communication, and those who have their hands preoccupied need alternative communication interfaces which do not require the use of their hands. All of these users require and benefit from a voice interface being added onto VTQuest. Through the use of the voice interface, all of the system’s features can be accessed exclusively with voice and without the use of a user’s hands. Using a voice interface also frees the user’s eyes from being used during the process of selecting an option or link on a page, which allows the user to look at the system less frequently. VTQuest Voice is implemented and tested for operation on computers running Microsoft Windows using Microsoft Internet Explorer with the correct SALT and Adobe Scalable Vector Graphics (SVG) Viewer plug-ins installed. VTQuest Voice offers a variety of features including an extensive grammar and out-of-turn interaction, which are flexible for future growth. The grammar offers ways in which users may begin or end a query to better accommodate the variety of ways users may phrase their queries. To accommodate for abbreviations of building names and alternate pronunciations of building names, the grammar also includes nicknames for the buildings. The out-of-turn interaction combines multiple steps into one spoken sentence thereby shortening the interaction and also making the process more natural for the user. The addition of a voice interface is recommended for web applications which a user may need to use his or her eyes and hands to multitask. Additional functionality which can be added later to VTQuest Voice is touch screen support and accessibility from cell phones, Personal Digital Assistants (PDAs), and other mobile devices.
Acknowledgments

I would like to thank God for providing me with all the opportunities in my life. My collegiate career has been full of ups and downs but with the aid and encouragement of everyone close to me I was able to persevere. In particular, I want to thank my grandmother and parents for helping support me financially throughout my college career. I want to thank Elizabeth for believing in me and standing with me while still being hundreds of miles away this whole year. The care packages and bimonthly visits were great boosts of encouragement. I want to thank my professors at Virginia Tech for challenging me and teaching me the skills necessary to become the computer scientist that I am today, particularly my mentor Dr. Balci for advising me through this whole thesis process and through the last two years of completing the five year program. Finally, I want to thank all my friends for giving me the much needed distractions and all the laughs along the way which I will always remember.
# Table of Contents

Abstract ................................................................................................................................................................................. ii

Acknowledgments ........................................................................................................................................................................ iii

List of Figures ........................................................................................................................................................................ vi

List of Tables ........................................................................................................................................................................ vii

List of Acronyms ....................................................................................................................................................................... viii

Chapter 1: Introduction ............................................................................................................................................................. 1
  1.1 STATEMENT OF THE PROBLEM .............................................................................................................................. 4
  1.2 STATEMENT OF OBJECTIVES .................................................................................................................................. 5
  1.3 OVERVIEW OF THESIS .............................................................................................................................................. 5

Chapter 2: Literature Survey ..................................................................................................................................................... 6
  2.1 BENEFITS AND LIMITATIONS OF VOICE INTERFACES ..................................................................................... 6
    2.1.1 Benefits .......................................................................................................................................................... 6
    2.1.2 Limitations .................................................................................................................................................. 6
  2.2 VOICE INTERFACE GUIDELINES ............................................................................................................................. 7
  2.3 INFORMATION VISUALIZATION ................................................................................................................................. 8
  2.4 VOICE TECHNOLOGIES FOR WEB-BASED SOFTWARE SYSTEMS ........................................................................ 9

Chapter 3: Design and Implementation ................................................................................................................................. 11
  3.1 HARDWARE AND SOFTWARE ENVIRONMENT ...................................................................................................... 11
  3.2 VTQUEST IMPROVEMENTS .................................................................................................................................... 11
  3.3 IMPLEMENTATION OF THE VOICE INTERFACE ..................................................................................................... 13
    3.3.1 Voice Technology Chosen .................................................................................................................................... 14
    3.3.2 Design Issues Considered ................................................................................................................................... 14
    3.3.3 Subsystems Accessible at Every Listener ........................................................................................................ 15
      3.3.3.1 Navigation Subsystem .............................................................................................................................. 15
      3.3.3.2 Hint Subsystem .......................................................................................................................................... 16
    3.3.4 Components Utilized on Multiple Pages ......................................................................................................... 19
      3.3.4.1 Building and Category Data Structure ................................................................................................... 19
      3.3.4.2 Viewing Building Names ......................................................................................................................... 20
      3.3.4.3 Determining Categories to View ........................................................................................................... 21
      3.3.4.4 Exploring the Map ................................................................................................................................... 22
    3.3.5 Out of Turn Interaction .................................................................................................................................... 23
    3.3.6 Errors ............................................................................................................................................................... 24
      3.3.6.1 Avoiding Errors ........................................................................................................................................ 24
      3.3.6.2 Handling Errors ....................................................................................................................................... 25
    3.3.7 Pages with Only Links ....................................................................................................................................... 25
    3.3.8 Find by Building Name ....................................................................................................................................... 26
    3.3.9 Find by Abbreviation ......................................................................................................................................... 27
    3.3.10 Find by Category ............................................................................................................................................ 28
    3.3.11 Find Category within Distance around a Building .......................................................................................... 29
    3.3.12 Get Directions ............................................................................................................................................. 30
  3.4 DESIGN ISSUES FOR THE VOICE INTERFACE ..................................................................................................... 32

Chapter 4: Verification and Validation ..................................................................................................................................... 33
  4.1 VERIFICATION ............................................................................................................................................................ 33
    4.1.1 Desk Checking .................................................................................................................................................. 33
    4.1.2 Documentation Checking .................................................................................................................................. 33
    4.1.3 Structured Walkthroughs ................................................................................................................................... 33
    4.1.4 Inspections ....................................................................................................................................................... 34
  4.2 VALIDATION ............................................................................................................................................................... 34
List of Figures

Figure 1.1 – Original VTQuest find by name page with floor level drop down 1
Figure 1.2 – Original VTQuest find by abbreviation page 2
Figure 1.3 – Original VTQuest find by category page 2
Figure 1.4 – Original VTQuest find within distance page 3
Figure 1.5 – Original VTQuest browse campus map page 4
Figure 3.1 – Campus map centered on McBryde Hall showing the new navigation buttons 13
Figure 3.2 – Examples in the hint for the Find Building by Name page 17
Figure 3.3 – Multiple hints on Find within Distance page 18
Figure 3.4 – Hint Subsystem state diagram which is a smaller part of 19
Figure 3.5 – Choosing a building 20
Figure 3.6 – Building names component state diagram which is a smaller part of 21
Figure 3.7 – Adding and Removing Categories to view 21
Figure 3.8 – Determining category state diagram which is a smaller part of 22
Figure 3.9 – Browse map state diagram which is a smaller part of 23
Figure 3.10 – Page with only links state diagram 26
Figure 3.11 – Find by building name state diagram 27
Figure 3.12 – Find by abbreviation state diagram 28
Figure 3.13 – Find by category state diagram 29
Figure 3.14 – Find within distance state diagram 30
Figure 3.15 – Get directions state diagram 31
List of Tables

Table 3.1 – Navigation commands with the actions taken when heard 16
Table 3.2 – Commands to control the map movements 22
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASV</td>
<td>Adobe SVG Viewer</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise JavaBean</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>InkML</td>
<td>Ink Markup Language</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Platform, Enterprise Edition</td>
</tr>
<tr>
<td>MS-IE</td>
<td>Microsoft Internet Explorer</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>SALT</td>
<td>Speech Application Language Tags</td>
</tr>
<tr>
<td>SVG</td>
<td>Scalable Vector Graphics</td>
</tr>
<tr>
<td>TTS</td>
<td>Text To Speech</td>
</tr>
<tr>
<td>VoiceXML</td>
<td>Voice Extensible Markup Language</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WAS</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>X+V</td>
<td>XHTML+Voice</td>
</tr>
<tr>
<td>XHTML</td>
<td>Extensible HyperText Markup Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

VTQuest was earlier developed as a fully functional client/server web-based software system using the Java 2 Platform, Enterprise Edition (J2EE) architecture. VTQuest is a map system for the Virginia Tech campus allowing users to search for buildings in a variety of ways and to find directions between two buildings. A user can search for a building by name (Figure 1.1), by abbreviation (Figure 1.2), by category (Figure 1.3), or categories within a radius around a particular building (Figure 1.4). The shortest-path algorithm is used to determine directions between two buildings. The user can also browse the map to look around the campus as shown in Figure 1.5. VTQuest currently can be accessed at http://albatross.cslab.vt.edu/VTQuest/.

![Figure 1.1 – Original VTQuest find by name page with floor level drop down selected and hint visible](image)
Figure 1.2 – Original VTQuest find by abbreviation page

Figure 1.3 – Original VTQuest find by category page
Figure 1.4 – Original VTQuest find within distance page
4

Figure 1.5 – Original VTQuest browse campus map page

The VTQuest Voice web-based software system is built on top of the VTQuest system and equips VTQuest with a voice interface which provides the user the ability to interact hands free eliminating the requirement of using a mouse or keyboard. This voice interface gives the user the ability to communicate with the system using voice, a mouse, or a keyboard to enter input, making VTQuest Voice a multimodal system.

1.1 Statement of the Problem

Many web-based software systems exist, but few provide a multimodal user interface. A user of VTQuest is restricted to the default form of communicating with the system using the mouse or keyboard. However, some users prefer to have more than this option while other users require additional options. Individuals with a handicap that prevents them from using their arms or
hands, users who are not familiar with the mouse and keyboard style of communication, and those who have their hands preoccupied need alternative communication interfaces which do not require the use of their hands. All of these users require and benefit from a voice interface being added onto VTQuest.

1.2 Statement of Objectives

The objective of the VTQuest Voice system is to give the original VTQuest system the flexibility to handle voice interaction so that a mouse and keyboard would not be required for all basic interactions with the system. This requires a voice interface for navigating the site, a voice interface to input queries to the site, and a voice interface to navigate the browsing of the map. By providing these voice interfaces, the system fully functions without the need to use a mouse or a keyboard. Although the system works without a mouse or a keyboard, the mouse and keyboard inputs are still available for the user along with the voice interface so the user can use whichever form of communication is desired. Having this technology available will create a good foundation to port the web-based software system to other devices such as a Personal Digital Assistant (PDA) or for use inside an automobile.

1.3 Overview of Thesis

First a survey of the existing literature for current web-based voice technologies available along with voice design guidelines and juxtaposition is presented. Next the design and implementation of the VTQuest Voice system and how it is built on top of the original VTQuest system are described. Then the verification and validation techniques used to assess the accuracy of the VTQuest Voice system are explained. Finally conclusions are presented including what future work can be done by extending VTQuest Voice.
Chapter 2: Literature Survey

This chapter surveys the voice user interface systems literature to analyze what is currently available and being researched mainly for web-based software systems and the guidelines to use to create a well formed voice interface. First we go over the benefits and limitations of voice interface systems and see how their limitations are being resolved. Next we overview the main technologies used to create voice interface systems to be integrated with web-based software systems. And finally we overview voice interface guidelines for systems which also have a visual interface.

2.1 Benefits and Limitations of Voice Interfaces

Voice interface systems have advantages and disadvantages over other forms of input depending on what is desired and needed by the user. These attributes should be weighed before deciding whether or not a voice interface would work well with an existing web-based software system.

2.1.1 Benefits

The most obvious and important benefit for a voice user interface is the ability for the user to communicate with the system by speaking. The reason why this is a benefit is because speaking is the most natural form of communication and humans have nearly a lifetime worth of experience with speaking. Speech offers a straightforward and natural form of input for unplanned thoughts and ideas [Stifelman et al. 1993].

Being able to communicate with only speaking and listening frees the user’s hands and eyes to do other tasks while interacting with a voice interface. This is particularly useful in dual task situations when the user’s hands and eyes are busy with another task [Cohen 1995]. If the tasks are all interrelated, this provides a multimodal experience for the user since he or she is able to use multiple forms of input at the same time to complete the task.

Ability to speak out of turn is another benefit if it is supported by the system. This places less cognitive demands on the user and allows more attention to be devoted to the content of the users’ input [Stifelman et al. 1993]. Being able to speak out of turn, or using mixed interaction, makes the communication between user and system more natural and gives much more control to the user.

2.1.2 Limitations

Background noise can interfere with the voice recognition process. Noise not generated by the user can cause the system to misinterpret what a user is saying, or even be interpreted when the user is not saying anything. This limitation is caused because voice interface systems cannot distinguish between input voice and other noise [Chino et al. 2000]. Fortunately, ways to work around the presence of background noise are being researched by [Barker et al. 2000; Green et al. 2001]. The work done by Barker et al. [2000] allows the system to recognize which noise is background with background noise which is constantly present. The work done by Green et al.
[2001] does not assume any particular type of background noise and uses multiple sources of input and combines them to receive the best response.

The system misinterpreting a user response can also happen when there is no background noise due to speech recognition systems being inherently error prone [Hauptmann 1995]. Misinterpretations can be caused by phrases sounding very similar, by a user having a heavy accent, or because of a limited vocabulary size of what the system understands and can respond to.

The client’s system hardware requirements would need to include a microphone and speakers in order to input with the microphone and receive responses through the speakers. Without the combination of both of these devices, proper voice interaction with a voice user interfaces is all but impossible.

2.2 Voice Interface Guidelines

Before going over voice interface guidelines in particular, first let us go over the eight golden rules [Shneiderman 1998] of interface design which apply to most interactive systems and apply them to voice interfaces.

The first rule is to strive for consistency. Since the user can not see the grammar in which they are constrained to use in most cases, they are not aware what is allowed and what is not [Pakucs 2002]. This can become a difficulty when users are trying to identify available voice commands to use as they move from page to page even if the pages seem to have the same components. As pointed out by James [2000], this is very important because if a user has the ability to perform an action, it is assumed he or she has the ability to perform an action which is closely related. An example of this would be if the user has the ability to say “scroll down”, then the user should have the ability to say “scroll up”. In this scenario, if the user tries to scroll up and the system does not support this feature, then the user may think that the system can not scroll down either which would be an incorrect assumption.

The second rule is to enable frequent users to use shortcuts. As Frostad [2003] explains, experienced users should have the ability to step over trivial steps and steps which are used for new user instructions. The system should allow barge-ins except during confirmation, error recovery, and agent handoffs. Also, experienced users should be able to utilize abbreviations and hidden commands [Huang 1997].

The third rule is to offer informative feedback. For every action performed by the user, there should be a corresponding reaction, or feedback, provided by the system. This feedback should be informative yet short in time so not to prolong the interaction since time is the most valuable commodity in a voice interaction [Stifelman et al. 1993]. When the system is offering feedback, whenever possible the user should be allowed to interrupt the prompt. When it is not possible to interrupt the prompt, the voice recognizer should start right when the prompt ends, or some audible signal should be used to indicate when speech may begin [Fraser 1994], but either way, consistency must be maintained.
The fourth rule is to design dialogs to yield closure. Each dialog, or transaction, should have a beginning, middle, and end state. Once the transaction is completed, the user should have a sense of accomplishment. The transaction could be a complete end-to-end transaction or a transaction to complete one page and at the end the user should have an indication that he or she is able to start a new transaction. As Lea [1994] states, tasks should be created out of smaller pieces, with frequent closure points. This enables the user not to remember as much and disruptions are minimized.

The fifth rule is to offer error prevention and simple error handling. For a voice interface, this is difficult to accomplish completely, but can be attempted by providing feedback for every response so that the user knows the machine received the input and what the system is doing with that input. To avoid recognition errors when the user is speaking to something other than the system, the system should have the ability to turn the voice recognition off and on [Turunen 1998]. To prevent errors, voice interfaces should use commands that sound different so that the voice recognizer does not accidentally mistake one for the other.

The sixth rule is to permit easy reversal of actions. If a mistake has been made or the user decides to take a different course of action, the user should easily be able to undo what has been done and reenter corrected information. Being able to easily undo an action enables users to use unfamiliar options without having the fear of encountering a situation which the user can not return out of [Krahmer 2001].

The seventh rule deals with supporting internal locus of control. This means that the system should give control of the interaction to the user and allow the user to take the initiative and control the interaction. This follows the rule by Gaines [1981] which encourages users to be the initiators and not the responders.

The eighth and final rule is to reduce short-term memory load. This rule is even more important for voice-only interfaces than for graphical user interfaces (GUIs) due to the time it takes to verbally convey all the information on a page compared to being able to look over the page. The user should not have to remember instructions from the beginning of the dialog during each succeeding turn [Fraser 1994]. The prompts given to the user should be short as discussed in the third rule not only to save time, but also to reduce the amount of information that the user must hold in short-term memory [Stifelman et al. 1993].

2.3 Information Visualization

Shneiderman [1996] advocates the Visual Information Seeking Mantra which states “Overview first, zoom and filter, then details-on-demand”. The first step is to overview, where all the information being used is presented. Next is the zoom and filter step. Filtering gets rid of unnecessary items from the information which is presented in the overview step. Zooming pin points the items of interest from the overview step. After the items of interest have been selected, details of the items can be obtained by getting the details-on-demand. Three other tasks exist which accompany this mantra according to Shneiderman. The first is to relate or view relationships among items. The second is to keep a history of actions by the user so that they can
be undone when necessary. The last is to extract the results so that they can be used at a later time.

This mantra has been altered by Pérez-Quiñones et al. [2003] for voice browsing without a visual interface to be “Situate, Navigate, Query, Details-on-demand”. The change is made because the current mantra by Shneiderman [1996] “makes extensive use of the tremendous visual perceptual abilities that humans have” and “a voice-only interaction channel lacks the visual bandwidth to transfer large amounts of data in parallel” [Pérez-Quiñones et al. 2003].

2.4 Voice Technologies for Web-based Software Systems

Two established technologies which integrate a voice interface into web-based software systems are Speech Application Language Tags (SALT) and Extensible HyperText Markup Language (XHTML) + Voice (X+V). These two competing technologies have many of the same attributes such as being “markup languages for creating applications that use voice input (speech recognition) and output (speech synthesis)” [Wilson 2003] for integration with a web interface. They both do a good job dealing with the “encapsulation of the hardware, Automatic Speech Recognition (ASR), Text To Speech (TTS), telephony and other platform issues” [Coin 2002]. However they were both created with different design goals in mind and from different heritages [Potter and Larson 2002].

SALT is a multimodal application language which was created to challenge Voice Extensible Markup Language (VoiceXML) by targeting multiple devices such as PC, handhelds, telephones, and others. With the goal of being accessible to multiple devices, its target platform is for web-based software systems. The main focus of SALT is on the speech interface and it therefore does not have many tags, only four top-level element tags with five children element tags [Microsoft 2005b]. With SALT targeting multiple devices, each device needs to be able to run the SALT script the same way. SALT accomplishes this by being able to run inside different Web execution environments [Potter and Larson 2002] on different devices. Therefore it is up to each device to support all of the SALT functionality in its own execution environment. SALT extends existing mark-up languages such as HTML, XHTML, and Extensible Markup Language (XML) using the Document Object Model (DOM) [SaltForum 2005; Potter and Larson 2002]. This makes it easier to add SALT functionality into existing web pages by using the existing execution model and event structure of the page and just adding SALT tags to perform the voice operations which correspond to the existing HTML and JavaScript events. This allows for a straightforward incorporation of speech [Potter and Larson 2002] into existing web-based software systems to make them multimodal.

X+V combines XHTML, VoiceXML, and XML Events [Wilson 2003] which are all World Wide Web Consortium (W3C) standardized languages and mature when compared to SALT. Building off an existing speech technology such as VoiceXML makes it easy for the X+V technology to enhance existing VoiceXML applications and convert them to become web-based software systems. However in doing this, the X+V systems still kept the same limitations which hinder VoiceXML. VoiceXML is designed for telephony applications. VoiceXML has specific elements which control the dialog flow [Moraes 2004] that the application is required to use and follow. The way in which VoiceXML handles the events of the flow is restrictive with the event
handling being single-threaded [Hocek 2002]. With this limitation to VoiceXML event handling, it impedes the event process to only handle synchronous events that occur only when the application resides in a specific state [Moraes 2004]. This makes VoiceXML not “well suited to an environment that needs to handle asynchronous events external to the VoiceXML application” [Moraes 2004]. This makes it difficult for the execution of the application to break out of the VoiceXML event structure to allow the developer to perform multiple actions at once and requires an existing web-based software system to change its event structure to fit to the one set by VoiceXML.
Chapter 3: Design and Implementation

3.1 Hardware and Software Environment

The VTQuest Voice is a client/server web-based software system built based on the J2EE architecture. VTQuest Voice runs under the IBM HTTP Server and WebSphere Application Server (WAS) version 5.1. The IBM HTTP Server is used in conjunction with WAS because WAS could not host the campus map Scalable Vector Graphic (SVG) file due to size constraints. However, WAS is still needed to host the JSP pages and Servlets, to perform session management, and to host the Enterprise JavaBean (EJB) container. The information used by the original VTQuest system to respond to user queries is stored on a DB2 Universal Database which is a Relational Database Management System (RDBMS). The information stored on DB2 is mapped to the WAS EJB container to perform the persistence of the VTQuest data in memory and to the file system used by both the VTQuest and VTQuest Voice systems. WAS and DB2 run on an IBM xSeries 225 Server running dual 2.4 GHz Intel Xeon processors with 1.5GB of memory and 145 GB of disk space running Windows Server 2000. VTQuest Voice currently can be accessed at http://sunfish.cs.vt.edu/VTQuestV/.

The VTQuest Voice client system is available to any browser which has the SVG 3.02 plug-in from Adobe and the SALT 1.0 plug-in from Microsoft. The SVG plug-in allows a user to view the campus map and floor plans and the SALT plug-in allows a user to interact verbally with the system running on Microsoft Windows XP or 2003 with Speech Tools installed. At the time of this writing, the only browser which fits these requirements is Microsoft Internet Explorer (MS-IE) version 6.0 [Microsoft 2005a] with the Adobe SVG Viewer (ASV) plug-in version 3.02 [Adobe 2005] and the SALT 1.0 plug-in installed. A client-system must have a microphone for audio input, a sound card for audio reproduction, and speakers for audio output in order for the user to be able to interact with the system with his or her voice. VTQuest Voice works without a microphone, sound card, speakers, or on other browsers which only has the ASV plug-in but the voice interaction is inaccessible.

The client-system is encouraged to train the Microsoft Speech Tools [Microsoft 2004]. The SALT plug-in attached to the browser uses the Microsoft Speech Tools so by providing voice training, the Speech Engine more accurately understands the user’s responses. It does not take very long to complete and can greatly help if the user has any type of accent.

3.2 VTQuest Improvements

VTQuest Voice is built on top of an existing web-based software system named VTQuest and described in Chapter 1. VTQuest is a Java-based enterprise web-based software system and has many of the same features and capabilities of the VTQuest Voice system except without a voice interface. The voice interface is the most significant improvement upon the original VTQuest system that VTQuest Voice made, but it is not the only one.
With the addition of a voice interface, the VTQuest Voice system transforms VTQuest into a multimodal web-based software system. A user can use voice, a mouse, and/or a keyboard to interact with the application on any page to accomplish any central task. Providing a voice interface offers all the benefits that are associated with voice applications. These include the ability to interact with the system using speech, the most natural form of communication to most humans, the ability to free the users’ hands and eyes to accomplish multiple tasks at once, and the ability to accomplish a multiple step task in one sentence and are discussed in more detail in section 2.1.1. With these benefits, the web page becomes more usable, especially for users who have a necessity to interact with their voice.

An improvement to VTQuest which did not directly relate to the addition of the voice interface is the way in which the map is displayed to the user. In the VTQuest system, the map is displayed to the user at a default starting location and then the user is required to move the map around to locate the buildings that the system found from his or her query by finding all the blinking buildings. VTQuest Voice improves upon this functionality by calculating all of the coordinates of the selected buildings to be viewed and creates a viewing square around them so that every building is visible when the map is displayed. If only one building is chosen, the map centers on the building at the second to lowest zoom level. If multiple buildings are selected, a window is created that fits around all of the selected buildings. The map centers on the center of the viewing window encompassing all the selected buildings, and the starting zoom level is determined by how far away the map needs to be presented to fit all of the buildings into view. An example of how a single building is centered is shown in Figure 3.1.

Another improvement on the display of the map to which the VTQuest Voice system makes is by creating a navigational interface for moving the map around. The VTQuest system has no visible controls to allow the user to realign the map. VTQuest only utilized the ASV manipulation of SVG images by using the mouse and holding the control, shift or alt keys. Holding the control key turned the cursor into a magnifying glass with a plus to allow the image to be zoomed in on the image. Holding the control and shift keys together turned the cursor into a magnifying glass with a minus allowing the image to be zoomed out. A user could also accomplish zooming in and out by right clicking on the SVG image and choosing these options on the drop down menu. To move the map around, the user has to hold the alt key and the left mouse button on the map and drag the map around by moving the mouse. Without reading the help file or having prior knowledge of how the ASV worked, the user is stuck with what could seem to be a static image. The visual navigation added by VTQuest Voice places controls on the map page so that the user can click on buttons now to perform the features which were only available before by using the ASV manipulation of SVG images. Arrow buttons on each side of the map allows the user to move the map in any direction. A zoom bar to show which zoom level is currently being displayed along with the ability to choose a different level or to zoom in or out just one level is to the right of the map. These new navigation buttons are shown in Figure 3.1.
3.3 Implementation of the Voice Interface

Before implementing the voice interface on top of the VTQuest system, several design issues are considered concerning the interaction execution of each page and with how to handle and avoid errors.
3.3.1 Voice Technology Chosen

The voice technology chosen is SALT. The primary reason for selecting SALT is compatibility. SALT is the only voice technology which works together in the same browser with the other technologies already embedded in the VTQuest system. These technologies already require the VTQuest system to be used on a browser which supports the ASV plug-in along with JavaScript manipulation of the SVG image. The only browser which handles all of these technologies is MS-IE 6.0 or higher. Since the only voice technology which MS-IE 6.0 supports fully is SALT, it made the decision very easy.

Luckily this restriction did not force the development into the wrong technology, as other reasons exist which make SALT a superior choice over X+V. SALT is much easier to integrate into an existing web-based software system than X+V. Since VTQuest Voice is built on top of an existing web-based software system this is very helpful. SALT does not require the flow of the interaction to change. The flow of interaction already in place only needs to be extended. SALT extends HTML by using the DOM which makes it very easy to use with JavaScript. JavaScript is also utilized in the original user interaction with the VTQuest system so adding onto the existing code is not a problem. The ability to extend the existing interaction instead of having to redo the interaction process to fit the VoiceXML model is very helpful.

If X+V had been chosen, the required browser would have been Opera. This is an issue because as of April 1st, 2005, MS-IE based browsers are used by 84% of the internet population and Opera is only used by 2% [Upsdell 2005]. Judging by this dominance in the browser market, it is much more of a hassle for users to download a new browser to view a web-based software system instead of having to only install a plug-in into MS-IE.

3.3.2 Design Issues Considered

Many issues need consideration when designing a voice interaction. The three main guidelines which were followed in developing the voice interface for the VTQuest Voice system besides the eight golden rules of interface design discussed in section 2.2 are:

1) To keep the interaction going until explicitly stopped by the user
2) To allow all information on the page to be accessible
3) To only change the presentation of information, not the information presented [Kemble 2001].

Keeping the interaction going until explicitly stopped by the user is important because if it ever stops prematurely, the page will have to be reloaded manually with the keyboard or mouse, breaking the voice only interaction. The easiest way to ensure this never occurs is to ensure every user action has a corresponding reaction. If the system ever does not react to a user request, then the voice interaction ends prematurely. Another step which needs to be taken to keep a continuing voice interaction is that every link on a page should always open inside the same window. If a new window is opened, then the voice interaction continues in the background of the previous window interfering with the user’s interaction with the current window. If the voice interaction is stopped on the previous window when the new window is
opened, when the new windows is closed, the voice interaction would need to be manually restarted, breaking the voice only interaction.

In order to allow all information on the page to be accessible, a few steps must be taken. Every page should fit on the users screen. The page should never be big enough so that the user has to scroll down on his or her browser to view the whole page. If a user does have to scroll down to see the entire page, then the user who is only using his or her voice is not able to see the bottom of the page. A scroll down option could be used but it would conflict with the select box scrolling ability which is considered to be more important. Information can also be hidden from the user if the information is hidden inside of a window in the screen being displayed, such as a select box. Whenever options exist which require scrolling to see them all, the user should be able to access them in some way. The three ways which VTQuest Voice handles these situations is by allowing the user to scroll a select box either up or down by saying “Scroll Up” or “Scroll Down”, by providing a command which the user can say that reads the options in the select box to the user, or to have all the options contained in the select box written out on the page elsewhere, such as in a hint box.

To only change the presentation of information, not the information presented [Kemble 2001], the voice interface should have all if not more functionality than the non-voice interface where capable. If the user can click on options, choose items from select boxes, or type in information to an input field, a user should be able to speak the selections to choose from the select boxes or speak the information he or she wishes to enter into the input field. To create the same user experience throughout the site for voice interaction, the user should be able to interact with the same type of input fields in the same way. If the user is able to access a link by speaking its name on one page, this should hold true for other links on other pages. This action should provide the same functionality as clicking on the link. This especially holds true for the navigation of the website. VTQuest Voice has a template which is used on every page. Links exist on that template which should therefore be accessible and during every part of the interaction on every page. This concept is what led to the creation of the navigation subsystem.

### 3.3.3 Subsystems Accessible at Every Listener

Certain subsystems are available on every page of the VTQuest Voice system, namely, the navigation subsystem and the hint subsystem. These subsystems are accessible from any voice listener at any time in order to be understood and executed. The backend grammar of the subsystems uses all upper case lettering to distinguish them from other commands. Every single listener’s response is first filtered through a method to check to see if the response contains all upper case lettering. If it does, then the response is sent through the navigational and hint subsystems and interpreted. If neither of the subsystems executes with the given response or if the response does not contain all upper case lettering then the response is redirected to the method which would handle the calling listener’s response.

#### 3.3.3.1 Navigation Subsystem

The VTQuest Voice navigation subsystem utilizes the master grammar provided in Appendix A.1 which is included in every voice interaction of the system at all times. No matter what the
prompt is currently asking for, the navigation grammar is also recognized so that a user can switch to a new page whenever desired. This supports the first and seventh golden rules of user interfaces as explained in section 2.2. The navigation grammar includes access to any page in the VTQuest Voice system. This includes links to every menu option of the website’s template, giving access to every menu item and additional pages not accessible from the menu at times. An example of this occurs when a user tries to go to the Find by Name page when currently on the Get Directions page. The Find by Name page is only viewable when a user is on the Maps section of the site, but this does not limit the voice link from reaching the page even though a link to the page is not visible.

The navigation subsystem also allows for general navigation such as reloading the page or moving to the previous page, like a browser’s back functionality. This is helpful when a user makes a mistake or the system misunderstands what the user says. The user can use either the reload or back command with his or her voice and the interaction either starts again if the page is reloaded or the user is redirected to the previous page which he or she was formerly on. This supports the sixth rule of user interfaces as explained in section 2.2.

The navigation subsystem also gives the ability to stop using voice interaction which is needed during any interaction as well. If the user no longer wants to interact with voice, he or she should be able to turn off the voice interaction at any time. With this command being in the navigation subsystem, the user has this ability. This supports the fifth rule of user interfaces as explained in section 2.2.

Table 3.1 shows the complete list of navigation commands. This is comparable to a state chart since this is all the actions the navigation system performs.

<table>
<thead>
<tr>
<th>Navigation Commands</th>
<th>Redirected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home / Welcome / Maps / Main Page</td>
<td>Main page</td>
</tr>
<tr>
<td>Find by Name / Name</td>
<td>Find by Name page</td>
</tr>
<tr>
<td>Find by Abbreviation / Abbreviation</td>
<td>Find by Abbreviation page</td>
</tr>
<tr>
<td>Find by Category / Category</td>
<td>Find by Category page</td>
</tr>
<tr>
<td>Find within Distance / Within Distance</td>
<td>Find within Distance page</td>
</tr>
<tr>
<td>Get Directions / Directions</td>
<td>Directions tab</td>
</tr>
<tr>
<td>Browse Map / Browse</td>
<td>Browse page</td>
</tr>
<tr>
<td>Help</td>
<td>Help page</td>
</tr>
<tr>
<td>Overview</td>
<td>Overview page</td>
</tr>
<tr>
<td>Contact</td>
<td>Contact page</td>
</tr>
<tr>
<td>Go Back / Back</td>
<td>The previous page</td>
</tr>
<tr>
<td>Refresh / Reload</td>
<td>Refreshes the current page</td>
</tr>
<tr>
<td>Turn Off / Turn Off Voice</td>
<td>Same page with no voice interaction</td>
</tr>
</tbody>
</table>

3.3.3.2 Hint Subsystem

The hint subsystem provides context-sensitive help when the user needs assistance. This subsystem is used to assist the user when he or she does not know the proper grammar which an
input may be expecting. The solution to answer the user’s dilemma is to provide an explanation of what the system is expecting along with a few examples of how the user can structure his or her responses. On every page of the system, if a user does not understand what to do next or forgets what he or she is being asked, the user may require a hint on how to proceed. The VTQuest Voice hint subsystem enables this feature by providing either one or multiple hints for every page. If a page has no input necessary, then the hint explains that the user should choose where he or she wishes to go next. When a user requests a hint for the page, the hint is either visual or verbal depending on the amount of room on the page to display a hint. On pages such as viewing the map, no room exists to display a hint without obstructing what the page is showing, so a verbal hint is the best solution. However when room exists for a visual hint to explain how to proceed, a yellow box appears to the user containing what the system is expecting from the user such as in Figure 3.2. In some instances a user can make inputs which are not being directly asked for. When this is available, a set of example responses is given with their effects such as in Figure 3.3.

![Find Locations By Name](image)

Figure 3.2 – Examples in the hint for the Find Building by Name page

When multiple hints exist on the same page, the hints are numbered. The user can no longer only say hint, he or she must specify which hint to view such as hint one for the first hint on the
page. If the user wishes to see multiple hints, the user must ask to see each hint individually. When there are multiple hints, each hint helps with a certain part of the page. An example of multiple hints in a page referring to individual parts of the page can be seen in Figure 3.3.

Figure 3.3 – Multiple hints on Find within Distance page

Whenever a hint is requested, after the hint is displayed or spoken, the user is directed back to the previous listener. The listener’s initial timeout is also increased to be the maximum allowed time of one minute [Microsoft 2005b] to allow the user sufficient time to read and comprehend the hint and to formulate a response. This is necessary because if a user requests a hint, then he or she is having trouble understanding how to respond. Therefore the user should be allowed
more time to think about what he or she wishes to say instead of the default three seconds before the system prompts that no request is heard.

![Diagram](image)

**Figure 3.4 – Hint Subsystem state diagram which is a smaller part of larger state diagrams for the system**

### 3.3.4 Components Utilized on Multiple Pages

Certain components are used on multiple pages. Since these components look the same, they are expected to act the same as well to keep with the uniform user experience across every page of the system to be in accordance with the first rule of user interfaces as explained in section 2.2.

#### 3.3.4.1 Building and Category Data Structure

The building and category data structure holds all of the building and category information along with their relationships to one another. It has an array holding Option objects to use in select boxes of building names as the text and the building key for the value, an array holding Option objects to use in select boxes of categories names as the text and the category key for the value, a hash map with the key the building number and the value the number of floors for the building, and another hash map with the key being the category key and the value being an array of all the buildings held in Option objects. The first array of building Options is used to populate building select boxes with all of the options. The second array of category Options is used to populate category select boxes with all of the options. The first hash map is used to get the number of floors a building has once a building has been selected. The second hash map is used to get the buildings which belong to a selected category. Having each category’s building array in a hash map stored by the category key makes it much easier to dynamically update the building select box with the categories’ buildings quickly. This data structure is stored in a large string in the application’s memory so that it does not need to be regenerated for every user who uses the system. If a new building, category, or floor plan is added to the backend, the application only needs to restart itself to flush the application memory for the data structure to be rebuilt to include the new data. This string is displayed on every page which uses the data structure.
3.3.4.2 Viewing Building Names

Viewing the building names consists of the two components shown in Figure 3.5. The main component is the building select box labeled Locations which contains all of the buildings and is populated by the Building and Category Data Structure’s building array referred to in section 3.3.4.1. Since the building box does not show all the buildings to choose from all at once on the screen, a user using only voice has the ability to scroll down and up to view all the buildings in the select box. The voice commands which accomplish this task are “Scroll Up” and “Scroll Down”.

![Figure 3.5 – Choosing a building](image)

The other component which is used with the building select box is the category select box which is labeled Location Categories and located above the Locations select box. Using this category select box, a user can filter the buildings inside the building select box based on the type of category selected by the category select box. This filtering process utilizes the Building and Category Data Structure’s category hash map which returns an array of buildings used to populate the locations select box with the categories buildings. The filtering is now performed using JavaScript and no longer requires the page to be reloaded. Since the category select box only shows one item at a time and can not be scrolled easily, a user only using his or her voice can ask what the different categories to choose from are by saying “List Categories”. The voice prompt then names the categories which the user can select from for the user to hear. A user may also state the letter in which the building he or she is looking for begins with as an additional filtering technique. A user can say “A” and only the buildings starting with the letter A are displayed in the Locations select box for the user to choose from. The complete state diagram which the user goes through is in Figure 3.6. The component is exited once a building name has been selected.

The listing of the buildings on the loading of this component is analogous to the overview step in Shneiderman’s [1996] mantra. The categories box and speaking the first letter of the building name match half of the second step in the mantra of filtering. The other half of the second step in the mantra of zooming is accomplished when the user selects the building name. The last step in the mantra, details on demand, is carried out by the pages which use this component to complete the manta.
3.3.4.3 Determining Categories to View

Choosing which categories to display is comprised of the two components shown in Figure 3.7 along with the add and remove category operations depicted by the buttons. The first component is a select box which contains all of the categories to choose from and is labeled All Categories. Since the category select box contains more categories than the size of the box, it can be scrolled up and down to see all of the categories available by saying “Scroll Up” or “Scroll Down”. The second component is a select box containing all of the categories selected to view and is labeled Selected Categories. A category can only be in one of these two components at once. A user can add categories to view by saying “Add <Category Name>” or simply “<Category Name>”. A user can take away categories to view by saying “Remove <Category Name>”. The categories are added and removed one at a time. After a category is added or removed, the user is prompted to choose whether he or she wishes to add or remove any more categories. If the user says “Yes” then the ability is provided to add or remove more categories. If the user says “No” then this component exits. The complete state diagram which the user goes through is in Figure 3.8.
The maps and building floor plans returned by user’s queries to find specific locations are SVG images which are dynamic and can zoom in, out, and move around. If a user is only using voice, all of these actions can be performed as well or else the SVG images would become static, defeating one of the purposes of using SVG. The same functionality which is given to the buttons which are added to compliment the built in ASV image manipulation as described in section 3.2 is utilized when responding to a user’s voice command to manipulate the map. When a map is displayed, the user is told that it can be manipulated in whichever way he or she wants and whenever the user want to go to a new page to just say its name. The user can say any command such as “Zoom In” and the system responds in the same way as if the user hit the zoom in button. This holds true for moving the image around as well, and also for re-centering the map to the initial starting position. The list of commands available while viewing a map is shown in Table 3.2 and the complete grammar can be viewed in Appendix A.11.

<table>
<thead>
<tr>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoom In</td>
</tr>
<tr>
<td>Zoom Out</td>
</tr>
<tr>
<td>Recenter</td>
</tr>
<tr>
<td>(Move / Pan / Scroll / Go) (Up / North)</td>
</tr>
<tr>
<td>(Move / Pan / Scroll / Go) (Down / South)</td>
</tr>
<tr>
<td>(Move / Pan / Scroll / Go) (Left / West)</td>
</tr>
<tr>
<td>(Move / Pan / Scroll / Go) (Right / East)</td>
</tr>
</tbody>
</table>

Table 3.2 – Commands to control the map movements
Once the user states one of the commands from the navigation subsystem described in section 3.3.3.1, this component exits. The complete state diagram which the user goes through is in Figure 3.9.

![Figure 3.9 – Browse map state diagram which is a smaller part of larger state diagrams for the system](image)

### 3.3.5 Out of Turn Interaction

The ability to handle out of turn interaction is one of the advantages of using a voice interface. With out-of-turn interaction, the user has the ability to speak a command which is not being currently asked for, but is still understood by the system and handled appropriately.

Two examples of out-of-turn interaction are the navigation and hint subsystems as described in sections 3.3.3.1 and 3.3.3.2. These subsystems can be accessed at any voice listener. No matter what the system is expecting, the voice listener recognizes commands from either of these two grammars.

Another benefit of out-of-turn interaction which VTQuest Voice takes advantage of is combining multiple steps into one verbal sentence. This is done on both the Get Directions and the Find within Distance pages. Parsing the user’s voice command for specific types of responses, one full sentence could be used to fill in every part of the form that the user is looking to search for, avoiding having to go through the form step by step.

The out-of-turn interaction support is not complete in the VTQuest Voice system however. If it were truly complete then a user would be able to say any valid response to any question and the
valid response would be recognized, filled in, and then the same question would be asked again. This would be the most helpful in the Find within Distance page, so the user could specify a distance or categories to view, or the starting location during any listening prompt. There is a problem with this however. The category names are used in two different parts of this page. A category name can be used to filter the starting locations or it can be used to add to the list of categories to view. Due to this restriction, complete out-of-turn interaction is not easily possible and therefore not completely supported.

3.3.6 Errors

Inevitably a user will cause an error to occur. The most common error is when a user tries to say something which is not in the specified grammar being used. What the user says may be valid but not supported by the grammar. If what the user says should be valid, then an error occurring should be avoided and the grammar should be made to support the response for future interactions. If what the user says is not valid, then this should be handled properly.

3.3.6.1 Avoiding Errors

The best way to avoid a user from using words outside of the grammar is to try and allow more correct responses for the system to recognize what the user says by expanding the grammar. This can be done in a couple of different ways.

The first is to add a wide range of acceptable ways in which a user can request something such as a building name. What this means is that instead of only being able to parse a user saying the building name, the system can also parse a sentence asking to view a building. An example would be a user trying to find McBryde Hall. Instead of only accepting a user saying “McBryde Hall”, the user can say additional words before and after the building’s name such as “How do I get to McBryde Hall” or “Go to McBryde Hall please”. A complete list of variations of what can go before and after the building name with the system still able to parse through to understand which building the user is looking for is in Appendixes A.2 and A.3. To avoid errors by allowing these extraneous phrases to be added onto the beginning and end of building names, the indexOf method is used by searching for each building name as a substring inside of the response given.

Another way in which errors should be avoided is to allow multiple ways to say the same building name. An example for this situation is Ambler Johnston Hall. A user may say the correct name, “Ambler Johnston Hall” and it shall be recognized. But what if the user leaves off the Hall and just says “Ambler Johnson”? This too shall be recognized and shall be mapped to point to Ambler Johnston Hall. To ignore if the user said the building name with or without Hall at the end of it, Hall is taken away from each building name before using it as the substring to be searched for. Another case would be if a user referred to Ambler Johnston Hall by its nick name, AJ. If the user says “AJ”, it shall also be recognized and mapped to point to Ambler Johnston Hall. Another case would be if more than one acceptable pronunciation exists for a building name, or if the pronunciation used is different than the one the system is expecting. Every pronunciation of the same building name should still point to the same building. In order to allow for nick names and other pronunciations, every instance of a building’s nick name is
searched as the substring and replaced with the correct building name which it represents. The
nick names and their corresponding building names are held in a hash map with the nick name as
the key. This makes it easy to access and iterate through.

3.3.6.2 Handling Errors

When an error occurs upon the user saying something that is not in the grammar, then the system
should handle this gracefully. This can occur when the system is listening to a user even if the
user is not directing his or her conversation toward the system. This can occur when the user is
in the middle of the interaction and receives a phone call or when the user is talking to someone
else in the room. The system should explain what went wrong and ask the user to try again
thereby notifying the user what he or she asked for must not be correct or can not be interpreted.
Each voice listener has a corresponding on failure prompt so that the error prompt can specify
exactly what is being looked for. If every voice listener used the same error prompt then it
would have to reference a generic command and not explain exactly what the listener is looking
for. The voice interaction should never be stopped due to an error.

3.3.7 Pages with Only Links

Four pages exist in the system which only have links on them with no form input and are static.
These pages are the Welcome page, the Help page, the Overview page, and the Contact page.
They are static in that no client scripting exists on these pages besides allowing voice interaction.
No matter what the user does, the content on the page does not change. All four of these pages
share the same state diagram as shown in Figure 3.10. The only prompt which the user is given
is to state where he or she wants to go next, but given the maximum time available to view the
page before the listener times out. The maximum time is one minute [Microsoft 2005b].
3.3.8 Find by Building Name

The find by building name page starts by prompting the user to state a building name and uses the Building Names component whose capabilities are described in section 3.3.4.2. Once the Building Names component exits, a building is selected. The user is prompted to confirm that the building selected is the correct one. This step helps to prevent errors. If the wrong building name is selected then the selecting the building name step starts over. If the selected building has no floor plans to choose from, the user is sent directly to the results page to show the building on the map. However, if floor plans exist for the building selected, a floors box becomes visible. This floors box contains the final step in the Shneiderman [1996] mantra of details on demand. The user may then select to get the building to be displayed on the map or to choose one of the floor plans to view. The user may say either the ordinal number of the floor such as “First” or the floor number such as “One”. Every combination of either the rank or floor number is recognized and interpreted correctly. If the user selects to just get the building to be displayed on the map, the map is generated with that building centered on the map at zoom level two as described in section 3.2. If the user selects a floor plan, the results page is directed to show that floor plan starting from a view which shows most of the floor at the same time. The maximum number of floors a building has at Virginia Tech is Slusher Hall with 12 floors. Since this is the maximum number of floors, the floors selects box can contain up to 13 elements without scrolling, an element for every one of the 12 floors maximum and one more for the
Show Location on Map spot at the top. Since this select box can display the maximum number of floors, it eliminates the user’s need to scroll to see all of the options available. A state diagram of what occurs is displayed in Figure 3.11.

Figure 3.11 – Find by building name state diagram

### 3.3.9 Find by Abbreviation

The find by abbreviation page starts by prompting the user to say the abbreviation. The listener accepts any 2 to 5 letter response which gives it much more flexibility, but also widens the range of acceptable error prone responses. If the user says an incorrect abbreviation, the system repeats back what it thought it heard so that if it misunderstood the user, the user can adjust and try the abbreviation again. Like the buildings select box, the abbreviation select box contains
more building abbreviations then the user can view at one time without scrolling down. Therefore the ability for the user to say “Scroll Down” or “Scroll Up” is integrated into the response so that the user can scroll through the abbreviations. Once the user selects the abbreviation, the map is generated with that building which the abbreviation belongs to centered on the map at zoom level two as described in section 3.2. A state diagram of what occurs is displayed in Figure 3.12.

![Figure 3.12 – Find by abbreviation state diagram](image)

### 3.3.10 Find by Category

The find by category page starts out by asking the user which categories he or she wishes to add. The page then enters into the determining category component described in section 3.3.4.3. When the component exits, the selected buildings are highlighted and sent to be viewed in the map. Since each category has at least one building, all of the buildings should be able to be viewed all at once to start off and then allow the user to zoom in on the particular buildings of interest. The process that accomplishes this is described in section 3.3.4.4. A state diagram of what occurs is displayed in Figure 3.13.
3.3.11 Find Category within Distance around a Building

The find category within distance around a building page, or commonly known as the Find within Distance page, selects a location to be the center of the search, allows the user to select categories to search for, and then allows the user to choose the search radius around the center building to search. The page starts by prompting the user to select the starting location to be the center of the search. The Building Names component is used to select the location so the user may use all of the component’s attributes and features when selecting the starting location. After the user selects the location, the determining category component is entered which is described in section 3.3.4.3. Once the categories are selected, the user is prompted to select the distance. Once the distance is selected, the request is processed and the user is redirected to the map of his or her results as described in section 3.3.4.4.

However, the user can eliminate this step by step process. At the first prompt where the user is asked to select the starting location, he or she may instead act out-of-turn and give the whole query at once. An example of this would be "Show me all <category name(s)> buildings within <distance> yards of <starting location>". The system then parses the user’s response and fills in the form items and the user is immediately redirected to the targeted map as described in section 3.3.4.4. If using this option, the user should plan what he or she is going to request before hand, because the prompt does not wait very long for the user to make up his or her mind in the middle of the response. A state diagram of what can occur is displayed in Figure 3.14.
3.3.12 Get Directions

The get directions page allows the user to choose a starting location and an ending location to find directions between. The page starts by prompting the user to select the location where the user wants to start from. The Building Names component is used to select the location so the user may use all of the component’s attributes and features when selecting the starting location.
After the location is selected, the user is prompted to select the finishing location. Again, the Building Names component is used to select the location. After the ending location is selected, the user is redirected to the targeted map as described in section 3.3.4.4.

However, the user can eliminate this step by step process. At either of the prompts where the user is asked to select a location, he or she may instead act out-of-turn and give the whole query at once. An example of this would be “How do I get from <Starting Location> to <Ending Location>”. The system then parses the user’s response and fills in the form items and the user is immediately redirected to the map as described in section 3.3.4.4. A user may also use the form “How do I get to <Ending Location> from <Starting Location>”. Either of these responses works. The system looks at the words separating the two building names to determine if the first building name is the starting or the ending location. A state diagram of what can occur is displayed in Figure 3.15.

![Figure 3.15 – Get directions state diagram](image)
3.4 Design Issues for the Voice Interface

We encountered several issues while developing the design and implementation of VTQuest Voice. We addressed the issues in a way to provide benefit to the most users.

The first issue is the ability to ignore any noise not made directly by the user. This became important when the system tried to handle interrupt responses, commonly referred to as barge-in. In order to allow barge-ins the system must be listening at the same time it is speaking. If the user is not using a headset to listen to the system, the microphone picks up the system and interprets what the system is saying to the user as if it is a response from the user. In order to not require all users to wear headsets when using the system, barge-ins are not allowed.

The second issue is the ability to handle complete out-of-turn interaction. The system is able to accept sentences and parse those sentences to skip over having to go through the step by step interaction. However the system can not handle a user submitting a response for a page other than the one being used. The system also does not support choosing which of the steps to complete first in a multi-step page. It only allows the user to complete the steps or skip them by using one full sentence.

The final issue which can not be directly solved by the system being created is the voice recognition accuracy of the ASR. The user is recommended and should train his or her voice, but even after training, the user response accuracy may still not be 100%. Without 100% accuracy, this problem persists through any voice application using the freely available current ASR technology.
Chapter 4: Verification and Validation

The VTQuest Voice web-based software system is verified and validated to achieve a satisfactory level of confidence. Testing was accomplished throughout the life cycle of the development and a variety of verification and validation (V&V) techniques were used throughout the process. Whenever errors were found through the V&V process, a V&V technique was used to correct the errors and to make sure the correction did not cause any other errors.

4.1 Verification

VTQuest Voice verification was conducted to ensure the transformational accuracy. The V&V techniques, Desk Checking, Documentation Checking, Inspections, and Structured Walkthroughs were employed for assessing accuracy.

4.1.1 Desk Checking

Desk checking, also known as self inspection, involves inspecting the code without executing it. Some typical desk checking activities include syntax review, cross-reference examination, convention violation assessment, detailed comparison to specifications, code reading, control flowgraph analysis, and path sensitizing [Beizer 1990]. These methods were used to desk check the VTQuest Voice system.

4.1.2 Documentation Checking

Documentation checking deals with examining the documentation for accuracy assessment. For the VTQuest Voice system the documentation includes user manuals, help files, and code documentation. The manuals are used by administrators of the system to add new buildings and paths to the system and were read through and tested to make sure the manual matched what is executed. The manuals were used for the original VTQuest system but they are updated with what changes need to be made to stay consistent with the voice interface. The help page was redesigned to be geared toward helping a user using the voice interface to answer the most common questions run into by users. The code documentation was read through to make sure that it was consistent with the code and that enough internal and method commenting was given to ensure that the code was understandable.

4.1.3 Structured Walkthroughs

Structured walkthroughs encourages discussion leading to questions on how pieces of the software work. This process helps discover error in the program and differences between the documentation and what the actual system. These walkthroughs typically involve a presenter, a coordinator, a scribe, a maintenance oracle, a standards bearer, a user representative, and other reviewers [Balci 2003]. For the structured walkthroughs for the VTQuest Voice system, all of these roles were performed by two people. The developer took the role of the presenter, coordinator, and scribe and the second team member took the rest of the roles. Having only two
members participate in a walkthrough is not the recommended but it worked for the purposes of walking through the VTQuest Voice system.

4.1.4 Inspections

Inspections are a more formal process than a walk through with specific steps laid out and a checklist of functionality and features to look for and make sure work correctly. The inspection of the VTQuest system mainly revolved around making sure the user interface components contained the correct data from the database, that each component worked properly, and that every interaction which was done by the voice interface had the same result when done with a mouse or keyboard on the GUI.

4.2 Validation

VTQuest Voice validation dealt with the assessment of behavioral and representational accuracy. The following V&V techniques were used: state transition analysis, syntax analysis, alpha/beta testing, comparison testing, debugging, functional black-box testing, interface testing for data, software, and user interfaces, regression testing, and white-box condition testing.

4.2.1 State Transition Analysis

To perform state transition analysis a finite number of states that the software execution goes through are identified and shown how they relate to one another [Balci 2003]. When analyzing the state transition, we ensured that all states are reachable, that all states are defined, that every state has an exit point, and that every state responds properly to all possible conditions. The state diagrams which make up the VTQuest Voice system are shown in Figure 3.4, Figure 3.6, Figure 3.8, Figure 3.9, Figure 3.10, Figure 3.11, Figure 3.13, Figure 3.14, Figure 3.15, and also in Table 3.1. Each state is shown to have an exit point and entrance point so that no state will be the final condition. VTQuest Voice was also tested to make sure that each state responded properly to all possible conditions.

4.2.2 Syntax Analysis

Syntax analysis is performed by the programming language compiler to make sure that the mechanics of the language are applied correctly [Balci 2003]. In the case for VTQuest Voice, this is performed by the java compiler. This is especially important with the JSP pages since if any syntax error exists, it will only be discovered after a user tried to view the page and instead of the page being displayed a compilation error is displayed in its place.

4.2.3 Alpha/Beta Testing

Alpha and beta testing is used to test completed software in a real world environment to gather feedback from real users. The VTQuest Voice system used alpha and beta testing whenever a stable release of the software was available. This helped to find any errors which were missed by the other testing procedures used.
4.2.4 Comparison Testing

Comparison testing is when more than one software product with the same intended use is available for testing [Balci 2003]. This is the case for the VTQuest Voice system because it needed to keep at least the same functionality of the original VTQuest system. When both of these systems were tested using the input data and test cases, they both generated the same behavior and had the same functionality. The VTQuest Voice system was tested using both the voice only interface and the mouse or keyboard GUI. A test case was used for each page to show that no page had changed functionality. For the Find by Name page, the test case was to filter the buildings to be academic only, to choose McBryde Hall and then to show the building on the campus map. For the Find by Category page, the test case was to add Academic, Administrative, and Dining Halls, remove Administrative, then submit to view all the academic and dining hall buildings. For the Find by Abbreviation page, testing that AJ was mapped to Ambler Johnston Hall and that it was found on the map. For the Find within Distance page, the buildings were filtered by Administrative, then Burruss Hall was selected, then Academic and Dining Halls were added as categories, then Academic was removed, then 500 yards for the distance was chosen and then the campus map was displayed with all Dining Halls within 500 yards of Burruss Hall. For the Get Directions page, Davidson Hall was chosen as the starting location and War Memorial Gym was chosen as the ending location and the maps was received with the path between the two buildings highlighted. On each map page displayed it was zoomed in once and moved up to show that the map manipulations still worked.

4.2.5 Debugging

Debugging is a two step iterative process. In the first step, testing was performed to reveal the presence of any bugs. Whenever a bug was found, the second step was performed to find the root cause of the problem and to make changes to the software to remove the bug. After the bug was fixed, the iterative process was repeated until no more bugs could be found.

4.2.6 Functional Black-box Testing

Black-box testing is performed by inputting data into the software and verifying that the correct output is produced. It is called black-box testing because the software is treated as a “black-box”; the only thing that is being used is the input into the box and the output out of the box. What is done inside of the box and how it does what it does is completely ignored. This was performed on the VTQuest Voice system with a wide variety of input data using the voice and the normal interfaces to input both valid and invalid data. The valid data was tested to make sure that the system produced the corresponding response to that data. The invalid data which was tested was with the voice interface to ensure that if an input was given that was outside of the acceptable grammar, the system would inform the user that the input could not be recognized and to refer to the hint on the page for any further information. Both the valid and invalid inputs generated the correct output.
4.2.7 Data Interface Testing

Data interface testing assesses the accuracy of all the data inputted into the software or outputted from the software during execution to validate it is all correct [Balci 2003]. This approach was used with the data inputted from the database to the VTQuest Voice system. In particular to validate all of the buildings appeared in the buildings select box and all the categories in the category select box.

4.2.8 Software Interface Testing

Software interface testing detects errors which occur as a result of the transition between different modules, or pages when it comes to web-based software systems. This testing was used to ensure that transitioning between different pages worked correctly especially when transferring from a query page to the map page. This testing was also used to make sure that a user could go straight to any page such as the results page for a query before submitting a query page. This caused an error in the Get Directions page for the original VTQuest system but was fixed in the VTQuest Voice system.

4.2.9 User Interface Testing

User interface testing deals with the evaluation of the interactions between the user and the software. This was used for the VTQuest Voice system to ensure that the voice user interface provided to the user was useful and that the interaction was smooth and seamless. A problem which this testing uncovered was that users would try and response to a system prompt before it was finished. This led to the beginning of the user’s response was not heard since he or she began talking before the system began listening. This issue is discussed in the Voice Help page for the system to explain to users what is happening.

4.2.10 Regression Testing

Whenever a bug is found and fixed, regression testing is used to make sure that by fixing the bug it did not create other bugs. This is done by retesting the system after every bug fix to ensure that no new errors have occurred due to that bug fix. Regression testing was used for VTQuest Voice whenever a bug was fixed to make sure that no new bugs were created. Every part of the page which the bug being fixed was apart of was retested to make sure that no new bugs occurred.

4.2.11 White-box Condition Testing

Whereas black-box testing completely ignored the internal working of the software, white-box testing ensures that the internals of the system work correctly. With condition testing, the software is executed using test data to execute as many logical conditions as possible [Balci 2003]. For the VTQuest Voice system, this was particularly important to make sure that each building and category name was a unique substring in any possible sentence utterance allowed by the grammar. This was done by creating a test case which compared every possible name to the entire grammar to make sure that its name was not a substring inside of another name. Since
the word hall was removed from every building name when comparing what the user said, it was also taken away in the grammar. This testing found that three buildings conflicted with three other buildings. Lane Hall conflicted with Lane Stadium, Williams Hall conflicted with Major Williams Hall, and War Memorial Gym’s nickname of War conflicted with War Memorial Chapel. These three conditions are taken into account whenever the user’s input is searched through for the building name. Whenever a new building name or nickname is added to the system, this test can be rerun to make sure that no new conflicts occur in accordance with the regression testing.
Chapter 5: Conclusions and Future Research

5.1 Conclusions

Adding the voice interface to VTQuest has given much more flexibility to the way users can access and interact with the system. It has given the user the ability to use the system in the manner which best suits their needs, expanding the possibilities of situations in which the mapping web-based software system can be used. When using the voice interface, a user’s hands and eyes are free to perform other tasks. The only time a user would need to look at the system with using the voice interface is when viewing results of a query being displayed on the campus map. Provided with the necessary hardware and browser, this system can be used in multiple different activities such as on a tour, while driving around campus, while eating at your computer, or for a student searching for his or her next class. When using the system while driving, the voice interface provides a safe way to access the system by allowing the driver to keep his or her hands on the steering wheel and only glance at the system when needing to situate where he or she is in the system or when viewing the campus map.

ASR technology needs further development before the usability of the software can be maximized. The user having to wait until the prompt is completed is a very restrictive aspect, however having the ASR and TTS running at the same time using SALT would have also caused its own limitations [Stifelman 1993]. A push to talk system was recommended by Stifelman [1993] but this would have broken the voice only interaction and required the use of the user’s hands. Until the ASR technology has been improved, a workaround will continue to need to be used.

With the addition of the voice interface, a foundation for future additions onto the system has been set up to be easily expandable with the help of the user defined event structure of SALT. This will come in helpful if any additional modalities are to be added to the system.

5.2 Contributions

This thesis reveals several contributions which the VTQuest Voice system has made to extend the original VTQuest web-based software system. These contributions include:

- Demonstrating the feasibility of a web-based software system utilizing voice communication by creating VTQuest Voice.
- Creation of a voice interface which can be used exclusively to interact with the system.
- Creation of flexible grammar which is used by the voice interface and can easily be extended when additional information is added to the system.
- Added functionality to allow for users to identify buildings by their nicknames.
- Providing full sentence input to allow for skipping step based inputs.
- Creation of a visible navigation system for navigating the campus map and the building floor plans.
• Added functionality to the map results to center on the buildings returned by the search.
• Verifying and validating the VTQuest Voice system to achieve a satisfactory level of confidence.
• Sped up the load time of each query page which uses the building names component and the determining categories component.
• Sped up the filtering time of the building names component.
• Created the additional filtering option in the building name component to filter buildings by the letter which they begin with.
• Improved the hint system usefulness and added the ability to work with more browsers.
• Fixed various bugs such as if a user tries to access the results map page before submitting a query to the system.

5.3 Future Research

VTQuest Voice can be improved through additional capabilities to the voice system as well as improvements in new areas which can be explored by using this technology as a basis.

An improvement could be made in the area of error handling for the voice interface. When the user says something which the system can not understand, the system should try and guess what the user said. This could be very helpful if the user is using a different pronunciation of the word than the system is looking for. This could be similar to a spell checker but for voice input.

The ability to ignore any noise not made by the user would also be extremely useful for the voice interface. An example of when the user’s environment may be impacted by background noise is when the user is traveling in a car. If background noise could be ignored, the barge-in event which had to be unimplemented due to the decision to include a wider user basis by not requiring a headset could be implemented.

Expanding the devices which the web application would work on would also be very beneficial. Including the use for PDAs, smart phones, or built into a car navigation system would allow for mobile access and could allow the user to view the maps in the middle of an outing in the Virginia Tech campus instead of just being able to plan his or her trip.

Expanding the modality of the application could also be useful. One area which seems to be the next logical step would be through the use of a touch screen such as with PDAs. This would most likely mean including Ink Markup Language (InkML) to the already growing sets of technologies used in the application.

A thorough usability evaluation for the VTQuest Voice system would be helpful to determine user’s acceptance for both the visual interface, the voice interface, and how these two interact. Based on these results, determine whether the current form of voice interaction is sufficient or whether a more unified approach should be used such as using incremental and expanding prompts, tapering, or hints as described by Yankelovich [1996].
Appendix A: VTQuest Voice Grammar

A.1 Master Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="masterGram"
xml:lang="en-US">
  <rule id="masterGram" scope="public">
    <item>
      <one-of>
        <item>GO TO</item>
      </one-of>
      <one-of>
        <item>
          <one-of>
            <item>WELCOME</item>
            <item>MAIN</item>
            <item>HOME</item>
          </one-of>
          <one-of>
            <item>SCREEN</item>
            <item>PAGE</item>
          </one-of>
        </item>
      </one-of>
      <item>HOME</item>
      <item>WELCOME</item>
      <item>BROWSE</item>
      <item>BROSWE MAP</item>
      <item></item>
      <one-of>
        <item>FIND</item>
        <item>FIND BY</item>
        <item>FIND BUILDING BY</item>
        <item>VIEW</item>
      </one-of>
      <one-of>
        <item>NAME</item>
        <item>BUILDING</item>
        <item>ABBREVIATION</item>
        <item>BUILDING ABBREVIATION</item>
        <item>CATEGORY</item>
        <item>CATEGORIES</item>
        <item>WITHIN DISTANCE</item>
      </one-of>
    </item>
  </rule>
</grammar>
A.2 Beginning Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="beginning"
xml:lang="en-US">
  <rule id="beginning" scope="public">
    <one-of>
      <item>Please</item>
    </one-of>
    <one-of>
      <item>Can you</item>
    </one-of>
    <one-of>
      <item>How do I get</item>
      <item>Go</item>
      <item>Where is</item>
      <item>Where can I find</item>
      <item>Locate</item>
      <item>Find</item>
      <item>Get</item>
      <item>Take me</item>
      <item>Show me</item>
    </one-of>
    <one-of>
      <item>to</item>
      <item>from</item>
    </one-of>
    <one-of>
      <item>Me</item>
    </one-of>
  </rule>
</grammar>
<item>the</item>
<item />
</one-of>
</rule>
</grammar>

A.3 Ending Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="ending"
xml:lang="en-US">
<rule id="ending" scope="public">
<one-of>
  <item>be found</item>
  <item>located</item>
  <item>please</item>
</one-of>
</rule>
</grammar>

A.4 Category Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="category"
xml:lang="en-US">
<rule id="category" scope="public">
<one-of>
  <item><ruleref uri="master_gram.grxml#masterGram"/></item>
  <item><ruleref uri="#letters"/></item>
  <item>Scroll Down</item>
  <item>Scroll Up</item>
  <item>Show More</item>
  <one-of>
    <item>Locations</item>
    <item>Buildings</item>
  </one-of>
</item>
<item>Add</item>
<item>Remove</item>
<rule id="categories" scope="public">
    <one-of>
        <item>List Categories</item>
        <item>What are the Categories</item>
        <item>All Categories</item>
        <item>Academic</item>
        <item>Administrative</item>
        <item>Corporate Research Center</item>
        <item>Dining Hall</item>
        <item>Dormitory</item>
        <item>Library</item>
        <item>Recreational Facility</item>
        <item>Sports Facility</item>
    </one-of>
</rule>

<rule id="letters" scope="public">
    <one-of>
        <item>A</item>
        <item>B</item>
        <item>C</item>
        <item>D</item>
        <item>E</item>
        <item>F</item>
        <item>G</item>
        <item>H</item>
        <item>I</item>
        <item>J</item>
        <item>K</item>
        <item>L</item>
        <item>M</item>
        <item>N</item>
        <item>O</item>
        <item>P</item>
        <item>Q</item>
        <item>R</item>
        <item>S</item>
        <item>T</item>
        <item>U</item>
    </one-of>
</rule>
A.5 Building Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="findBuilding"
xml:lang="en-US">
  <rule id="findBuilding" scope="public">
    <one-of>
      <item><ruleref uri="cat_gram.grxml#category"/></item>
      <item><ruleref uri="#innerFindBuilding"/></item>
    </one-of>
  </rule>
  <rule id="innerFindBuilding" scope="public">
    <ruleref uri="beginning.grxml#beginning"/>
    <one-of>
      <item><ruleref uri="#building"/></item>
      <item><ruleref uri="#nickname"/></item>
    </one-of>
  </rule>
</grammar>
<item>Air Conditioning Facility</item>
<item>Ambler Johnston</item>
<item>Andrews</item>
<item>Architecture Annex</item>
<item>Art and Design Learning Center</item>
<item>Barringer</item>
<item>Brodie</item>
<item>Burke Johnston Student Center</item>
<item>Burruss</item>
<item>Campbell</item>
<item>Cassell Coliseum</item>
<item>Cheatham</item>
<item>Cochrane</item>
<item>Cowgill</item>
<item>Cranwell International Center</item>
<item>Dairy Science Complex</item>
<item>Davidson</item>
<item>Derring</item>
<item>Dietrick</item>
<item>Donaldson Brown Hotel and Conference Center</item>
<item>Durham</item>
<item>Eggleston</item>
<item>Engel</item>
<item>Femoyer</item>
<item>Founders</item>
<item>Fraclin Biotechnology Center</item>
<item>Garvin</item>
<item>Grounds Building</item>
<item>Hahn</item>
<item>Hancock</item>
<item>Harper</item>
<item>Health and Sately Building</item>
<item>Henderson</item>
<item>Hillcrest</item>
<item>Holden</item>
<item>Hutcheson</item>
<item>Information Center</item>
<item>Visitor Center</item>
<item>Johnson</item>
<item>Lane</item>
<item>Lane Stadium</item>
<item>Worsham Field</item>
<item>Lee</item>
<item>Litton Reaves</item>
<item>Major Williams</item>
<item>McBryde</item>
<item>McComas</item>
<item>Media Annex</item>
<item>Media Building</item>
<item>Merryman Athletic Facility</item>
<item>Miles</item>
<item>Military Building</item>
<item>Monteith</item>
<item>Moss</item>
<item>Motor Pool</item>
<item>NWS Balloon Launching Facility</item>
<item>National Weather Service</item>
<item>New Residence Hall East</item>
<item>Newman</item>
<item>Newman Library</item>
<item>Norris</item>
<item>O Shaughnessy</item>
<item>Owens</item>
<item>Pamplin</item>
<item>Parking Services Building</item>
<item>Patton</item>
<item>Payne</item>
<item>Peddrew Yates</item>
<item>Power Plant</item>
<item>Price</item>
<item>Pritchard</item>
<item>RB14</item>
<item>RB15/16</item>
<item>Randolph</item>
<item>Rasche</item>
<item>Research II</item>
<item>Research VII</item>
<item>Research X</item>
<item>Research XII</item>
<item>Robeson</item>
<item>Sandy</item>
<item>Saunders</item>
<item>Seitz</item>
<item>Shanks</item>
<item>Shultz</item>
<item>Slusher</item>
<item>Smyth</item>
<item>Solitude</item>
<item>Southgate Center</item>
<item>Squire Student Center</item>
<item>Sterrett Facilities Complex</item>
<item>Student Services Building</item>
<item>Thomas</item>
<item>Torgersen</item>
<item>University Bookstore</item>
<item>VIA Medical School</item>
<item>VTLS</item>
<item>Vawter</item>
<item>WPI</item>
<item>Wallace</item>
<item>War Memorial Chapel</item>
<item>War Memorial Gym</item>
<item>Whittemore</item>
<item>Williams</item>
</one-of>
<one-of>
<item>Hall</item>
</one-of>
</rule>

<rule id="nickname" scope="public">
<one-of>
<item>War</item>
<item>Chapel</item>
<item>Church</item>
<item>VIAM</item>
<item>Medical School</item>
<item>Med School</item>
<item>Bookstore</item>
<item>Bridge</item>
<item>Torgersen Bridge</item>
<item>Student Services</item>
<item>Squires Student Center</item>
<item>Squires</item>
<item>Squires Food Court</item>
<item>Slusher Tower</item>
<item>Shultz Express</item>
<item>Shultz Dining Center</item>
<item>Peddrew</item>
<item>Yates</item>
<item>Parking Services</item>
<item>Hokie Grill</item>
<item>Owens Eatery</item>
<item>O Shag</item>
<item>Library</item>
<item>New Res</item>
<item>New Residence East</item>
<item>New Res East</item>
<item>NWS</item>
<item>Merryman</item>
</one-of>
</rule>
<item>McComas Gym</item>
<item>Football Stadium</item>
<item>Fralin</item>
<item>East Eggleston</item>
<item>West Eggleston</item>
<item>East Egg</item>
<item>West Egg</item>
<item>Donaldson Brown Hotel</item>
<item>Hotel</item>
<item>D 2</item>
<item>Dietrick Express</item>
<item>Deets</item>
<item>Deets Place</item>
<item>General Store</item>
<item>Cranwell</item>
<item>West End</item>
<item>West End Market</item>
<item>AJ</item>
<item>Cassell</item>
<item>GBJ</item>
<item>Johnston Student Center</item>
A.7 Floor Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmns="http://www.w3.org/2001/06/grammar" root="floor"
xml:lang="en-US">
  <rule id="floor" scope="public">
    <one-of>
      <item><ruleref uri="master_gram.grxml#masterGram"/></item>
      <item>
        <one-of>
          <item>First</item>
          <item>Second</item>
          <item>Third</item>
          <item>Fourth</item>
          <item>Fifth</item>
          <item>Sixth</item>
          <item>Seventh</item>
          <item>Eighth</item>
          <item>Ninth</item>
          <item>Tenth</item>
          <item>Eleventh</item>
          <item>Twelveth</item>
          <item>Thirteenth</item>
        </one-of>
        <one-of>
          <item>Floor</item>
        </one-of>
      </item>
      <item>1</item>
      <item>2</item>
      <item>3</item>
      <item>4</item>
      <item>5</item>
      <item>6</item>
      <item>7</item>
      <item>8</item>
      <item>9</item>
      <item>10</item>
      <item>11</item>
      <item>12</item>
      <item>13</item>
      <item>Get Building</item>
      <item>Display Location</item>
      <item>Display Location On Map</item>
  </one-of>
</grammar>
A.8 Abbreviation Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="findAbb"
xml:lang="en-US">
<rule id="findAbb" scope="public">
    <one-of>
        <item><ruleref uri="master_gram.grxml#masterGram"/></item>
        <one-of>
            <item><ruleref uri="cat_gram.grxml#letters"/></item>
            <item repeat="2-5">
                <ruleref uri="cat_gram.grxml#letters"/>
            </item>
        </one-of>
    </one-of>
</rule>

<rule id="abbreviation" scope="public">
    <one-of>
        <item repeat="0-1">
            <ruleref uri="cat_gram.grxml#letters"/>
        </item>
        <item>AA</item>
        <item>AGNEW</item>
        <item>AIRCN</item>
        <item>AJ</item>
        <item>ALUM</item>
        <item>ART C</item>
        <item>ARW</item>
        <item>BAR</item>
        <item>BOOK</item>
        <item>BRO</item>
        <item>BUR</item>
        <item>CAM</item>
        <item>CHAP</item>
        <item>CHRNE</item>
        <item>CIC</item>
        <item>CO</item>
        <item>COL</item>
    </one-of>
</rule>
<item>DAV</item>
<item>DER</item>
<item>DJPAV</item>
<item>DTRIK</item>
<item>DURHM</item>
<item>EGG</item>
<item>ENGEL</item>
<item>FEM</item>
<item>FNDR</item>
<item>FRALN</item>
<item>GARV</item>
<item>GBJ</item>
<item>GRNDS</item>
<item>GYM</item>
<item>HAHN</item>
<item>HAN</item>
<item>HARP</item>
<item>HILL</item>
<item>HOLD</item>
<item>HOSP</item>
<item>HSBLD</item>
<item>HUTCH</item>
<item>JCH</item>
<item>JOHN</item>
<item>LANE</item>
<item>LEE</item>
<item>LIBR</item>
<item>LITRV</item>
<item>MAJWM</item>
<item>MCB</item>
<item>MCCOM</item>
<item>MEDIA</item>
<item>MIL</item>
<item>MILES</item>
<item>MON</item>
<item>MOSS</item>
<item>MPOOL</item>
<item>MRYMN</item>
<item>NEW</item>
<item>NOR</item>
<item>NRHE</item>
<item>NWS</item>
<item>NWSB</item>
<item>OSHA</item>
<item>OWENS</item>
<item>PAM</item>
<item>PAT</item>
<item>PAYNE</item>  
<item>PKN</item>  
<item>PRICE</item>  
<item>PRIT</item>  
<item>PY</item>  
<item>RAND</item>  
<item>RASCH</item>  
<item>RB</item>  
<item>RB14</item>  
<item>RE12</item>  
<item>RE2</item>  
<item>RE7</item>  
<item>REAX</item>  
<item>ROB</item>  
<item>SANDY</item>  
<item>SAUND</item>  
<item>SEITZ</item>  
<item>SGCTR</item>  
<item>SHANK</item>  
<item>SHULT</item>  
<item>SLUSH</item>  
<item>SMYTH</item>  
<item>SOL</item>  
<item>SQUIR</item>  
<item>SSB</item>  
<item>STAD</item>  
<item>STCTR</item>  
<item>THOM</item>  
<item>TORG</item>  
<item>TVFLM</item>  
<item>VAW</item>  
<item>VIAM</item>  
<item>VISIT</item>  
<item>VTLS</item>  
<item>WAL</item>  
<item>WHIT</item>  
<item>WMS</item>  
<item>WPI</item>  
</one-of>  
</rule>  
</grammar>
A.9 Within Distance Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="withinDistance"
xml:lang="en-US">
  <rule id="withinDistance" scope="public">
    <one-of>
      <item>
        <one-of>
          <item>Show</item>
          <item>Show Me</item>
        </one-of>
      </item>
      <one-of>
        <item>All</item>
      </one-of>
      <item repeat="1-25">
        <one-of>
          <item><ruleref uri="cat_gram.grxml#categories"/></item>
          <item>and</item>
          <item>buildings</item>
          <item>all</item>
          <item>the</item>
        </one-of>
      </item>
      <item><ruleref uri="building_gram.grxml#findBuilding"/></item>
    </one-of>
  </rule>
</grammar>

A.10 Distance Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="distance"
xml:lang="en-US">
<rule id="distance" scope="public">
  <one-of>
    <item><ruleref uri="master_gram.grxml#masterGram"/></item>
    <one-of>
      <item>50</item>
      <item>100</item>
      <item>150</item>
      <item>200</item>
      <item>250</item>
      <item>500</item>
      <item>1000</item>
    </one-of>
    <one-of>
      <item>Yards</item>
    </one-of>
  </one-of>
</rule>

A.11 Alignment Grammar

<grammar version="1.0" tag-format="semantics-ms/1.0"
xmlns="http://www.w3.org/2001/06/grammar" root="alignment"
xml:lang="en-US">
  <rule id="alignment" scope="public">
    <one-of>
      <item><ruleref uri="master_gram.grxml#masterGram"/></item>
      <item>
        <one-of>
          <item>pan</item>
          <item>move</item>
          <item>scroll</item>
          <item>go</item>
          <item>up</item>
          <item>down</item>
          <item>left</item>
          <item>right</item>
          <item>north</item>
          <item>south</item>
          <item>east</item>
        </one-of>
      </item>
    </one-of>
  </rule>
</grammar>
<item>west</item>
<item>closer</item>
<item>further away</item>
<item>away</item>
</one-of>
</item>
<item>
<item>zoom</item>
<one-of>
<item>in</item>
<item>out</item>
</one-of>
</item>
<item>center</item>
<item>recenter</item>
</one-of>
</rule>
</grammar>
Bibliography


VITA

Thomas W. Schneider

Current Address (August 2004-2005):
902 #10 Brightwood Manor Drive
Blacksburg, VA 24060
Cell: (540) 818-6565
E-Mail: tschneid@vt.edu

Permanent Address:
11507 Summer Oak Drive
Germantown, MD 20874-1931
Home: (301) 972-3121

Education
M.S., Computer Science, Graduated May 2005
B.S., Computer Science; Minor: Mathematics, Graduated May 2004, magna cum laude
Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, Virginia
Grad GPA: 3.47/4.00 Undergrad GPA: 3.67/4.00

Computer Skills
Programming Languages:
Java        C/C++   JSP/Servlets
Java Spaces  JSF      UML
JDBC        JUnit    JavaScript/HTML
SQL         PHP      ASP

Software:
WebSphere Application Server (WAS)      Eclipse
WAS Developer (WSAD)                    Visual Studio
Weblogic Server                         Visio
Rational Rose                           Photoshop

Experience
International Business Machines (IBM): Summer Internship – Summer 2004 – Research Triangle Park, NC
• Created documentation for the Service Data Object (SDO) JDBC Data Mediator Service (DMS) which will be part of the Infocenter for WebSphere Application Server (WAS) 6.0 available online and shipped with the product.
• Wrote test cases using Eclipse and JUnit for the JDBC DMS to strengthen the test suite.
• Created a web application using JSF to further test the JDBC DMS in a complete application environment.
• Member of the Future Blue council and coordinated a movie night every Sunday evening for the co-ops.

Virginia Tech Computer Science Department: Teaching Assistant – Fall 2003 to Spring 2004 – Blacksburg, VA
• Taught four lab sections of Intro to Object-Oriented Development using Java to students in groups of two or three, assisting with any problems they encountered with the lab or the automatic grader (Web-CAT).
• Graded the students’ projects and labs by reviewing their code and adding remarks after the Web-CAT process.
• Held office hours every week to help students with any problems they were having with the course.
• As an undergrad, assisted in four labs with a GTA by grading homework and helping students during lab.

National Association of Securities Dealers (NASD): Technology Internship – Summer 2003 – Rockville, MD
• Created reporting application for testing and certification system with transactions exceeding 50,000 per day.
• Used JSP/Servlet technology and open source projects to run the web application on a BEA Weblogic Server 7.1.
• Produced requirements and design documentation using Visio for the system while following extreme programming methodology; continually added reports when requested and persistently refactored existing code.
• Completed a code review of the new system with the development team.
• Participated in the business analysis of a new development project from the ground floor.

**International Society for Quality of Life Studies:** Webmaster – Summer 2003 to Present – Blacksburg, VA
• Main responsibilities are to maintain the website, create new pages, and update existing pages.
• Made the conference registration forms and publication orders look more like the PDF version and to update the total price automatically using JavaScript and ASP.
• Created the Educational Resources section, the SINET section, and the contact us pop up window.
• Added a free search engine to the website to be able to search all the unrestricted pages.
• The website is located at [http://market1.cob.vt.edu/isqols](http://market1.cob.vt.edu/isqols).

**Community Indicators Consortium:** Webmaster – Summer 2003 to Present – Blacksburg, VA
• Created the whole website using ASP but waiting on all partners’ approval before completing the other sections.
• Added the free search engine to be able to search the entire website.
• Came up with the concept for the logo and hired my artistic friend to create it.
• The website is currently located at [http://market1.cob.vt.edu/isqols/cic](http://market1.cob.vt.edu/isqols/cic).

**Industrial Finishes:** Head Catalog Creator – Summers 2001 and 2002 – Rockville, MD
• Created a 64 page full colored catalog of the store’s inventory and services using Photoshop and PageMaker.
• Converted catalog to PDF format for web access available on [http://www.industrial-finishes.com](http://www.industrial-finishes.com)
• Created a spray systems sales brochure that went out to 1,500 customers.
• Interacted with multiple clients and distributors to learn the best way to present each product.

**SpaceWorks:** Quality Assurance – Summer 1999 – Rockville, MD
• Performed beta testing for several business-to-business applications.
• Worked with telephony order entry/provisioning system and e-commerce applications.

**Awards and Activities**
President Fall 2004 to Spring 2005, and Webmaster Fall 2003 to Spring 2004 of Upsilon Pi Epsilon
GE/VT Student Leadership Conference: Spring 2003
Golden Key Club: Fall 2002 to Present
Softball Team Captain: Spring 2002 to Present
National Society of Collegiate Scholars: Spring 2001 to Present
Microsoft Student Developer Organization: Spring 2001 to Present
Career and Technical Education Award for Excellence in Networking Operations and Programming in all Montgomery County Public Schools: June 2000
Management Award (SpaceWorks): August 1999
Maryland Boys State: June 1999

62