

**A MULTIPLE-WINDOWS INTERFACE FOR  
INTERNET TOOLS**

**By**

**ROMONA M. BRISCO**

**Bachelor of Science**

**Langston University**

**Langston, Oklahoma**

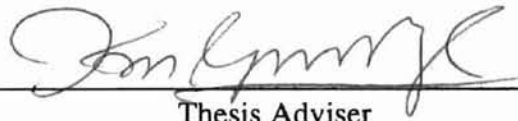
**1993**

**Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December, 1996**

---

**A MULTIPLE-WINDOWS INTERFACE FOR  
INTERNET TOOLS**

Thesis Approved:



Thesis Adviser



Dean of the Graduate College

---

## **ACKNOWLEDGMENTS**

I first would like to thank God, because none of this would be possible without him. I truly and sincerely thank my advisor and mentor, Dr. K.M. George, for all his time, help, support, counseling and guidance he has given me. I also sincerely thank and appreciate Dr. In Hai Ro at Langston University for his support and guidance throughout my college education. I also thank him for the knowledge he has given me in preparation for future success and achievements. I thank both of my committee members, Dr. J.P. Chandler and Dr. G.E. Hedrick for all their help.

I deeply appreciate the professors who gave me guidance and support outside their duty as professors. I also appreciate those professors, faculty and staff members of Langston University who supported me and provided me with the necessary background education and knowledge to further my education and future career goals.

I would like to give special thanks to my parents, Diane and Alfred Brisco, and my family and friends who supported all my endeavors. I thank my sister, Kimberley Brisco, and my brother, Amar Brisco, who was there when I needed them the most. Lastly, I like to thank those special friends that provided me with the help, support, confidence, love and encouragement I needed while obtaining my degree.

## TABLE OF CONTENTS

Chapter	Page
<b>1 INTRODUCTION</b> .....	1
1.1 Overview.....	2
<b>2 DEVELOPMENT TOOLS</b> .....	6
2.1 Existing Tools.....	6
2.2 Tcl/Tk (Tool Command Language/Toolkit).....	10
2.3 WISH (Windowing Shell).....	11
2.4 XF.....	14
2.5 EXPECT.....	21
<b>3 THE INTERNET TOOLS</b> .....	23
3.1 WWW (World Wide Web).....	23
3.2 FTP (File Transfer Protocol).....	25
3.3 Telnet.....	27
3.4 Rlogin (Remote Login).....	27
<b>4 MULTIPLE WINDOWS INTERFACE (MWI)</b> .....	29
4.1 Existing GUIs.....	29
4.2 MWI Design.....	31
4.2.1 Constructing the Interface.....	31
4.2.2 The Model.....	34
4.2.3 The Design.....	36
<b>5 CONCLUSION</b> .....	40
<b>BIBLIOGRAPHY</b> .....	43
<b>APPENDIX I Implementation (Source Code)</b> .....	47
<b>APPENDIX II Glossary</b> .....	58

## LIST OF TABLES

Table	Page
I. MWI vs. Other Web Browsers.....	41

## LIST OF FIGURES

Figure	Page
1. Command-line option using WISH.....	12
2. GUI using WISH at command-line option.....	13
3. XF Tools.....	14
4. Example GUI.....	15
5a. Parameters for “button” widget.....	18
5b. Parameters for “menubutton” widget.....	18
5c. Parameters for “menu” options.....	19
5d. Parameters for “text” widget.....	19
6. Final GUI.....	20
7. Example procedure for menu option “B” in menubutton.....	20
8. Expectk process.....	22
9. An Example HTML File.....	24
10. URLs for different protocols.....	24
11. Sample FTP session.....	26
12. Telneting to the Autonomous University of Barcelona.....	27
13. Using Rlogin to the Hubble Space Telescope Daily Report.....	28
14. Main Window.....	32

15. Internet Tools Menu.....	33
16. Result of accessing the WWW in MWI.....	33
17. The result of accessing WWW and a Window.....	34
18. An Active Automaton.....	36
19. MWI as an Active Automaton.....	37
20. Event Loop.....	37
21. Software Architecture.....	38
22. Tree Structure of MWI.....	39
23. Local University WWW Interface.....	39

# CHAPTER 1

## INTRODUCTION

This thesis develops a Multiple Windows Interface (MWI) for accessing Internet Tools from an X-terminal. One of the objectives of the MWI is to provide users with a simple point and click access to the internet tools World Wide Web, FTP, rlogin, and telnet. The major characteristic of MWI is that it provides a unified interface to several services in parallel. Users of MWI will benefit from this interface in many ways. First, for those users who do not have access to multiple windows, this interface provides them with multiple Xterms or windows if needed. Second, if users wish to retrieve information from more than one source, they can accomplish that with a few clicks of the mouse. They also can have access through the interface to the web and to an FTP site or another internet capability simultaneously. Third, this is a unified graphical user interface in which processes run simultaneously or concurrently which in essence saves time. Also, the user can roam freely between the windows to access his or her information. While waiting for something to load or download, the user can do something else useful. There is very little typing to do in this interface if one prefers it. The user should only have to type if Xterm windows are used and if he or she wants to access a different web page other than the ones provided by the interface. MWI provides users with one web page in which the user can access other pages by links. The links provided by this web page are



the ones that were mostly encountered on the net. Because this is a multiple window interface to access internet tools, it facilitates high resource utilization.

## 1.1 Overview

The use of Graphical User Interfaces (GUIs) is becoming the trend for today's developers. They are becoming the "dominant model for the user interface of microcomputer operating systems [16]". The command-driven and text-based interfaces are no longer the norm. "Command interfaces are thought to be too complex for infrequent end users, so a menu-driven interface may be substituted on end user systems [41]". Developers today want to provide users with an interface that is easier to use and to understand. They want an interface that is also more efficient and sound rather than a complex, difficult and unmanageable interface. According to [17], understanding what a user interface is and how to build one are the two sides to the user interface problem. Some users may face this problem while trying to create an application that satisfies their needs. User-Interface Management Systems (UIMS) and User Interface Development Environments (UIDE) aim to ease the burden of interface building [13][35]. Several UIMS and UIDEs are described in [3], [7], [13] and [35]. If developers are to develop a graphical user interface that is event-driven, it would satisfy most users in achieving their tasks. Building this type of interface would make constructing an interface easy to maintain and to develop [17]. Typing commands as well as learning commands can be tedious and difficult. Therefore, a GUI is an attempt to eliminate "typed commands" by allowing users to point at icons on the screen and click with a mouse to direct the computer to perform different tasks. This makes it easy for a beginner who may find it

hard to learn all the commands needed to use a command-driven interface. Also, it can be useful to an expert who might have otherwise forgotten a command.

According to [11], both the Apple Macintosh and Microsoft Windows proved that GUIs are easy to use and powerful. The definition for GUIs may be defined differently in some aspects by users or corporations, but essentially the whole concept used is similar by all concerned. In [38], a GUI is defined as one that “runs in computer graphics mode”. A “true GUI”, according to Microsoft, is one that offers a WYSIWYG (What You See Is What You Get) screen for output, “graphically oriented” using icons, “looks good” and is “easy to work with”, “manipulates on-screen elements”, and offers widgets such as menus, window objects, and “dialog controls [38]”.

Internet tools are used to find, retrieve or download information or resources available world-wide. Some internet tools used to achieve these tasks are the World Wide Web, File Transfer Protocol, Remote Login and Telnet. In [39], the author defines these internet tools and briefly guides the user through the internet and tells the user how to use each one of the tools. Users using these internet tools to accomplish such tasks find that retrieving or downloading information can become time consuming and could be more efficient in accessing information. A solution to this problem would be to provide a tool that allows simultaneous access of internet tools. Such a tool would allow users to obtain or retrieve information more efficiently and at a faster rate on the average. An approach would be to design a windows application that would accomplish this task. In [17], the author discusses how to design GUI applications that consist of factors that make a good interface. The factors stated in [17] are 1) having a consistent and strong interface so as

not to perplex and astonish the user, and 2) making the interface friendly by giving good “feedback”.

This thesis is concerned with the design and implementation of A Multiple-Windows GUI to support simultaneous access to internet tools. A multiple-windows interface (MWI) would allow a point and click access method for internet tools. It is a user interface, constructed to be event-driven by mouse clicks. Even though the interface will not speed up a specific task, by providing a means of doing several tasks in parallel, it supports overall efficiency. Factors that need to be considered in constructing such a user interface would be 1) the tools used in building the interface and its efficiency as to how fast one can access information via the interface, 2) programming language and the efficiency of that language in programming specific tasks needed in the interface, 3) the difficulty level of programming in constructing the interface, 4) user-friendly environment, and 5) performance.

The MWI is different from present interface tools that appear to be similar. The MWI will be 1) a non-commercialized product, 2) a multiple window, multitasking interface in performing tasks in parallel and 3) an interface designed for X-Windows which requires no initial interface related setup for accessing windows or internet tools.

MWI is constructed under the UNIX environment on the Sequent system. It is implemented in a shared memory multiprocessor machine. MWI is designed, constructed and programmed to access multiple windows and tasks at the click of a mouse. MWI will allow users to transfer between windows while waiting for information, therefore the user always has an option of doing some other task. MWI has a unique look as well as easy to follow menu options. Lastly, MWI has an excellent help menu on all objects and tasks in

the interface. We provide a survey of work similar to MWI in Section 4.1. The next chapter is devoted to GUI development tools.

This thesis is organized into five chapters and two appendices. In Chapter 2, overall views and descriptions of development tools are given along with how the interface works using these tools. In Chapter 3, emphasis is placed on the internet tools and how this interface is designed to access these tools. Chapter 4 focuses on the MWI design. Chapter 5 is the concluding chapter of this thesis. MWI is compared to other graphical user interfaces (GUIs) that are currently available in the public domain, and a section of the concluding chapter presents further research work and development that will enhance the MWI making it more efficient and more universal to use. Appendix I will focus on the code that was used to implement the internet tools section of MWI. Appendix II contains a glossary.

## **CHAPTER 2**

### **DEVELOPMENT TOOLS**

MWI is developed using GUI creation tools, and in this chapter, several tools are reviewed and described.

#### **2.1 Existing Tools**

There are several tools or applications that are available for constructing graphical user interfaces (GUI). User interfaces can be built using Microsoft Windows, SQL 3.0, Motif, Visual Basic, XF, the X toolkit, the Tk toolkit and others. The spectrum of tools vary from high-level programming languages to special purpose packages. Before developing, designing and programming a GUI, the tools used in constructing the interface should be examined. The tools used should make the interface easy to build, to maintain and to extend. The user should be able to build a simple interface using the tool without difficulty or confusion. Also, the user should be able to maintain the interface by making modifications or alterations to the original interface without any complications. In this thesis, a tool is considered efficient if it helps to reduce the development time. There are several factors that contribute to efficiency: 1) template of widgets that can be configured with mouse clicks only; 2) little or no programming on the part of the user is required; 3) in depth knowledge of any programming language is not required. Being proficient in a

certain language in order to construct an interface does not make it user-friendly. The overall quality of the finished interface is that it should be easy to use and to understand, well-designed and fast in accessing information [21][38].

In [36], the authors discuss the creation of GUIs using 3GLs (Third-Generation Languages) and OOP (Object-Oriented Programming) languages. 3GLs are languages such as Fortran, Pascal, and C. An example of an OOP language is C++. According to the authors in [36], the code for a simple windows application using C/Windows API (Application Programming Interface) programming was complex and difficult for mainframe developers. Therefore, using this software to construct GUIs would require experienced C programmers, but this method would only be easy for them. However, users not fluent in C or other languages who want to build windows applications would find a particular software method that generates the code for the interface useful. Methods like this will generate the code for the user by a simple mouse click on buttons, icons, menus or other objects. Some methods that are designed to do this are Microsoft Windows, Visual Basic, Digitalk, Powerbuilder 3.0 and XF.

Microsoft Windows Applications creates multiple windows and one or more applications may be displayed concurrently on the screen. A user can create this application by clicking on objects in the application. Another application which provides similar features is Motif [22]. Although Motif and other applications are similar in features and characteristics, there are distinct differences that make each application unique. For instance, "there are pronounced differences between Microsoft's GUI standard and the GUI standards of Motif and other applications on the Unix and VAX/VMS platform [36]". Using Microsoft Windows one can develop a PC-based

application, because it “is a multitasking, object-oriented, graphically oriented platform, users are presented with a simple and consistent interface that allows them to interact with an application using graphical constructs as opposed to operating system commands [36]”. Therefore, the users can create an application easily by clicking on boxes, buttons, pulldown menus and other widgets without the task of programming. This application should be considered when wanting to construct an interface because it satisfies most of the requirements stated in the introduction. On the other hand, GUI development with Motif is difficult because the process can be frustrating [9]. Some frustrations that arise using Motif are that Motif has no help information, a lot of useful widgets are not part of the library, there is a lack of widgets, and the awkwardness of Motif API [9]. This method would not be a good idea to use in building an application based on the degree of difficulty and frustration if a Windows type tool is available.

The Macintosh GUI has three different tools to use in order to create a GUI. The Finder, is a “file management program that consists of a menu bar that manages files using icons and windows [34]”. The User Interface Toolbox allows developers to create text-entry windows and use dialog boxes to browse directories. The Quickdraw handles windows and menus. In the article [34], the author does not discuss how each of these elements were implemented, but how they are used and their purpose in creating the GUI. Also in the same article, the developers that the author talks about in creating the Macintosh GUI using these elements are the Apple developers who seem to have experience. Although the Macintosh Finder is easy to use, the elements involved in creating this package would seem complicated for beginners trying to develop a similar

GUI [34]. Therefore, compared to other methods, using Macintosh would pose a high level of difficulty.

X-Windows, which runs under the Unix environment provides a user-friendly interface with no pre-defined windows. The toolkit used is the X-toolkit which uses a Motif widget set built on Xlib ("X-library is a C language interface to the X Window System, which includes numerous data types, constant definitions, macros and about 500 C library routines" [43]). Therefore, "programmers can directly program in Xlib, however it requires a lot of redundant code to maintain the conventions [14]". Although using X-Windows along with the X toolkit seems sound and convenient to use, developers unfamiliar with Xlib programming will find it difficult. Coding is not required if widgets are used, but at some point a user may have to program a specific task in the interface. X-Windows may provide an efficient interface, but one of the main goals of interface development is that the user must be comfortable overall using the toolkit to build the interface and handle the programming language or tools involved as well. Lastly, a toolkit that seems to answer all the questions and concerns in constructing an interface would be the Tk toolkit used along with the Tcl programming language and the XF application. Tk creates all kinds of widgets including menus, icons, buttons and other objects. Tk is a better choice than the tools used by Microsoft Windows and Macintosh, Motif, and the X toolkit because not only is it easy to use and the programming is for beginners, but it also provides a user-friendly and efficient interface that can be used to create any interface [29].

There is no toolkit or application that is perfect or that will satisfy everyone. However, a toolkit that 1) makes the interface easy to construct and maintain, 2) has



limited programming or easy to learn programming tools, 3) is efficient and 4) is user-friendly is a better choice than the other toolkits. In comparison, the Tk toolkit using Tcl as the programming language and XF as the interface development tool ranks high among the tools that were previously reviewed. Tk and Tcl can be used to build an interface without the XF application, however, if the XF application is not used then the Tk toolkit along with Tcl language maybe grouped in the category with the other toolkits and applications because some users may have a high degree of difficulty. Users will have the tasks of programming the entire interface and will need to know more than just the basics of the Tcl language and Tk toolkit. Clicking on buttons, menus and other widgets without having to go through layers of menus to select a widget, the color or configuration of templates, provides users with an efficient interface in terms of development time compared to other methods. Use of programming is only needed to construct new widgets or to provide specific commands or tasks used in interacting with the interface. Step by step instructions, code and graphics are provided by the Tk toolkit, Tcl language and the XF application. In the next four sections, we review Tcl/Tk, WISH, XF and Expect. These four development tools were used to implement MWI.

## 2.2 Tcl/Tk

Tcl stands for “tool command language” and is a simple scripting language for controlling and extending applications. Its interpreter is a library of C procedures that can be easily incorporated into applications and each application can extend the core Tcl features with additional commands for that application [29]. Therefore, one can write Tcl

code within a C program. Benefits in using Tcl are rapid development, user convenience and including different library packages.

Tk, a toolkit for the X-Window System [29], is a Motif-like widget set that gives access to the widgets via Tcl commands [7]. This toolkit “allows a user to create GUIs for the X Window System by writing Tcl scripts [29]”. Tk creates all kinds of widgets including menus, icons, buttons and other objects. Tk is very easy to use. Step by step examples of how to build and construct menus, icons and other widgets and the code for them are show in [29]. These step by step examples are very helpful in constructing simple to more complex GUIs. Therefore, beginners would have no problem achieving their goal using this toolkit.

### **2.3 WISH**

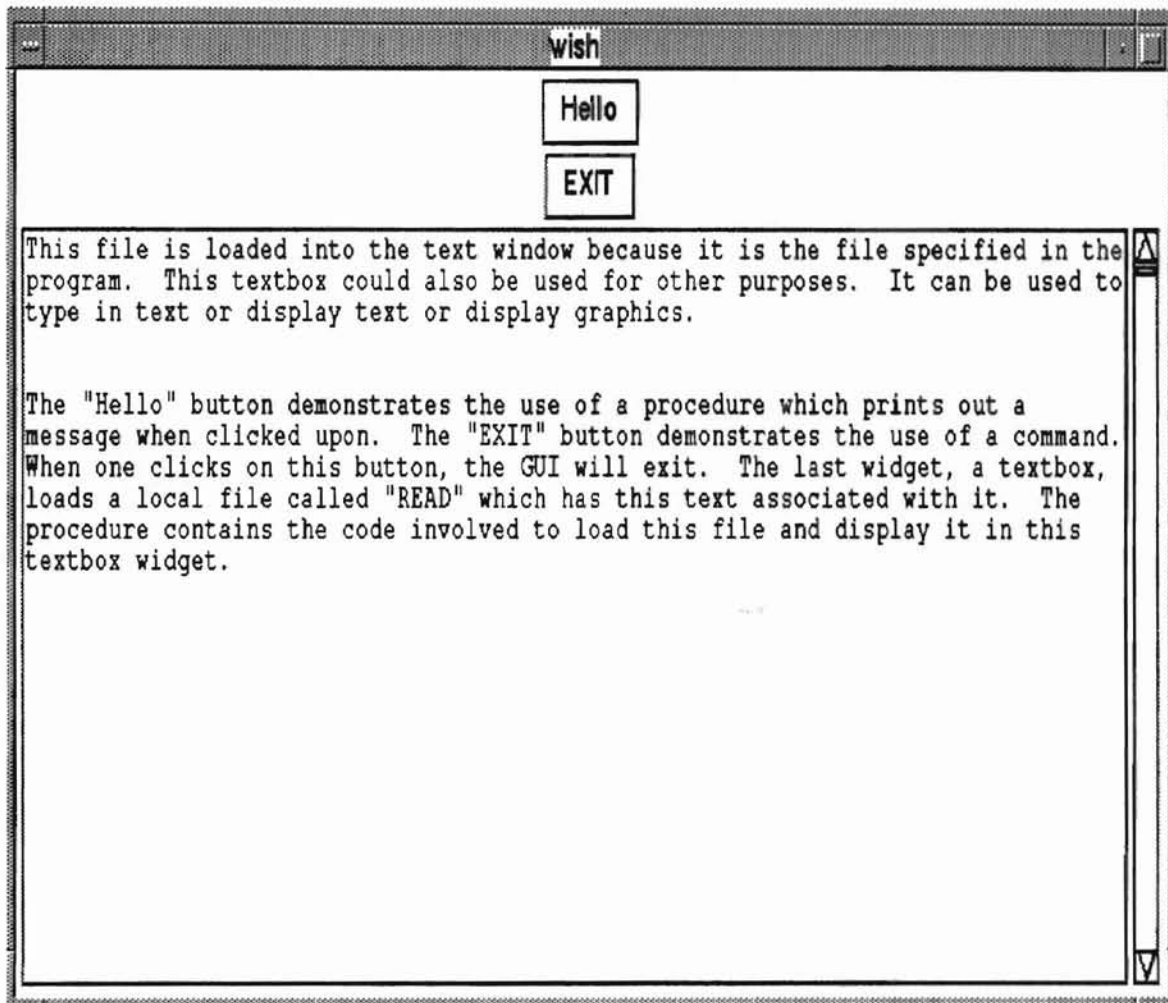
WISH, a windowing shell, is a program that consists of Tcl and Tk. WISH reads commands from the command-line or from a file and displays the output to the screen. If one prefers the command-line option then one would type Tcl commands at the command-line. WISH then will interpret these Tcl commands and build an interface according to the specification provided by the user. However, if the file option is preferred, WISH will load the specified file which has to be in the Tcl/Tk language. Figure 1 illustrates the command-line option using WISH which specifies an interface. Figure 2 is the resulting interface created by WISH using this method.

After MWI was constructed, the interface was tested using WISH. Initially, WISH was used also in debugging code for new commands and procedures for tasks/events.

WISH is also used in connection with XF. Once a GUI is created in XF one can load it in WISH instead of in XF. However, adding new commands and procedures at the command-line in WISH is not suggested for beginners and programmers with little knowledge of Tcl/Tk [47]. WISH is a good tool in developing interfaces.

```
% button .b -text "Hello" -command HI
.b
% proc HI {} { puts stdout "Hello there stranger!"
}
% pack .b
% button .b1 -text "EXIT" -command exit
.b1
% pack .b1
% text .text -relief raised -bd 2
.text
% scrollbar .scroll -command ".text yview"
.scroll
% pack .scroll -side right -fill y
% pack .text -side left
% proc loadFile file { .text delete 1.0 end
    set f [open $file]
    while {[eof $f]} {
        .text insert end [read $f 1000]
    }
    close $f
}
% loadFile READ
% □
```

Figure 1. Command-line option using WISH.



**Figure 2. GUI using WISH at command-line option.**

## 2.4 XF

XF played an important role in the development of MWI. Therefore a very detailed description of XF is presented in this section.

XF is an integrated programming environment that supports the development of GUIs [7]. XF is a user-friendly tool in which one could easily design and implement a graphical user interface. In fact, XF is a UIDE for developing GUIs. Different widgets and menus provided by XF can be used to create well-structured interfaces. Refer to

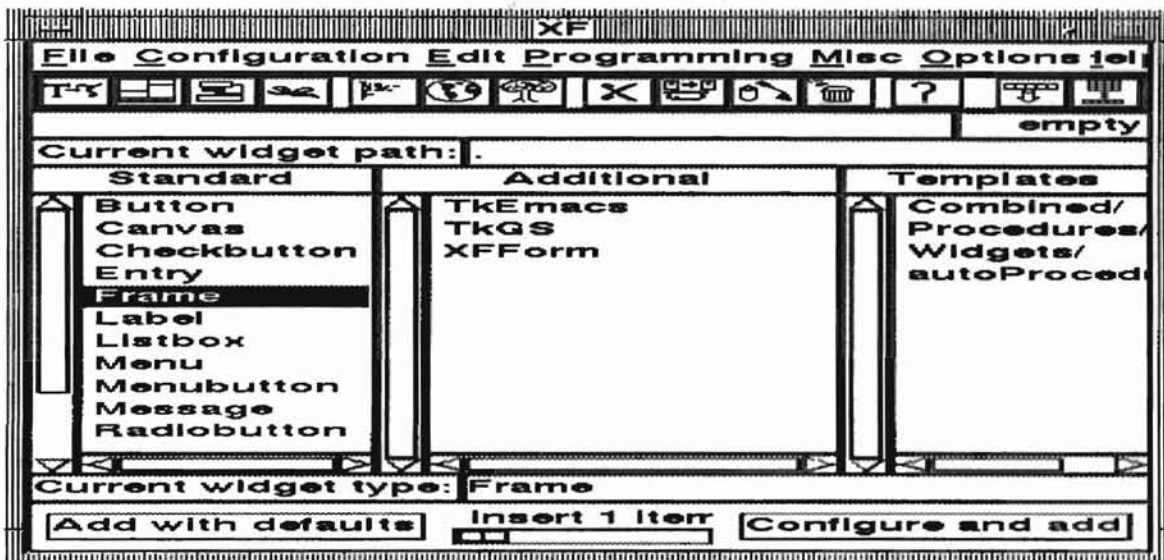
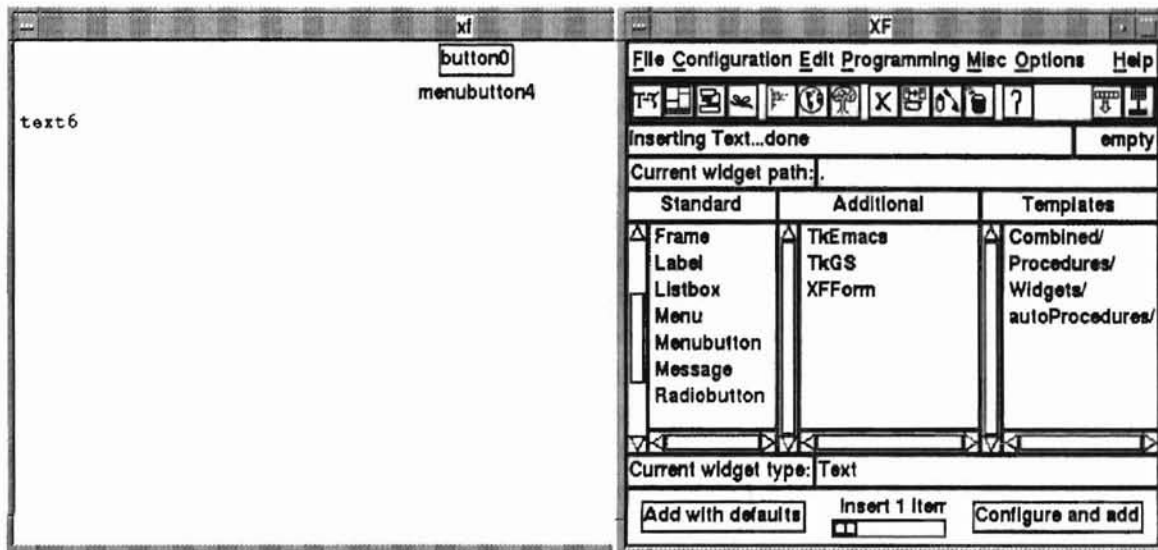


Figure 3. XF tools.

Figure 3 for the tools XF provides for interface development. XF constructs widgets such as buttons, menus, lists, etc. with mouse clicks. Figure 4 illustrates a Window created using XF. The example GUI in Figure 4 contains 1) a button that displays a message when clicked upon, 2) a menubutton containing a menu of options that displays messages and clears the textbox, and 3) a text widget containing text. All of these widgets were created by mouse-clicks. This tool allows easy manipulation of existing interfaces constructed under XF as well. XF assists in the creation of interfaces by relieving the

developer of the task of coding or programming to great extent. The code is automatically created once a widget is selected and the action is invoked by coding commands. In coding such commands, one can create their own events or special tasks by writing Tcl code. For example, to create the sample GUI in Figure 4, one would type in “xf” at the prompt to start the application. To begin, one would double-click the leftmost mouse button on “button” to create a button widget. Next, double-click the



**Figure 4. Example GUI.**

leftmost mouse button on “menubutton” to create a menubutton widget. After inserting the menubutton widget, double-click the middle mouse button on the widget, move to the main window of XF and double-click the leftmost mouse button on “menu”. This accomplishes the creation of a menu of options. To create the text widget, set the “current widget path” located in the main window of XF to “.”. This means that widgets are created in the root window. To set the current widget path one would double-click the middle mouse button in an empty space in the work section of XF, in this case, underneath the menubutton, or double-click in the “current widget path” space with the

middle mouse button until "." is displayed. If one is inserting a widget and the "current widget path" is not set to root then one may accidentally insert a widget within a widget. This causes an error if the widget cannot include another widget. For example, a menu or menubutton widget can contain other menu or button widgets, but a button widget cannot contain another button widget; i.e. button on top of a button. All the widgets are contained in the "Standard" column in the main window of XF. However, one could use a template in the "Templates" column to modify/create new widgets. After inserting the widgets, the action/event behind the widget has to be implemented. One can either save the GUI and edit the code or do it in XF by selecting "Commands" and/or "Procedures" from the "Programming" menu in the main window of XF. One could configure the widgets before or after coding the tasks or events by double-clicking on the widget with the rightmost mouse button and specifying the requirements, or one can select "Parameters" from the "Configuration" menu in the main window of XF. The widgets in Figure 4 were given the parameters shown in Figures 5a, 5b, 5c, and 5d respectively. The results are shown in Figure 6. While specifying the parameters, if one chooses to give the widget a command at that time then one would specify that command in the "Command" box in the parameter window. The procedure for menu option B in the menubutton widget is shown in Figure 7 as an example. Procedures, events, commands and widgets can also be written in the C programming language. However, in this case all procedures, events, commands and widgets were created by XF and/or Tcl code. By one creating his or her own events or tasks, one could construct and design a very good interface. Once the interface is designed and saved as a file, it can also be edited or modified to produce new code for widgets, commands and procedures. For the example GUI in Figure 4, one

would go to the "File" menu in the main window of XF using the mouse, select "Save As", specify the name of the file in the "Filename" space provided and then exit XF by selecting the "Quit" option in this menu. To edit the file for making modifications, such as creating new commands, one would use an editor in which he/she is already familiar.



**Button parameters::button0**

Widget path: .

Name: button0

Symbolic name:

Background: White

Bitmap:

Font: -Adobe-Helvetica-Bold-R-N

Foreground: Red

Label: A button

Size: 

Width	Height
0	0

Text variable:

Command:

Special | General | Geometry | Binding

OK | Apply |  Apply permanently | Cancel

Figure 5a. Parameters for "button" widget.

**Menubutton parameters::menubutton4**

Widget path: .

Name: menubutton4

Symbolic name:

Background: White

Bitmap:

Font: -Adobe-Helvetica-Bold-R-N

Foreground: Blue

Label: A menubutton

Name of menu: .menubutton4.m

Size: 

Width	Height
0	0

Text variable:

Underline:

Menu | Special | General | Geometry | Binding

OK | Apply |  Apply permanently | Cancel

Figure 5b. Parameters for "menubutton" widget.

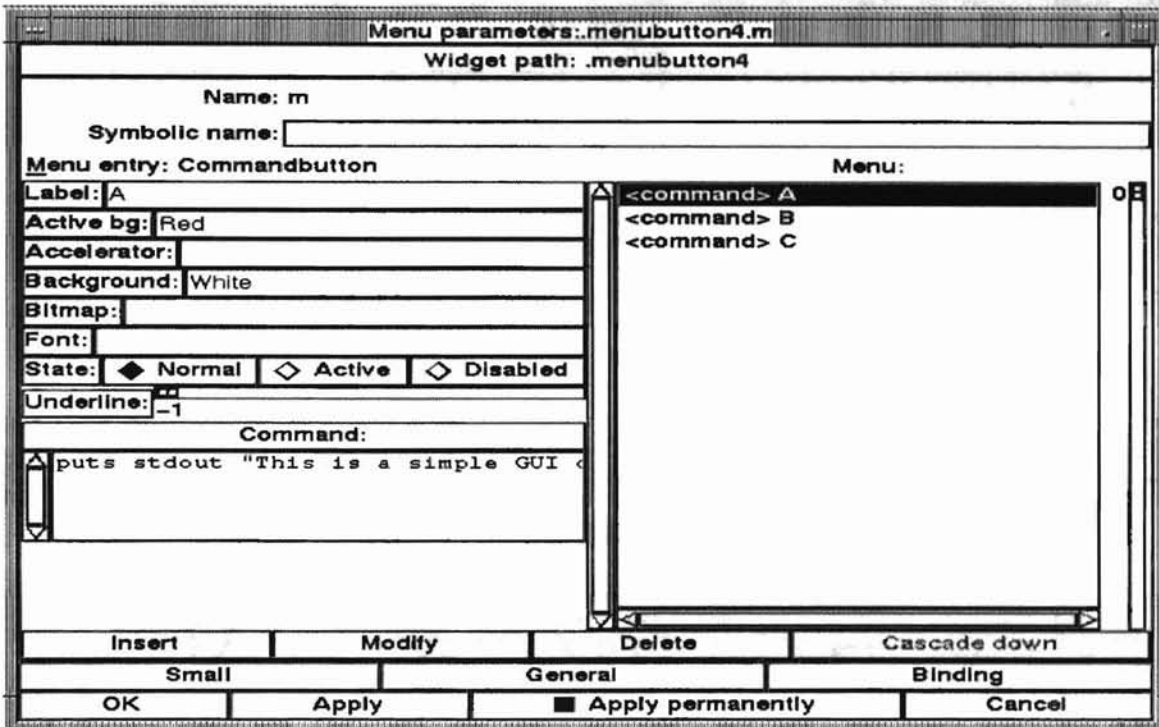


Figure 5c. Parameters for “menu” options.

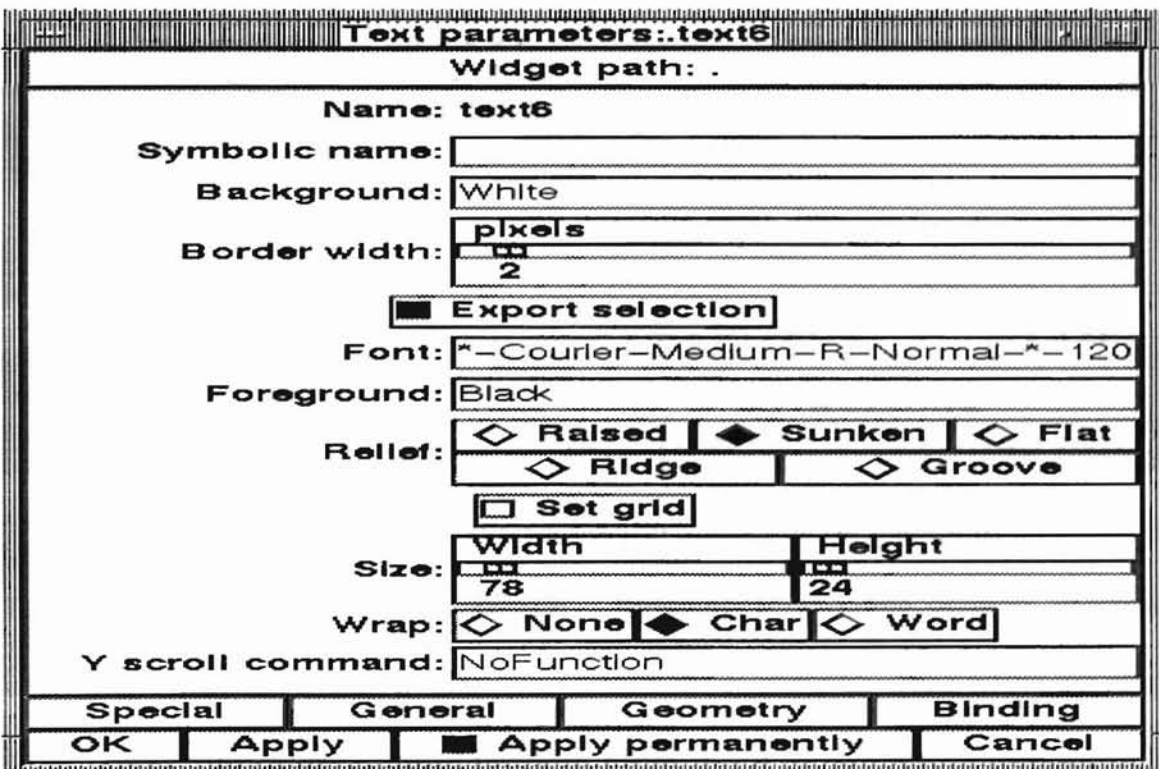


Figure 5d. Parameters for “text” widget.

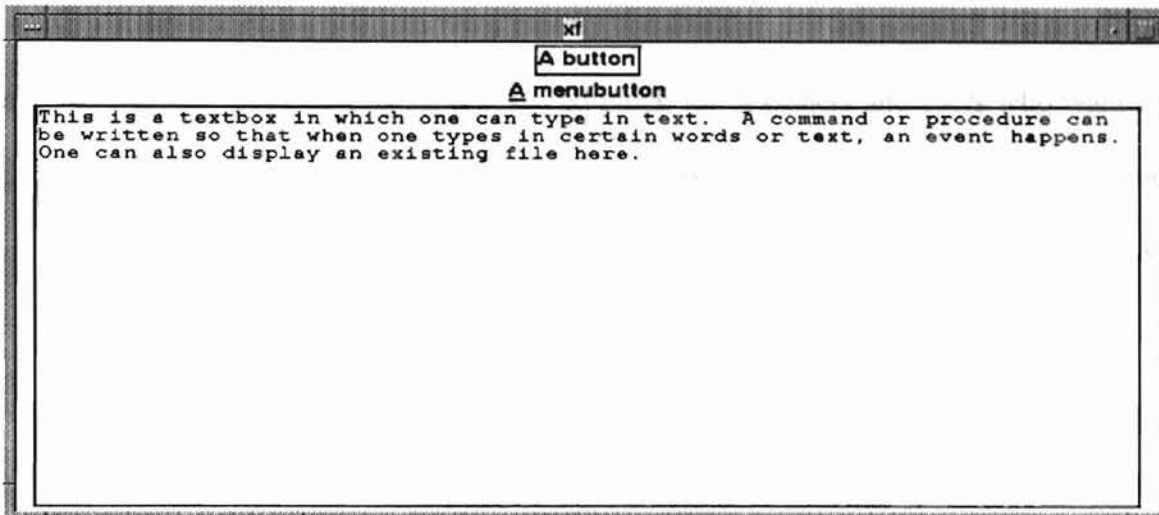


Figure 6. Final GUI.

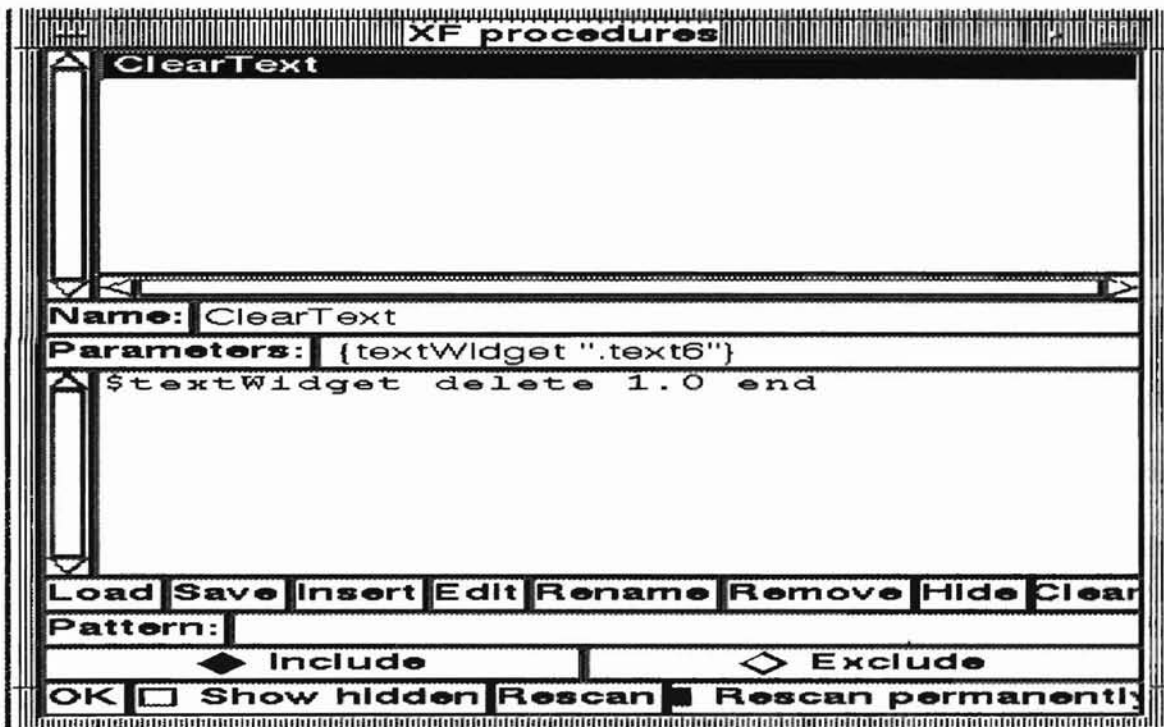


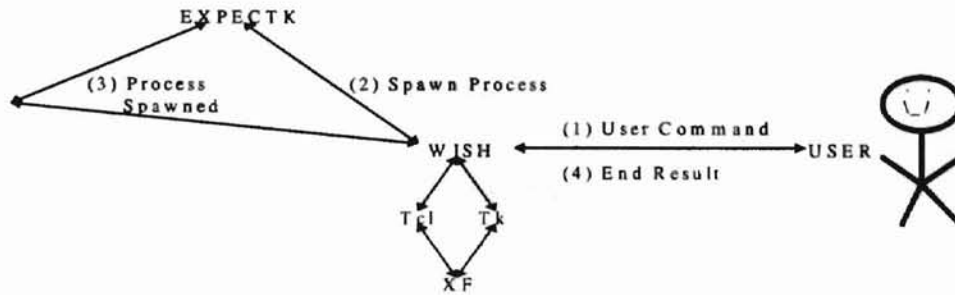
Figure 7. Example procedure for menu option "B" in menubutton.

There are several other tools in XF for constructing very simple to more complex interfaces. XF has other uses as well. However, in the example GUI, one would not need all of the tools, only those specified for the example. For more information on other tools and uses of XF, refer to [7].

XF constructs interactive interfaces rapidly because the user accesses all widgets from a list [7]. This makes development easy and quick for beginners and users who want an efficient interface done in little time. These benefits make XF flexible and less time consuming in development. The user can create an easy or more complex interface by adding more capabilities. In the instance of this thesis, I have added events to the interface by adding Expect code. Expect is explained in further details in the next section. According to [7], using XF will not only “result in better code” but will also allow users to develop an interface in a short period of time with little help from the tutorial. In [7] the author discusses the benefits from using XF such as fun, flexibility, support for “group development and standard interfaces” and “immediate access to the resulting interface” [7].

## **2.5 Expect**

Expect is a program that is used to control interactive programs and can be written in C, C++ or Tcl [19]. Expect can do what any shell script does, but mainly is used for automating programs. Expect is easy to use and is used in real applications [19]. Expect can be combined with Tk to produce Expectk that allows one to build X applications. As a general purpose language, Expect can be called a communicating script because it is specifically designed to interact with interactive programs [19]. Figure 8 shows an example of the user communicating with an interactive process via Expect. Expect is also useful for spawning processes, handling multiple processes simultaneously, interacting with multiple processes and adding



**Figure 8. Expectk process.**

extensions. In creating the MWI, Expectk code was mainly used in creating the event of “Window” and the accessing of “Internet Tools”. This code was used to implement these tasks because the user would use the window like a shell expecting a prompt and a response in return and access the internet tools simultaneously while interacting with MWI. Therefore, communication from the interface by the window via Expectk would allow this process to happen along with the technique of automating internet tools and displaying output to the user in the window provided. For example, if one would open a window in the interface then the Expectk script sends a command to the interface, and vice versa, to execute the task/event. This tool made it efficient in providing the necessary elements to complete specific tasks in the MWI.

## CHAPTER 3

### INTERNET TOOLS

Internet tools are used by people who want to obtain information or resources. These tools can also be used for other purposes. The Internet Tools mentioned in this thesis, the World Wide Web (WWW or the Web), File Transfer Protocol (FTP), Remote Login (rlogin) and telnet can be accessed simultaneously by any user of MWI. These tools are discussed in the next sections.

#### 3.1 World Wide Web (WWW)

The WWW was developed around 1989 at CERN (Conseil Europeen pour la Recherche Nucleaire or European Organization for Nuclear Research) at the European Laboratory for Particle Physics [44] by Tim Berners-Lee and Robert Cailliau [37][45]. The WWW has become the “fastest growing part of the Internet [2]”. The Web uses two protocols as a means of obtaining resources and data. First, Hypertext Transfer Protocol (HTTP) is a client-server protocol where the client is a WWW browser making a request to the server for information. The WWW server responds to the request by transferring the information requested back to the client. Second, Hypertext Markup Language (HTML) “files are text files containing the elements of the documents [5]”. HTML files contain tags which denote the beginning or end of an element such as a paragraph or

header. Figure 9, taken from [5], illustrates this example. To access a site via the Web, one would give a Universal Resource Locator (URL) as described in [37] or Uniform Resource Locator (URL) described in [39].

```
<HTML>
<HEAD>
<TITLE>example of basic HTML</TITLE>
</HEAD>
<BODY>
<H1>Example of basic HTML document</H1>
<HR>
<P> The format in which this HTML document is displayed
      depends on the browser used to view it.</P>
<HR>
<ADDRESS>djb / djb@acm.org</ADDRESS>
</BODY>
</HTML>
```

**Figure 9. An Example HTML File taken from [5].**

This locator opens the address specified in the URL. Figure 10, taken from [39] is the format used for accessing site addresses for different protocols. These protocols include http, gopher, ftp, news and mail for their respective servers. The format for these protocols should be in URL form such that “URL addresses should be understood as follows: <protocol>://<computer Internet address>:port number/file or directory> [39]”.

```
http://www.informs.org/
gopher://sil.s.umich.edu:1704/
news:sci.op-research
mailto:msodhi@umich.edu
```

**Figure 10. URLs for different protocols taken from [39].**

People are using the web for many different purposes. Using the Web is very easy and it is also universal. For this reason, and because of its growth, the Web has “become synonymous with the Internet [39]”. The WWW “is a technology that allows you to weave related information on the Internet into hypertext documents [39]”. Therefore, one

could easily access information or resources on the internet by clicking on hypertext links. These links are created by using HTML which will bring about Web browsers. Web browsers are programs that read Hypertext Markup Language (HTML) files, access newsgroups and web servers, have e-mail capabilities, find information quickly and “make information quickly and inexpensively available worldwide [39]”. Some examples of web browsers are Netscape, Mosaic, and Lynx. Most web browsers display both in text-only and graphics. However, Lynx is a text-only browser.

### **3.2 File Transfer Protocol (FTP)**

FTP and anonymous FTP are “used for both a communication protocol (file transfer protocol)” and for transferring files from remote computers to one’s local computer [39]. For anonymous FTP, one would not have to have an account (userid and password) on a computer but anonymous access instead to transfer files between machines. Two ways to access ftp sites are by web browsers and graphical ftp programs. Using these options makes it easy because downloading or viewing files are done by point and click access. Also, by using these two options, one can automatically decompress and view, retrieve and save files. Basic commands for retrieving and storing files are “get filename” and “put filename” respectively. FTP also has e-mail capabilities called Ftpmail. Ftpmail “is a special type of e-mail server” [39] that sends archived files to a user. The basic five steps of FTP according to [26] are “connect, log in, change directories, grab the file, get out”. Figure 11, taken from [26], shows an example of an FTP session using these basic steps.



```
Step 1
home> ftp wuarchive.wustl.edu
Connected to wuarchive.wustl.edu
220 wuarchive.wustl.edu FTP server...ready.

Step 2
Name(wuarchive.wustl.edu:align):anonymous      (Log in as "anonymous.")
331 Guest login ok, send your complete e-mail address as password.
Password: align@                               (You won't see what you type,
                                               but enter your e-mail address here.)
230 Guest login ok, access restrictions apply.

Step 3
ftp> cd mirrors/msdos/database                 (Change to appropriate directory.)
250 CWD command successful.

Step 4
ftp> binary                                   (Set file type to binary if necessary.)
200 Type set to I.
ftp> get roadmile.zip mileage.zip             (Retrieve file and rename to mileage.zip)
200 PORT command successful.
150 Opening BINARY mode data connection for roadmile.zip (55711 bytes).
226 Transfer complete.
local: roadmile.zip remote: roadmile.zip
55711 bytes received in 67 seconds (0.81 Kbytes/s)

Step 5
ftp> quit
221 Goodbye.                                 (Disconnect and return home.)
home>
```

Figure 11. Sample FTP session taken from [26].

WILSON STATE UNIVERSITY

### 3.3 Telnet

Telnet is a tool that allows one to login to remote computers and obtain information that is located at the remote site. One could login remotely using a user id and password. However, some sites allow public access. One would have to know the address of the computer in order to complete a telnet command. An example in Figure 12, taken from

```
host> telnet babel.uab.es
Trying 158.109.0.14...
Connected to babel.uab.es.
Escape character is '^]'.
Connecting...
Connection #1 established to BABEL.UAB.ES
Ordinador BABEL de les
Biblioteques UAB. Entreu HELLO UAB.BIB:
```

Figure 12. Telneting to the Autonomous University of Barcelona [27].

[27], illustrates a telnet session. This example shows a telnet to the Autonomous University of Barcelona by command-line instructions. There are also other ways to telnet. However, in this thesis, the example shown in Figure 12 is appropriate.

### 3.4 Remote Login (rlogin)

The rlogin command is generally the same as telnet. However, one could set up aliases to make both the connection to the host and to log in, and “the username at the remote computer can be sent from the command-line [27]”. This makes rlogin an easier choice than telnet. An example rlogin session is given in Figure 13.

```
host> rlogin stinfo.hq.eso.org -l stinfo
Last login: Sat Sep 25 04:40:48 from 153.90.2.2
SunOS Release 4.1.3(MC3) #1: Tue Jun 15 14:32:45 MET DST 1993

>> Welcome to the STINFO Bulletin Board
    (Revised, June 1993)
>> Enter your E-mail address: align@montana.edu

Please choose a bulletin board:
HST status reports (h)
European HST news (e)
Quit (q)
Enter h, e, or q>
```

Figure 13. Using Rlogin to the Hubble Space Telescope Daily Report [22].

## CHAPTER 4

### MULTIPLE WINDOWS INTERFACE (MWI)

In the previous chapters we described the tools used in the development of MWI and the target of MWI. The focus of this chapter is MWI. The underlying model and software architecture are the topics of this chapter.

#### 4.1 Existing GUIs

There are several GUIs that support internet applications. Multithreading and multiple documents are two features that distinguish some of these applications. There are few tools that provide these techniques because most applications either do one or the other or the application does not provide simultaneous access between internet tools. A few applications that provide the techniques are Microsoft Windows [36][38], X-Windows [14], and Netscape Navigator [8][31]. Other applications that are similar to these but do not provide the technique described above are discussed in [23] and [24]. The applications discussed are either single purpose applications or applications that do not provide simultaneous access to internet tools.

Ventanna Mosaic is a browser that supports multiple documents but not multithreaded downloads [8]. Another application, SPRY Mosaic, was developed in 1989 by SPRY

Incorporated, a seattle-based company "known for its TCP/IP stack" [33]. SPRY was founded in 1989 and "was acquired by CompuServe Incorporated in 1995" [46]. SPRY is now called CompuServes Internet Division. SPRY Mosaic contains a network file manager that "allows multiple active FTP sessions open for managing remote drives [8]". SPRY Gopher, allows the viewing of gopher sites in multiple windows. Although SPRY contains the internet tools, mail, news and gopher and the facility for viewing mail and gopher sites by multiple windows, SPRY does not access different internet tools simultaneously. Another application that accesses internet tool applications is MultiNet for Windows. Although MultiNet for Windows includes internet tools and multiple telnet sessions that can be open at one time, again, there is no simultaneous access between the internet tools provided. The Netscape Navigator browser supports not only multiple documents, but can "open multiple windows on one machine" [8] and is considered to be multitasking. However, Netscape does not provide parallelism in accessing internet tools. Netscape does, however, "perform multiple simultaneous downloads [18]". What Netscape does not provide is an easy way to telnet and so "in order to access the information stored on Telnet resources," one will "need a Telnet and a 3270 helper application [31]". This means that one's computer is turned into a dumb terminal (VT100) that allows text-only access to mainframe data. To accomplish this, one would have to first install and configure the "helper" applications. Therefore, Netscape is not applicable and not as efficient as the MWI. Another application, Internet In A Box "is an all-inclusive multimedia Windows interface providing a suite of Internet applications including E-mail, newsgroups, Telnet, Gopher, and Mosaic [32]". Although this application supports the above internet applications and "works with both PCs and

local area networks" [32], the internet applications can not be accessed simultaneously and the internet tool FTP is "lacking" in some aspects [30]. Lastly, an application that has the "ability to manage several concurrent windows each having their own window to its own virtual PC" [38], is the Windows 386 version. This application is multitasking, allows multiple windows and allows programs to run concurrently but it is complex and is DOS-based.

## **4.2 MWI Design**

### **4.2.1 Constructing the Interface**

The MWI was constructed using the development tools discussed in chapter 2. The MWI is made up of widgets, icons, menus, buttons, and labels. Each widget and icon in the interface provides the user with some action whether it is displaying a message, providing some specific information, providing help or accessing one of the internet tools. The primary objective of the interface as mentioned in the introduction is to assist the user to access internet tools concurrently. The main interface, shown in Figure 14, provides the interconnection for the set of functions a user may access. The button, Internet Tools, at the bottom of this window provides the user with a smaller window containing a menu and some information to get him/her started on internet applications. This window is shown in Figure 15. From this window, a user can choose an item from the menu which contains other items. These menu items include internet tools that the user can access. Once the user clicks on the menu item and drags the cursor to the submenu of that particular internet tool, the action will access an internet tool or perform some other action. An example of the next level is shown in Figure 16. In this case the user has chosen the Web. While accessing an internet tool, the main menu window will

stay active on the screen until the user has exited this window. This feature allows a user to select other options to access different internet tools. The main menu window should be exited after a user has finished using MWI. There is a miscellaneous menu option in the main menu window in which the user can choose to quit from its submenu. The main menu of the "Internet Tools" window is currently the most important part of the interface because it allows access to the internet tools provided. If one chooses, accessing more than one tool or task in MWI can also be accomplished. An example is shown in Figure 17. In this figure, one would access the WWW from the internet tools menu and click on the button "Window" from the main window in MWI.

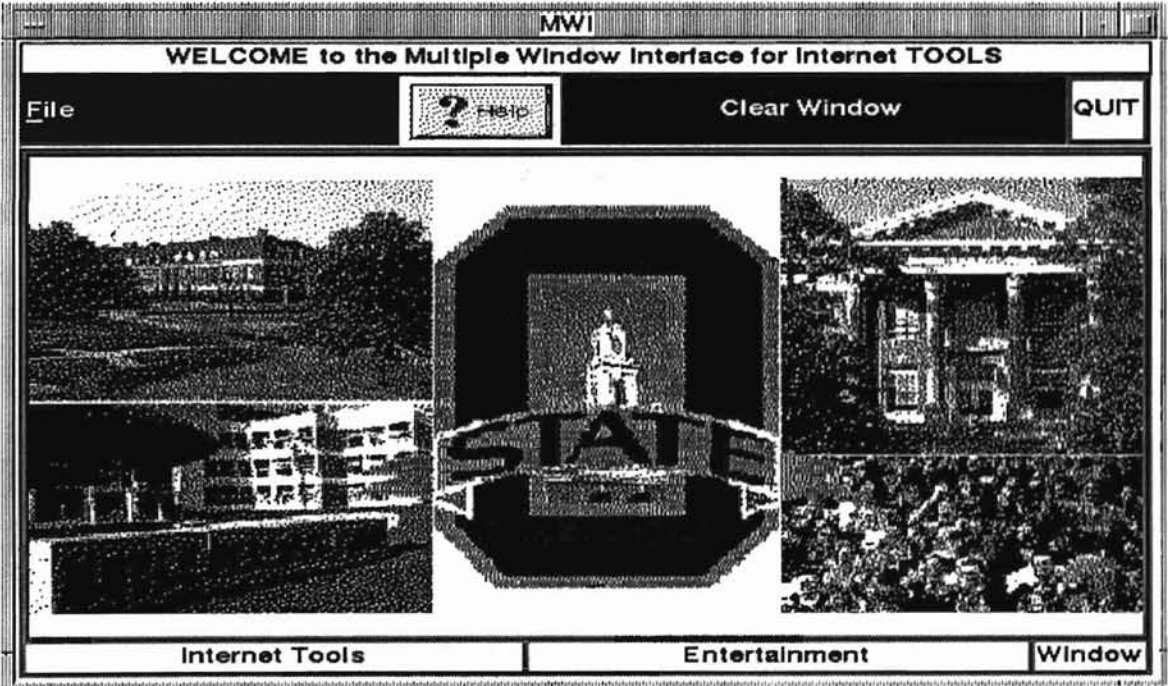


Figure 14. Main Window.

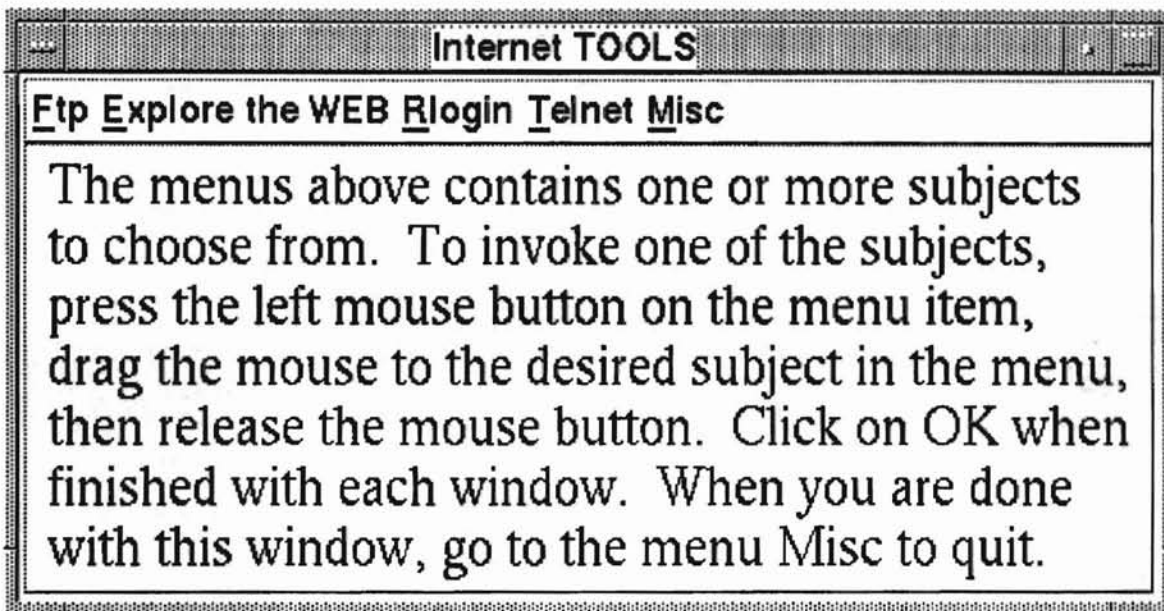


Figure 15. Internet Tools Menu.

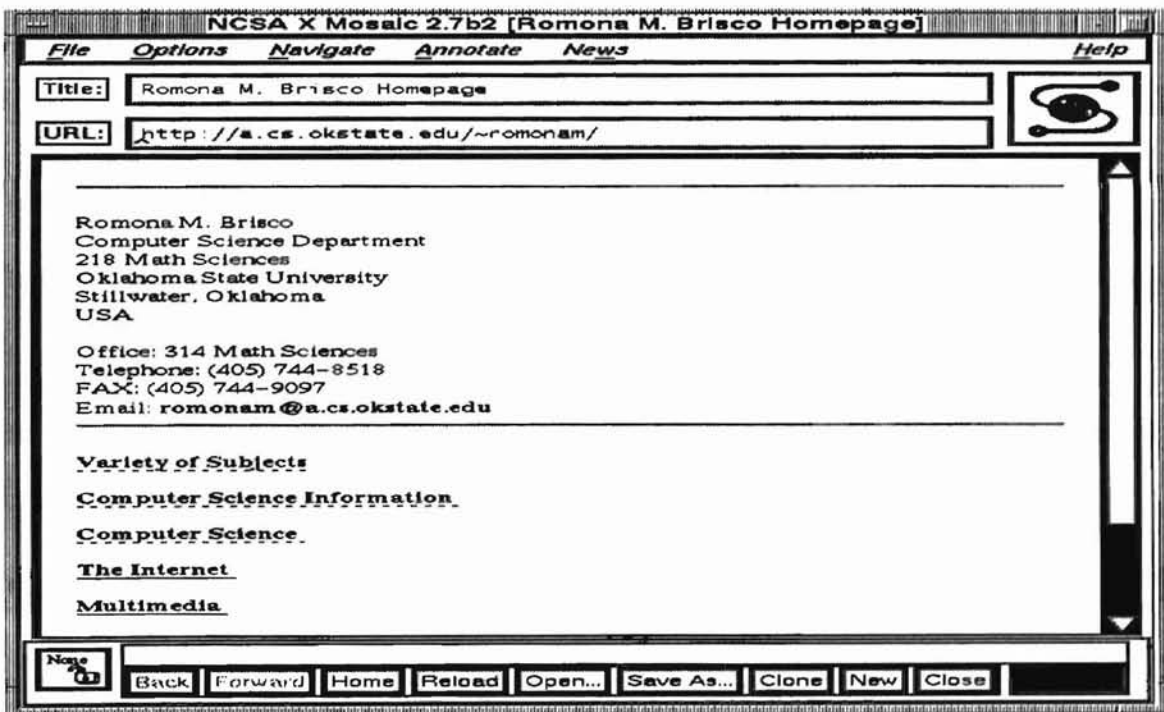


Figure 16. Result of accessing the WWW in MWI.



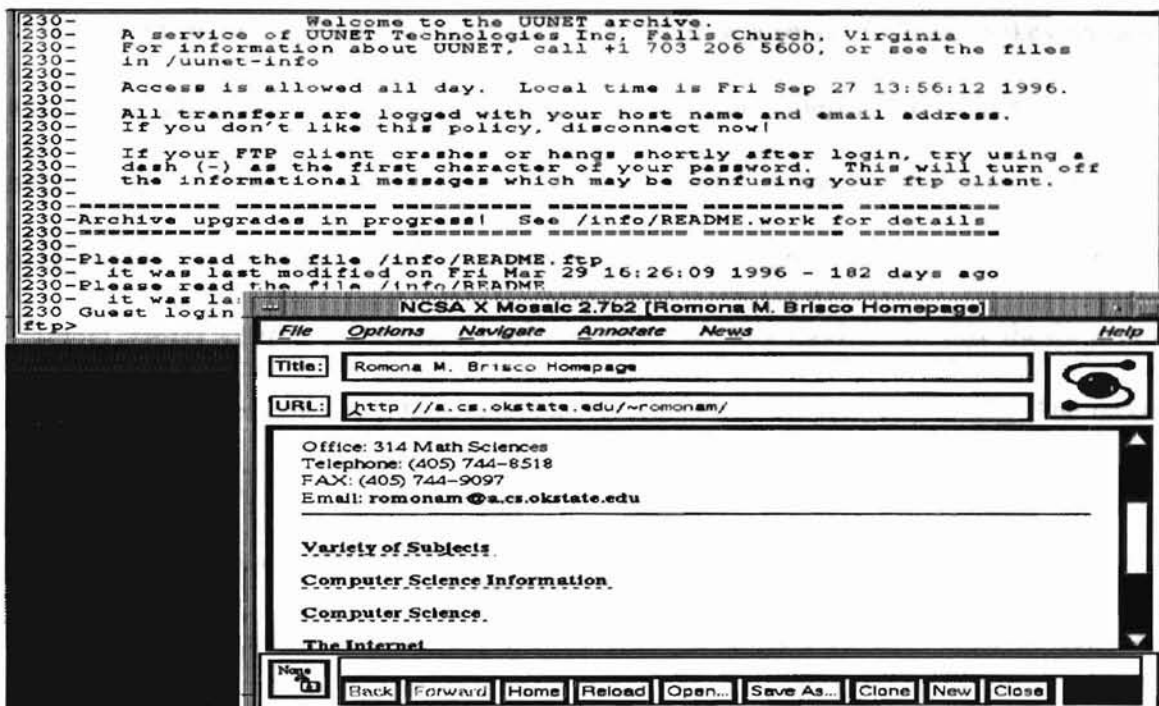


Figure 17. The result of accessing WWW and a Window.

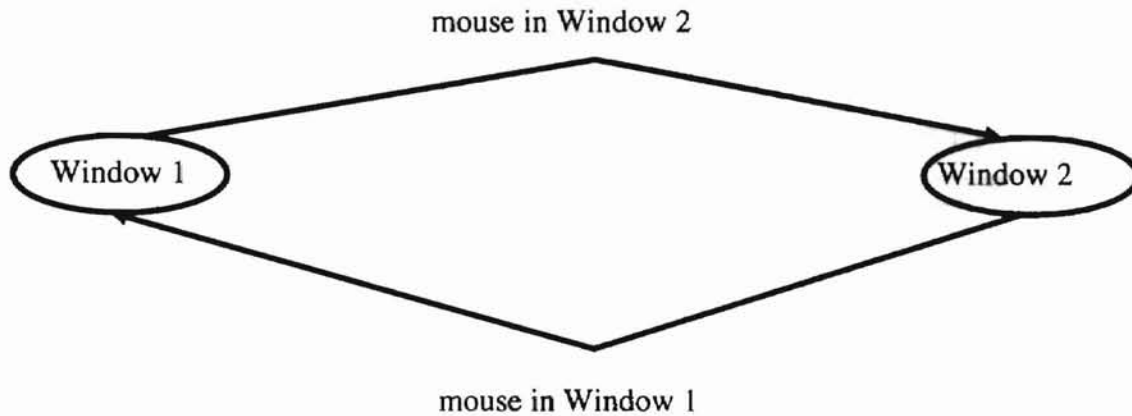
This example shows how multiple actions are performed simultaneously in MWI. The “Window” is another important part of the interface because like an Xterm, a user can type commands in this window and also see the displayed results. In order to access this window the user will have to click on the button “Window” from MWI. The other buttons and menus provided in this interface are for providing help and entertainment to the user. They include answers to questions about buttons, menus, saving files, welcome messages and entertainment. Together, these elements achieve the design objectives of the MWI.

#### 4.2.2 The Model

There are several computing models for user interaction used in software development. Currently, a very popular example is the client-server model [12]. In the

client-server model, software is organized as a client or a server. Client makes requests and server performs the tasks. Clients and servers are independent objects using the request-reply communication scheme. Another model is the event-driven computing model [10]. In this case objects respond to events. Graphics packages such as SRGP (Simple Raster Graphics Package) [10] are based on this model. In this thesis a new model is developed and used. It is called an active automaton model (AAM). The model is described below:

In AAM, software is organized as self-contained objects. All objects are capable of executing concurrently. A property named focus is associated with each object. An active-object is defined as a pair (object, focus). A focus can be on or off. An event is a message sent to an object. An event can change the focus of an object. An active automaton is defined as a state machine where active objects are states and events are input. The transition function defines a new state depending on the current state and the current event. An example is shown in Figure 18. In figure 18, when the mouse is in window 1, its focus is on and the states are (window 1, on) and (window 2, off). When the mouse moves to window 2, the states change to (window 1, off) and (window 2, on). All objects are executing concurrently. An event can effect changes in more than one state. The difference between the event-driven model and the AAM is that in the event-driven model applications wait for an event to occur, and in AAM all applications run concurrently.



**Figure 18. An Active Automaton.**

#### 4.2.3 The Design

The user interface designed in this thesis, the MWI, is designed to be an active automation. In Figure 19, MWI is represented as an active automaton. The automaton in this figure contains the states Rlogin, FTP, WWW and Telnet. The arcs are the transitions from one state to another. This figure also shows that the arcs are bi-directional between states. This means that any of the internet tools can be accessed simultaneously with each other. The highest level implementation structure of the MWI follows the underlying model. The implementation was accomplished by an event loop. Figure 20 shows the event loop at the abstract level.

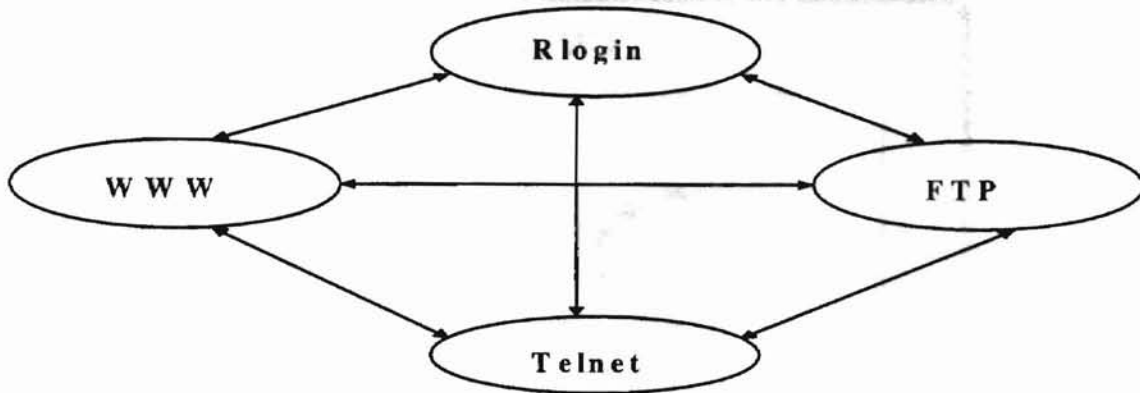


Figure 19. MWI as an Active Automaton.

*Event Loop:*

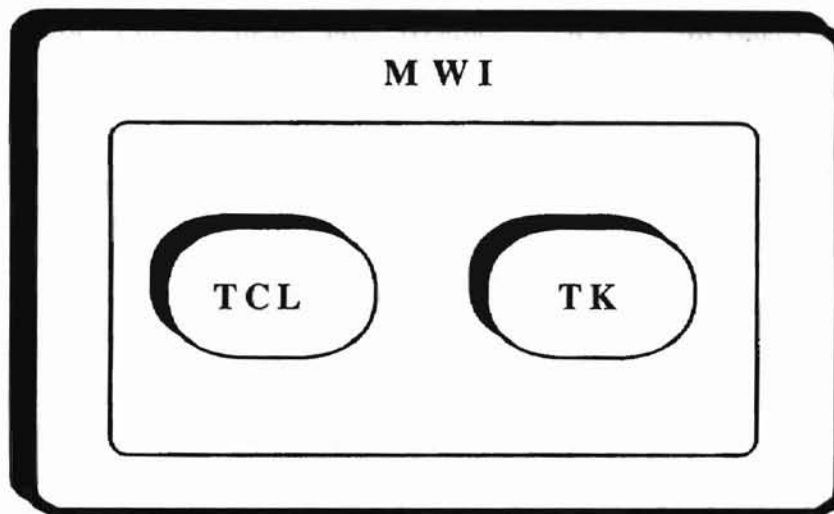
```

While(not quit) {
  switch(event) {
    WWW
      /* Execute World Wide Web Interface */
    FTP
      /* Start File Transfer Protocol Process */
    Rlogin
      /* Start Remote Login Process */
    Telnet
      /* Start a Telnet Process*/
    . . .
  }
}

```

Figure 20. Event Loop.

The software architecture of MWI consists of the development tools Tcl and Tk discussed in Chapter 2. The source code of MWI is based on the Tcl and Tk. Figure 21 illustrates the software architecture design in terms of the relationship of MWI to other software.



**Figure 21. Software Architecture.**

The organization of the components of MWI correspond to the software components. The components can be organized as a tree. The tree structure is represented in Figure 22. All the nodes in this figure make up the MWI. First, the root node in Figure 22 is "MWF" because it is the "root" to all other objects in MWI. Second, the "HELP" node provides users with pertinent information about objects in MWI. Third, the "Internet Tools" node which is the main focus of this interface, accesses the internet tools provided by MWI simultaneously. Lastly, the "Entertainment" node provides miscellaneous entertainment to the user. This ensures that the user may have "fun", or will not become "bored", while waiting for one's information or resources. The child nodes of these parent nodes are as follows: 1) The child nodes for "HELP" are "Getting Started", "Application", "Windows", "Buttons", and "Miscellaneous", 2) The child nodes for "Internet Tools" are "Web", "FTP", "Rlogin", "Telnet", and "Miscellaneous" and 3) The

child nodes for "Entertainment" are "Weather", "News", "Pictures", "Messages", and "Miscellaneous". This tree contains all the elements in the MWI.

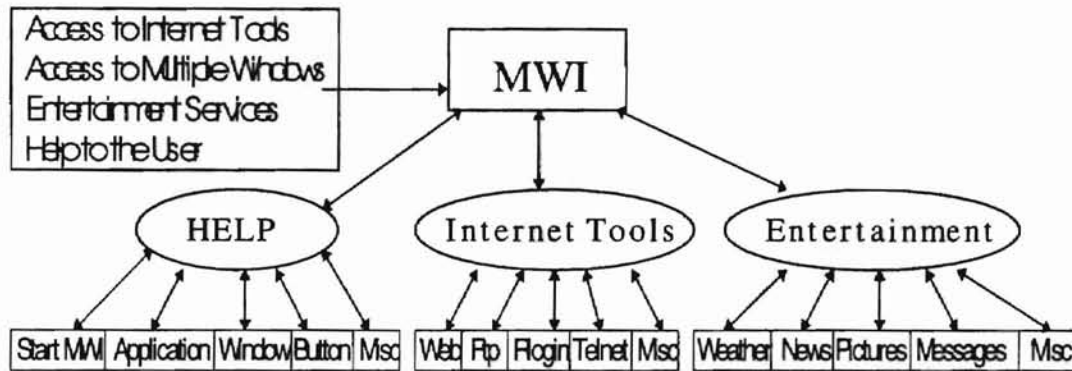


Figure 22. Tree Structure of MWI.

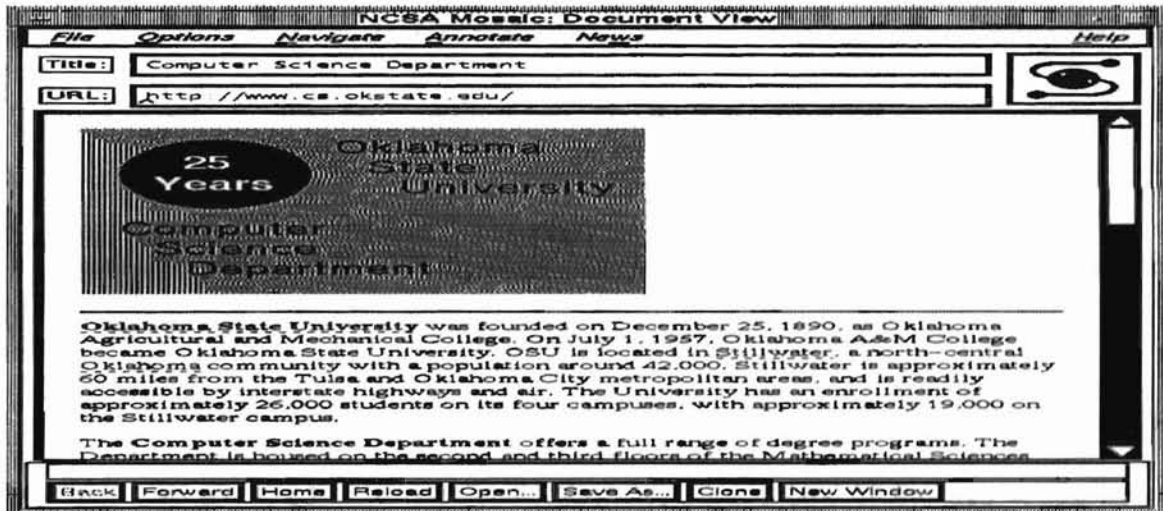


Figure 23. Local University WWW Interface.

## CHAPTER 5

### CONCLUSION

There are applications that appear similar to the Multiple Window Interface for Internet Tools [23][24]. However, MWI differs from them in several ways. Most applications are single purpose applications with the capability to browse the web [23]. However, MWI can browse the web and at the same time explore other internet tools. In other words, MWI provides the same capability of single purpose interfaces and also allows access to any other internet tools simultaneously. For example, in using the web browser mosaic, if a user types in the command "mosaic", the interface that appears is the home page of WWW of a local university, or the homepage a user. An example is shown in Figure 23. MWI is a unified interface that parallelizes the tasks of accessing internet tools. This is possible since the implementation platform is a multiprocessor machine. Therefore, MWI has incorporated a WWW interface that begins with a home page that provides a set of links that are the most commonly encountered on the web as illustrated in Figure 16. By implementing this feature, MWI is more efficient, convenient, and faster than NCSA mosaic or any other web browsers discussed in this thesis. The list of links used in MWI were chosen based on the high traffic encountered at these sites. Some links are by subjects and others are not. By clicking on the links provided by MWI, the

user can get to his/her point of interest on the net faster because it avoids various links between the starting point and the final destination. Another reason why the links provided by MWI are more efficient compared to other browsers is that while the user is waiting for his/her information, he/she can explore the interface by clicking on different widgets, menus and icons to find out other interesting information or entertain themselves. Mosaic and Netscape can provide the same information or resources as MWI by clicking on links. However, using Mosaic and Netscape involves browsing various links before one gets to their final destination point of resources. In other words, one of the major differences between MWI and other web browsers is concurrency. Table I provides a comparison of MWI with other browsers. The criteria for comparison and the browsers to compare are taken from [23].

Factor	MWI	InterAp	Netscape Navigator	Quarterdeck Mosaic	NCSA Mosaic
1. Navigation	****	***	***	***	***
2. Bookmarks	**½	****	***½	***½	**½
3. Customization	****	***½	***½	***½	***
4. Downloading	****	****	***½	***½	***
5. Concurrency	*****	*	*	*	*

\*\*\*\*\*Excellent    \*\*\*\*Good    \*\*\*Acceptable    \*\*Poor    \*Unacceptable

Refer to PC Computing[23] pp.161 for further explanation of Ratings.

Table I. MWI vs. Other Web Browsers.



MWI has proven to be a better choice over the GUIs discussed in this thesis. In conducting this research MWI has also proven to be a user-friendly interface and more efficient than most GUIs used in the public domain.

Future research work may include enhancement of MWI to provide access to all internet tools (gopher, archie and news). However, it is portable to systems which do not support X-Windows, XF, Tcl, Tk, Expect and Wish. Another direction for future work is to investigate methods to use Sequent's parallel programming library with the objective of improving performance.

## BIBLIOGRAPHY

- [1] Aho, Alfred V. and Ullman, Jeffrey D., Foundations of Computer Science, W. H. Freeman and Company, 1992.
- [2] Berghel, H., The Inevitable Demise of the Web, Applied Computing Review, Volume 3 Number 2, ACM Press, Fall 1995.
- [3] Bherat, Krishna and Brown, Marc H., Visual Oblique: A System for Building Distributed, Multi-User Applications by Direct Manipulation, SRC-Technical Report #130a, Digital Systems Research Center, Palo Alto, CA, October 1995.
- [4] Bourne, John R., Object-Oriented Engineering Building Engineering Systems Using Smalltalk-80, Richard D. Irwin and Asken Associates, Inc., 1992.
- [5] Bouvier, Dennis J., Versions of Standards of HTML, Applied Computing Review, Volume 3 Number 2, ACM Press, Fall 1995.
- [6] Brisco, Romona M., A Multiple Windows Interface for Internet Tools, Proceedings of the ISCA (International Society for Computers and their Applications) 11th International Conference, Computers and Their Applications, San Francisco, CA, March 1996.
- [7] Delmas, Sven, XF Design and Implementation of a Programming Environment for Interactive Construction of Graphical User Interfaces, Technical University of Berlin, MS Thesis, 1993.
- [8] Feinman, Todd, Internet Chameleon, PC Magazine, October 10, 1995, Volume 14 Number 17, pp. 178-182, 187-188.
- [9] Flanagan, David, Internet access tools: Motif Tools Streamlined GUI Design and Programming with the Xmt Library, O'Reilly & Associates, Inc., 1994, pp. 3-4.
- [10] Foley, J., Dam, Van A., Feiner, S. and Hughes, J., Computer Graphics: Principles and Practices, Second Edition, Addison-Wesley Publishing Company, Reading, MA, 1996.

- [11] Grainger, Brian E., GUI Drag-and-Drop Tools Ease OI Database and Device Interface Definition, *Instrumentation & Control Systems* 1995, pp. 53-55.
- [12] International DCE Workshop, DCE--the OSF distributed computing environment: client/server model and beyond, Karlsruhe, Germany, October 1993.
- [13] Johnson, Jeff A., Nardi, Bonnie A., Zarmer, Craig L., and Miller, James R., ACE: Building Interactive Graphical Applications, *Communications of the ACM*, April 1993, Volume 36 Number 4, pp. 41-55.
- [14] Kummetha, V.C.S. Reddy, A level-linked R\* tree structure with an application using X-Window graphical interface, Oklahoma State University, MS Thesis, 1993.
- [15] Kwan, Thomas T., McGrath, Robert E. and Reed, Daniel A., NCSA's World Wide Web Server: Design and Performance, *Computer*, November 1995, Volume 28 Number 11, pp. 68-74.
- [16] Laudon, Kenneth C. and Laudon, Jane P., *Essentials of Management Information Systems*, Prentice Hall, Inc., 1995.
- [17] Leavens, Alex, *Designing GUI Applications for Windows*, M&T Books, 1994, A Division of MIS: Press, Inc., A Subsidiary of Henry Holt and Company, Inc..
- [18] Lewis, Peter H., Best Web Browsers, *PC World*, June 1995, Volume 13 Number 6, pp. 137.
- [19] Libes, Don, *Exploring Expect*, O' Reilly & Associates, Inc., 1995.
- [20] Marcus, Aaron, Smilonich, Nick and Thompson, Lynne, *The Cross-GUI Handbook for Multiplatform User Interface Design*, Addison-Wesley Publishing Company, 1995.
- [21] Mayhew, Deborah J., *Principles and Guidelines in Software User Interface Design*, Prentice Hall, Inc., 1992.
- [22] McMinds, Donald L., *Writing your own OSF/Motif Widgets*, Prentice Hall Inc., 1995.
- [23] Meyerson, Adam, The Ultimate Web Browser, *PC Computing Magazine*, September 1995, Volume 8 Number 9, pp. 152-162.
- [24] Mullet, Kevin and Sano, Darrell, *Designing Visual Interfaces*, Prentice Hall, Inc., 1995.
- [25] Notess, Greg R., Comparing Commercial WWW Browsers, *Online*, May 1995, Volume 19 Number 3, pp. 43-49.

- [26] Notess, Greg R., Learning to FTP, Online, March 1994, Volume 18 Number 2, pp. 79-82.
- [27] Notess, Greg R., Telnet Explored, Online, January 1994, Volume 18 Number 1, pp. 94-96.
- [28] Osterhaug, Anita, Guide to Parallel Programming On Sequent Computer Systems, Second Edition, Sequent Computer Systems, Inc., 1989.
- [29] Ousterhout, John, The Tcl and Tk Toolkit, Addison-Wesley Publishing Company, 1994.
- [30] Pasicznyuk, Robert and Zumalt, Joe, Four Internet Browsers--A review., Journal of Academic Librarianship, March 1996, pp. 163-164.
- [31] Pfaffenberger, Bryan, Netscape Navigator Surfing the Web and Exploring the Internet, Academic Press, Inc., 1995.
- [32] Resnick, Rosalind, Graphical Interfaces for the Internet, PC Novice, January 1995, Volume 6 Number 1, pp. 76-79.
- [33] Rodriguez, Karen, CompuServe buys SPRY for Internet Links, InfoWorld, March 20, 1995, Volume 17 Number 12, pp. 14.
- [34] Rose, Phillip F. H., The Macintosh Finder Pure GUI, PC Magazine, September 12, 1989, Volume 8 Number 15, pp. 133-134.
- [35] Rudolf, Jim and Waite, Cathy, Completing the Job of Interface Design, IEEE Software, November 1992, Volume 9, pp. 11-32.
- [36] Sayles, Johnathan S., Karlen, Steve, Molchan, Peter, and Bildoeau, Gary, GUI-Based Design and Development for Client/Server Applications, John Wiley & Sons Publishing Company, 1994.
- [37] Schulzrinne, Henning, World Wide Web: Whence, Whither, What Next?, IEEE Network, March 1996, Volume 10 Number 2, pp. 10-17.
- [38] Seymour, Jim, The GUI An Interface You Won't Outgrow, PC Magazine, September 12, 1989, Volume 8 Number 15, pp. 97-109.
- [39] Sodhi, Man Mohan S., An OR/MS Guide to the Internet, Interfaces, November-December 1995, Volume 25 Number 6, pp. 14-29.
- [40] Stevens, W. Richard, UNIX Network Programming, Prentice Hall, Inc., 1990.

[41] Tenopir, Carol, The User-System Interface, Library Journal, August 1989, Volume 114 Number 13, pp. 80-81.

[42] Weiss, Mark Allen, Data Structures and Algorithm Analysis in C, The Benjamin Cummings Publishing Company, Inc., 1993.

[43] Yang, Cui-Qing and Ali, Mahir, S., Xlib by Example, X Version II Release 5, Academic Press, Inc., 1994.

[44] A short history of Internet Protocols at CERN,  
<http://wwwcn.cern.ch/pdp/ns/ben/TCPHIST.html>.

[45] CERN-European Laboratory for Particle Physics,  
<http://www.cern.ch/CERN/GeneralInfo.html>.

[46] SPRY Corporate Information, <http://www.sprynet.com/about/corpinfo/index.html>.

[47] wish-Simple windowing shell, <http://xpi.com/tix/doc/tcltkman/wish.html>.

**APPENDIX I**  
**IMPLEMENTATION (SOURCE CODE)**

```
#!/contrib/bin/wish -f

set auto_path "$tk_library/demos $auto_path"

wm title . "Internet TOOLS"

#-----

# The code below creates the main window, consisting of a
# menu bar and a message explaining the basic operations
# of the program.

#-----

frame .menu -relief raised -borderwidth 1

message .msg -font -Adobe-times-medium-r-normal--*-180* -relief raised -width 500 \
-borderwidth 1 -text "The menus above contains one or more subjects to choose from. To
invoke one of the subjects, press the left mouse button on the menu item, drag the mouse
to the desired subject in the menu, then release the mouse button. Click on OK when
finished with each window. When you are done with this window, go to the menu Misc
to quit. "

pack .menu -side top -fill x
```

```

pack .msg -side bottom -expand yes -fill both

menubutton .menu.menu -text "Ftp" -menu .menu.menu.m \
    -underline 0

menu .menu.menu.m

.menu.menu.m add command -label "uu.net" \
    -command FTP1 \
    -underline 0

set id 0

proc FTP1 {} {
    global id

    set id 1

    if {$id == 1} {
        puts "FTP id = 1"

        test

    }

    # exec ftp ftp.uu.net &
}

.menu.menu.m add command -label "cs.berkeley.edu" \
    -command FTP2 \
    -underline 0

proc FTP2 {} {

```

```

exec ftp ftp.cs.berkeley.edu &
}

.menu.menu.m add command -label "a.cs.okstate.edu" \
    -command FTP3 \
    -underline 0
proc FTP3 {} {
    exec ftp ftp.a.cs.okstate.edu &
}

.menu.menu.m add command -label "wuarchive.wustl.edu" \
    -command FTP4 \
    -underline 0
proc FTP4 {} {
    exec ftp wuarchive.wustl.edu &
}

.menu.menu.m add command -label "aud.alcatel.com" \
    -command FTP5 \
    -underline 0
proc FTP5 {} {
    exec ftp ftp.aud.alcatel.com &
}

.menu.menu.m add command -label "sunsite.edu" \
    -command FTP6 \

```



-underline 0

```
proc FTP6 () {
```

```
    exec ftp ftp.sunsite.edu &
```

```
}
```

```
menubutton .menu.text -text "Explore the WEB" -menu .menu.text.m -underline 0
```

```
menu .menu.text.m
```

```
.menu.text.m add command -label "World Wide Web..." -command Web1 \
```

```
    -underline 0
```

```
proc Web1 {{w .bindings}} {
```

```
    catch {destroy $w}
```

```
    toplevel $w
```

```
    dpos $w
```

```
    wm title $w "Links to the World Wide Web"
```

```
    wm iconname $w "Text Bindings"
```

```
    button $w.ok -text OK -command "destroy $w"
```

```
    text $w.t -relief raised -bd 2 -yscrollcommand "$w.s set" -setgrid true \
```

```
        -width 60 -height 28 \
```

```
        -font "-Adobe-Helvetica-Bold-R-Normal-*-120-*
```

```
    scrollbar $w.s -relief flat -command "$w.t yview"
```

```
pack $w.ok -side bottom -fill x
```

```
pack $w.s -side right -fill y
```

```
pack $w.t -expand yes -fill both
```

```
# Set up display styles
```

```
if {[tk colormodel $w] == "color" } {
```

```
    set bold "-foreground red"
```

```
    set normal "-foreground {}"
```

```
} else {
```

```
    set bold "-foreground white -background black"
```

```
    set normal "-foreground {} -background {}"
```

```
}
```

```
$w.t insert 0.0 {\
```

To get to a web page, move the mouse over a URL address.

Once the URL address is highlighted, press mouse button 3.

```
}
```

```
insertWithTags $w.t \
```

```
{1. Connect to Web.} d1
```

```
# insertWithTags $w.t \n\n
```

```
# insertWithTags $w.t \
```

```
foreach tag {d1 d2 d3} {
```

```
    $w.t tag bind $tag <Any-Enter> "$w.t tag configure $tag $bold"
```

```

    $w.t tag bind $tag <Any-Leave> "$w.t tag configure $tag $normal"
}

$w.t tag bind d1 <3> web1

proc web1 {{w .bindings}} {
    global id
    catch {destroy $w}
    toplevel $w
    dpos $w
    wm title $w "Internet Tools"
    wm iconname $w "WWW"
    button $w.ok -text OK -command "destroy $w"
    text $w.t -relief raised -bd 2 -yscrollcommand "$w.s set" -setgrid true \
        -width 60 -height 28 \
        -font "-Adobe-Helvetica-Bold-R-Normal-*-120-*"
    scrollbar $w.s -relief flat -command "$w.t yview"
    pack $w.ok -side bottom -fill x
    pack $w.s -side right -fill y
    pack $w.t -expand yes -fill both

    # Set up display styles
    if {[tk colormodel $w] == "color"} {
        set bold "-foreground red"
    }
}

```

```

    set normal "-foreground {}"
} else {
    set bold "-foreground white -background black"
    set normal "-foreground {} -background {}"
}
if {$id == 0} {
    puts "WWW id = 0"
    test
}
$w.t mark set insert 0.0
bind $w <Any-Enter> "focus $w.t"
}
# exec xmosaic -home http://a.cs.okstate.edu/~romonam/ &
# $w.t tag bind d2 <3> mkPlot
# $w.t tag bind d3 <3> mkCanvText
# $w.t tag bind d4 <3> mkArrow
# $w.t tag bind d5 <3> mkRuler
# $w.t tag bind d6 <3> mkScroll
# END OF PROCEDURE web1
}
# Procedure: test
proc test {} {

```

global id

```
switch $id \
```

```
0 { puts "Exec WWW" } \
```

```
1 { puts "Exec FTP" } \
```

```
2 { puts "Exec RLogin" } \
```

```
3 { puts "Exec Telnet" } \
```

```
default { puts "ERROR" } \
```

```
}
```

# The procedure below inserts text into a given text widget and

# applies one or more tags to that text. The arguments are:

#

# w        Window in which to insert

# text     Text to insert (it's inserted at the "insert" mark)

# args     One or more tags to apply to text. If this is empty

#         then all tags are removed from the text.

```
proc insertWithTags { w text args } {
```

```
  set start [$w index insert]
```

```
  $w insert insert $text
```

```
  foreach tag [$w tag names $start] {
```

```
    $w tag remove $tag $start insert
```

```
  }
```

```

foreach i $args {
    $w tag add $i $start insert
}
}

menubutton .menu.scroll -text "Rlogin" -menu .menu.scroll.m \
    -underline 0
menu .menu.scroll.m
.menu.scroll.m add command -label "Hosts..." -command funListBox2 -underline 0
proc funListBox2 {{w .l2}} {
    catch {destroy $w}
    toplevel $w
    dpos $w
    wm title $w "Hosts"
    wm iconname $w "Listbox"
    wm minsize $w 1 1

    message $w.msg -font -Adobe-times-medium-r-normal--*-180* -aspect 300 \
        -text "A listbox containing several hosts is displayed below, along with a
scrollbar. You can scan the list either using the scrollbar or by dragging in the listbox
window with button 2 pressed. If you double-click button 1 on a host, then you have
selected that host. Click the \"OK\" button to bo connected."

```

```

frame $w.frame -borderwidth 10

button $w.ok -text OK -command "destroy $w"

pack $w.msg -side top

pack $w.ok -side bottom -fill x

pack $w.frame -side top -expand yes -fill y

scrollbar $w.frame.scroll -relief sunken -command "$w.frame.list yview"

listbox $w.frame.list -yscroll "$w.frame.scroll set" -relief sunken \
    -geometry 20x20 -setgrid 1

pack $w.frame.list $w.frame.scroll -side left -fill y

$w.frame.list insert 0 a.cs.okstate.edu

bind $w.frame.list <Double-1> {exec rlogin a.cs.okstate.edu}

#\

# "$w config -bg \[lindex \[selection get\] 0\]
# $w.frame config -bg \[lindex \[selection get\] 0\]
# $w.msg config -bg \[lindex \[selection get\] 0\]"
}

menubutton .menu.misc -text Misc -menu .menu.misc.m -underline 0

menu .menu.misc.m

.menu.misc.m add command -label "Bitmaps" -command mkBitmaps \
    -underline 0

.menu.misc.m add command -label "Quit" -command "destroy ." -underline 0

```

```
pack .menu.menu .menu.text .menu.scroll .menu.misc -side left

# Set up for keyboard-based menu traversal

bind . <Any-FocusIn> {
    if {"%d" == "NotifyVirtual" && ("%m" == "NotifyNormal")} {
        focus .menu
    }
}

tk_menuBar .menu .menu.menu .menu.text .menu.scroll .menu.misc

# Position a dialog box at a reasonable place on the screen.

proc dpos w {
    wm geometry $w +300+300}
```



## APPENDIX II

### GLOSSARY [1, 4, 7, 10, 14, 19, 29, 40, 42, 46]

**API** - Applications Programming Interface, a set of functions that a user process can call.

**Dialog box** - type of window that presents choices to the user and provides a graphical means to input information to an application.

**Dialog controls** - dialog box controls are user interface components that appear primarily in dialog boxes.

**Expect** - a program that is used to control interactive programs. A communicating script that interacts with interactive programs.

**Finite automaton** - a graph-based way of specifying patterns.

**Menus** - provides an easy way to use visual interface that allows the user to browse and select an item from a list of choices or commands that the application provides rather than having to recall the commands, options, or data from memory.

**Multiprocessor** - computer that incorporate multiple identical processors (CPUs) and a single common memory.

**Nodes** - a set of points usually represented by circles.

**OOP** - Object Oriented Programming, a methodology that provides description of objects, including functional, behavioral and declarative function.

**Protocol** - a set of rules and conventions between the communicating participants.

**Shared Memory** - two or more processes share a memory segment.

**SRGP** - Simple Raster Graphics Package, a device independent graphics package that exploits raster capabilities.

**Tcl** - Tool Command Language, a scripting language for controlling and extending applications.

**TCP/IP** - Transmission Control Protocol, a connection-oriented protocol that provides a reliable full-duplex, byte system for a user process. Internet Protocol, the protocol that provides the packet delivery service for TCP, UDP, and ICMP.

**Tk** - toolkit for the X-Window System that allows a user to create GUIs by writing Tcl scripts.

**Widgets** - a window with a particular appearance and behavior. Widgets are divided into classes such as buttons, menus, and scrollbars.

**Window(s)** - a window is an area within the screen ( or on the desktop) with which a user conducts a dialog with a computer system.

**WISH** - Windowing Shell that supports the development and execution of graphical user interfaces.

**WYSIWYG** - What You See Is What You Get, an acronym for what you see is what you get. It generally refers to the degree of one-to-one correspondence between information displayed on the screen and information displayed on printed output or stored in files.

**XF** - an integrated programming environment that supports the development of GUIs. An interface used to build/create other interfaces.

**X-lib** - X library, the lowest level of programming interface to X. A programming interface that has subroutine package written in C and is provided by the X-Window system.

**X-Windows** - interface that runs under UNIX environment.

VITA

ROMONA M. BRISCO

Candidate for the Degree of

Master of Science

Thesis: A Multiple-Windows Interface for Internet Tools

Major Field: Computer Science

Biographical:

Personal Data: Born in Chicago, Illinois, to Diane and Alfred Brisco.

Education: Graduated from Martin Luther King Jr. High School, Chicago, Illinois, June 1989; received Bachelor of Science degree in Computer Science from Langston University, Langston, Oklahoma, in July 1993; completed requirements for the Master of Science degree at Oklahoma State University, Stillwater, Oklahoma, in December 1996.

Experience: Graduate Research Assistant and Teaching Assistant at Oklahoma State University, Stillwater, Oklahoma; Software Developer at BELLCORE, Piscataway, New Jersey; GUI developer and Assistant System Administrator at Argonne National Laboratory, Argonne, Illinois.

Professional Organizations: ACM Student Chapter (Oklahoma State University); Delta Sigma Theta Sorority Incorporated.