

***DEVELOPMENT OF AN INTERACTIVE COMPUTER-BASED  
MULTIMEDIA DESIGN MANUAL***

Utah Department of Transportation  
Report UT-95.02

Mountain-Plains Consortium  
Report MPC 95-41

by

William J. Grenney  
Brent C. Robinson  
Thad E. Senti

Utah State University  
Logan, Utah

June 1995

### **Acknowledgments**

This has been a cooperative project with the Utah Department of Transportation and the Mountain-Plains Consortium, University Transportation Centers Program of the U.S. Department of Transportation. We would like to thank these agencies for their aid and support.

### **Disclaimer**

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

## CONTENTS

	<i>Page</i>
LIST OF TABLES .....	iii
LIST OF FIGURES .....	v
EXECUTIVE SUMMARY .....	vii
INTRODUCTION AND OBJECTIVES .....	1
GLOSSARY .....	5
REVIEW OF LITERATURE .....	9
Decision Support Systems for Water Resource Applications .....	9
Decision Support Systems with Multimedia .....	12
FEATURES .....	15
Hypertext and Hypergraphics .....	15
Browser .....	17
Popup Windows .....	19
Button Bars and Pulldown Menus .....	20
Search .....	21
History List .....	22
Bookmarks .....	22
Annotation .....	22
External Program Links .....	23
HyperCalc .....	23
Glossary .....	25
Multimedia .....	25
Audio .....	26
Graphics .....	27
Movie Clips .....	29
Equation Solvers .....	30
Decision Support System .....	33
DEVELOPMENT TOOLS .....	35
Hypertext Authoring Systems .....	35
Hypertext authoring tool evaluation .....	36
Viewer authoring process .....	39
Browser .....	40
Decision Support System .....	40
Knowledge encapsulation .....	42
Inference engine .....	43

Object-oriented programming .....	44
Flexpert implementation .....	46
C Programming Language .....	55
Equation Solvers .....	55
HyperCalc .....	56
Integration of Applications .....	56
Hypertext module .....	57
Logic control module .....	57
Numerical module .....	58
AUTHORING PROCESS .....	59
User Needs Analysis .....	59
Definition of Application and Design Standards .....	61
Application definition .....	61
Design standards .....	61
Selection of Development Tools .....	62
Knowledge Acquisition and Representation .....	64
Prototype Development .....	67
Debugging Process .....	67
Updating of Manuals .....	68
EVALUATIONS .....	71
User Evaluation .....	71
Author Evaluation .....	78
SUMMARY AND CONCLUSIONS .....	83
REFERENCES .....	87
APPENDICES .....	91
Appendix A. Equations Implemented in Equation Solver .....	93
Appendix B. Culvert Design Decision Support System Rule File .....	99
Appendix C. Culvert Design Decision Support System Variable File .....	111
Appendix D. Culvert Design Decision Support System Activity File .....	121
Appendix E. Equation Solver C Code .....	127
Appendix F. Computer-based Manual Evaluation Form .....	155
Appendix G. Computer-based Manual Evaluation Results .....	161

**LIST OF TABLES**

<i>Table</i>		<i>Page</i>
1	Microsoft Multimedia Viewer 2.0 and Folio Views 3.0 Evaluation Summary . . . . .	36
2	Structural Outline and Terminology Used for the Development of Computer-Based Manuals . . . . .	65
3	User Importance Ratings of the Computer-Based Manual Features . . . . .	73
4	User Comparison of Computer-Based Manual to Printed Version . . . . .	75
5	Advantages and Disadvantages of the Computer-Based Manual Perceived by the Evaluators . . . . .	77



## LIST OF FIGURES

<i>Figure</i>	<i>Page</i>
1 Examples of (A) Hypertext and a (B) Hypergraphic .....	16
2 Browser Interface .....	18
3 Popup Window Example .....	20
4 HyperCalc Interface .....	24
5 Audio Control Device .....	27
6 Movie Playback Device .....	30
7 Equation Solver Interface .....	31
8 Inheritance Example .....	46
9 Flexpert Application Process .....	47
10 "Rule Guru" .....	48
11 "Variable Guru" .....	49
12 "Activity Guru" .....	50
13 Flexpert Object Templates Menu .....	54
14 Hypertext, Logic, and Numerical Modules Interaction .....	57
15 Computer-Based Manual Authoring Process Flow Chart .....	60
16 Debugging Process .....	68
17 Users Evaluation of Computer-Based Manuals Feature .....	73
18 Users Evaluation of Computer-Based Manuals Feature (continued) .....	74
19 User Comparison of Computer-Based and Printed Manuals, Average Reply and Confidence Intervals are Shown .....	76

20	Author Evaluation of Computer-Based Manuals Feature . . . . .	78
21	Author Evaluation of Computer-Based Manuals Features (continued) . . . . .	79
22	Author Comparison of Computer-Based and Printed Manuals . . . . .	80



## EXECUTIVE SUMMARY

Manuals of instruction are important engineering tools because they provide the information necessary to perform important design procedures and calculations correctly and accurately. Effective manuals can result in savings of both time and money by outlining approved design procedures as well as familiarizing designers with the important policies and procedures of their departments. Since accepted policies and procedures frequently change, manuals must be updated continually. This process can be very expensive and time consuming.

There is a growing realization that technical reference materials have greater impact if presented in a computer-based format. Computer-based manuals allow the author to convey information through the use of text, graphics, movies, and audio, as well as use the intelligent capabilities of decision support systems, and other programs. Computer-based manuals help users access information more quickly than conventional hard copy manuals. The former also aids the user in the design process and greatly simplify the process of updating and making corrections or changes to manuals.

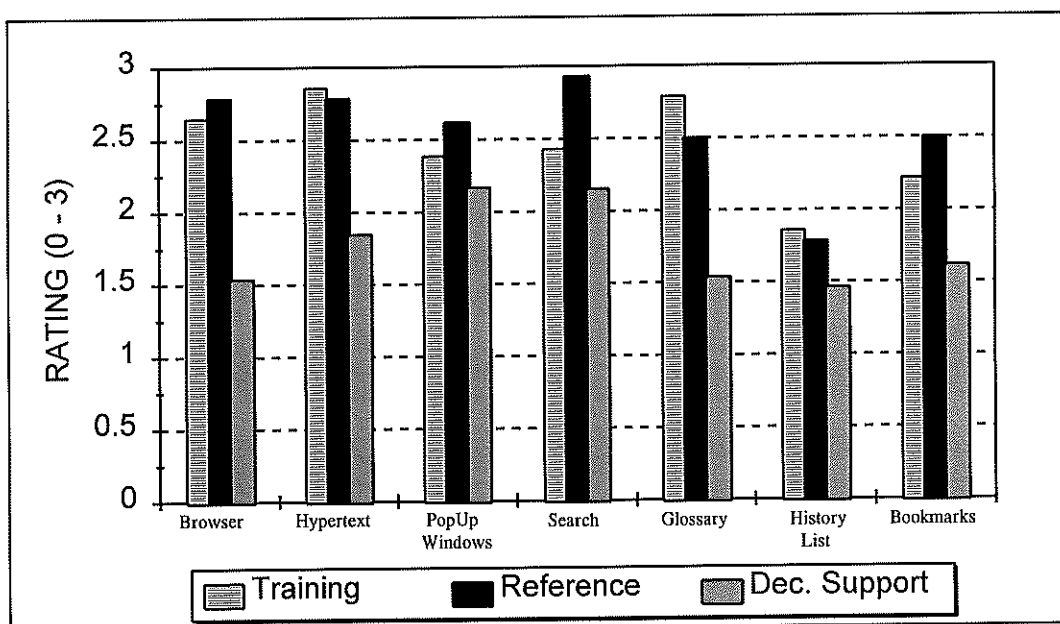
The objective of this project was to develop a computer-based design and training manual by implementing tools and techniques normally used for the development of rule-based decision support systems. Chapter 9 of the American Association of State Highway and Transportation Officials (AASHTO) Model Drainage Manual entitled "Culverts," and the United States Geological Survey Water-Resources Investigations Report 83-4129 entitled "Methods for Estimating Peak Discharge and Flood Boundaries of Streams in Utah" were used as a

prototypes. The final application is a single computer-based manual, referred to as the "Drainage Library", that is implemented on IBM compatible microcomputers operating under Microsoft Windows 3.1. The following specific tasks were accomplished during this project:

- 1) Two separate decision support systems were authored. The first guides novice users through a step-by-step process for the design of a culvert. The second helps the user select appropriate equations to determine flood depth and peak discharge values for a given location within the state of Utah.
- 2) Commercial hypertext authoring tools were analyzed to determine which tool best met desired needs for organizing the contents of the manual. Microsoft Multimedia Viewer was chosen and subsequently used to create the hypertext of the computer-based manual.
- 3) The manual was directly linked with other computer programs, allowing the user to access these programs from within the drainage manual application. The Drainage Library was linked with the decision support systems, Federal Highway Administration drainage software, and commercial software products such as Word Perfect, Quattro Pro, and Auto Cad.
- 4) Multimedia enhancements were implemented into the manual. Pictures, video, and audio were added to the text of the manual to increase the amount of information presented and to clarify concepts.
- 5) Equation solvers were developed which allow necessary calculations to be performed quickly and accurately on the computer while using the manual. Individual equation solvers were implemented for each design equation found within the manual. Equation solvers were also used to replace nomographs and charts, as well as to provide conversions between English and metric units.
- 6) A table of contents (browser) was organized with hypertext jumps to provide easy access of desired information within the manual.
- 7) Search capabilities were added to allow the user to find desired topics or words and quickly access related information.

- 8) A unit conversion package, HyperCalc, was developed and linked with the Drainage Library to allow a user to quickly perform unit conversions.
- 9) An easy access glossary was organized to provide an expanded definition of terms. Popup windows were also incorporated within the main body of the manual to explain or define unclear concepts and terms.
- 10) Tables and graphs were displayed quickly and concisely.

Fourteen professionals from the Utah Department of Transportation participated in an evaluation of the Drainage Library software. They were asked to rank seven attributes (browser, hypertext, popup windows, search, glossary, history list, and bookmarks) on a scale of 0 (not important) to 3 (very important) for each of three types of applications: training, reference, and decision support. The results of the evaluation are shown in the following figure.



Users' evaluation of computer-based manuals features (0 = not important, 1 = little importance, 2 = important, 3 = very important).

It was generally perceived that movie clips, pictures, and audio were most effective for training applications. Hypertext, popup windows, searches, bookmarks, and annotation were most useful for reference. Equation solvers, external program links, and decision support systems were ranked highest in the area of decision making.

The evaluators were also asked to list what they thought were the major advantages and disadvantages of the computer-based manual. The results are shown in the following table:

Advantages and disadvantages of the computer-based manual perceived by the evaluators

Advantages	Disadvantages
Easy to update	Cost of development
Offers interesting learning environment	Cost of required hardware
Potential to achieve quick solutions	Required computer knowledge
Better use of resources	Learning curve
Faster for obtaining information	Accessibility - can carry a book with you, but won't always have a computer
Multimedia useful for explaining concepts	
Easy to find desired data	
Search capabilities	
Flexibility	
Manual, equations, and software packages can all be bound together	

The evaluators generally agreed that the computer-based manuals let users access information much more quickly than conventional manuals, and that the computerized manual offered more flexibility and would be easier to update and maintain. The major disadvantage was the cost for development and hardware. The evaluators overwhelmingly judged the overall effectiveness of the computer-based manual much higher than a printed manual for the same information.



## INTRODUCTION AND OBJECTIVES

Manuals of instruction are important engineering tools because they provide the necessary information to perform important design procedures and calculations correctly and accurately. Effective manuals can result in savings of both time and money by outlining approved design procedures as well as familiarizing designers with the important policies and procedures of their departments. Since accepted policies and procedures frequently change, manuals must be updated continually. This process can be very expensive and time consuming.

There is a growing realization that technical reference materials can have greater impact if presented in a computer-based format. Computer-based manuals allow the author to convey information through the use of text, graphics, movies, and audio, as well as utilize the intelligent capabilities of decision support systems, and other programs. Computer-based manuals help users access more information more quickly than conventional hard copy manuals allow. They also aid the user in the design process, and greatly simplify the process of updating and making corrections or changes to manuals.

The objective of this project was to develop a computer-based manual of instruction by implementing tools and techniques normally used for the development of rule-based decision support systems. Chapter 9 of the American Association of State Highway and Transportation Officials (AASHTO) Model Drainage Manual titled "Culverts" and the United States Geological Survey Water-Resources Investigations Report 83-4129 titled "Methods for Estimating Peak Discharge and Flood Boundaries of Streams in Utah" were used as a prototype. The final

application is a single computer-based manual, referred to as the Drainage Library, implemented on IBM compatible microcomputers operating under Microsoft Windows. The following specific tasks were accomplished:

- 1) Two separate decision support systems were authored. The first guides novice users through a step-by-step process for the design of a culvert. The second helps the user select appropriate equations to determine flood depth and peak discharge values for a given location within the state of Utah.
- 2) Commercial hypertext authoring tools were analyzed to determine which tool best met desired needs for organizing the contents of the manual. Microsoft Multimedia Viewer was chosen and subsequently used to create the hypertext of the computer-based manual.
- 3) The manual was directly linked with other computer programs, allowing the user to access these programs from within the drainage manual application. The Drainage Library was linked with the decision support systems, Federal Highway Administration drainage software, and commercial software products such as Word Perfect, Quattro Pro, and Auto Cad.
- 4) Multimedia enhancements were implemented into the manual. Pictures, video, and audio were added to the text of the manual to increase the amount of information presented and clarify concepts.
- 5) Equation solvers were developed which allow necessary calculations to be performed quickly and accurately on the computer while using the manual. Individual equation solvers were implemented for each design equation found within the manual. Equation solvers were also used to replace nomographs and charts, as well as provide conversions between English and metric units.
- 6) A table of contents (browser) was organized with hypertext jumps to provide easy access of desired information within the manual.
- 7) Search capabilities were added to allow the user to find desired topics or words and quickly access related information.



- 8) A unit conversion package, HyperCalc, was developed and linked with the Drainage Library to allow a user to effortlessly perform unit conversions.
- 9) An easy access glossary was organized which can be used to define unknown terms. Popup windows were also incorporated within the main body of the manual to explain or define unclear concepts and terms.
- 10) Tables and graphs were displayed quickly and concisely.

This report is divided into seven sections which explain the process of developing a computer-based manual of instruction. The first section, Glossary, defines concepts and terms which appear in the remainder of the document. The Review of Literature provides an overview of decision support systems applied to water resource and multimedia applications. No publications were found that combine both of these elements, so they were addressed individually. The Features section describes the various features and enhancements incorporated into the final application. The Development Tools section describes the evaluation and selection of software tools needed to implement the desired features. The Authoring Process section describes how the information within the manual was organized and how the various features were integrated into the application using commercial authoring tools and rule-based decision support system technology. The Evaluations section presents the results of evaluations performed by fourteen professionals at the Utah Department of Transportation and the authors of the Drainage Library. The Conclusions section presents the final conclusions of this project.



## GLOSSARY

This section of this report was written to define terms and concepts with which a reader might not be familiar. Many of these terms are also explained in more detail within the other sections of this document.

Author	The author is the individual who creates and organizes a computer-based manual.
Bitmap	A bitmap is a commonly used format for drawings, scanned photographs, or artwork in a windows environment. Bitmaps are made up of sets of individual bits that contain information about the color and intensity of each pixel contained in an image and have a *.BMP filename extension.
Browser	The browser is an expandable table of contents that allows the user to access information found within the computer-based manual.
Button Bar	The button bar consists of a row of buttons displayed at the top of each main topic of a document.
C (C++)	C and C++ are computer programming languages. They were used to create the browser and equation solvers as well as perform complex mathematical calculations within the DSSs found in the Drainage Library.
Decision Support System (DSS)	DSSs are programs which query a user for the necessary information to make a decision and present the user with possible solutions.
Drainage Library	The Drainage Library is the computer-based manual that includes:

- 1) Chapter 9, "Culverts," of the American Association of State Highway and Transportation Officials (AASHTO) Model Drainage Manual
- 2) The United State Geological Survey Water-Resources Investigation Report 83-4129 titled "Methods for Estimating Peak Discharge and Flood Boundaries of Streams in Utah."

#### Equation Solvers

Equation solvers are programmed functions which allow the user to perform calculations and solve design equations within the computer-based manual. The user can enter values for desired variables and the computer performs the calculations and displays the results.

#### Flexpert

Flexpert is the decision support system (DSS) authoring tool used to create the DSSs contained within the Drainage Manual.

#### Folio Views

Folio Views is a hypertext authoring tool.

#### Hot Spot

Hot spots are regions (text or a graphic) that when activated perform a particular action associated with the hot spot. Hot spots activation occurs when a user places the mouse pointer over the hot spot and clicks the mouse button. Hot spots are typically used to perform a jump, display a popup window, or run a command.

#### HyperCalc

HyperCalc is an unit conversion utility which is accessed from within the Drainage Manual.

#### Hypergraphic

Hypergraphics are images which have been defined by the author to be used as hot spots.

#### Hyperjump

A hyperjump takes the user from one point within a computer-based document to another predefined point.

#### Hypertext

Hypertext is text which the author has defined to be used as hot spots. Within the Drainage Library,

hypertext is a light blue color to distinguish it from the other text of the manual.

Jump	A hyperjump takes the user from one point within a computer-based document to another predefined point.
Microsoft Multimedia Viewer	The hypertext authoring tool which was used to create the Drainage Library.
Multimedia	Multimedia is the combination of text, graphics, audio or video segments within a single document on a computer.
Popup Windows	Popup windows are simplified windows that appear on top of the main window within a hypertext document. Also known as popups.
Pulldown Menus	Pulldown menus are located directly above the button bar and appear in the form of text. By placing the cursor over the pulldown menu text, a list of commands appears for the user to choose from.
Scrollbar	Scrollbars are located on the far right side of main windows. Scrollbars allow users to move up and down within a topic by dragging the slider box with the mouse or clicking on the arrows.
Topic	Hypertext documents created using Microsoft Multimedia Viewer are made up of topics. Topics could be compared to individual pages of a book and consist of paragraphs containing graphics and text.
User	The individual who actually uses the computer-based manual is called the user.



## REVIEW OF LITERATURE

### Decision Support Systems for Water Resource Applications

An extensive literature review was performed by Grenney, Wallace, and Senti (1993) on the use of decision support systems for water resource applications. According to Grenney, very little research has been performed on this topic, but a brief summary of the relevant literature follows.

Water management personnel are seeking new ways to do a better job of managing and operating complex water systems using new computer technology (Labadie and Sullivan, 1986). Decision support systems (DSS) are interactive computer tools that play an important role in accomplishing this objective.

Decision support systems can be used to incorporate both data and models to help in the problem-solving process. They can also be used to aid senior management who must make important decisions, but are not necessarily technical experts in the field of the problem. The DSS is designed to assist the decision maker or user by presenting him with alternative solutions to problems rather than providing him with a single answer. The decision maker can then use decision-making skills to choose the best possible alternative. DSS must be highly interactive, visual programs which help the user advance towards a solution in a stepwise manner, defining the problem and analyzing the impacts and trade-offs of possible alternatives (Grenney, Wallace, and Senti, 1993).

There are numerous ways that DSSs have been implemented; however, according to Johnson, the architecture of most of them contains three general types of subsystems: (1) data base subsystems, (2) model subsystems, and (3) dialog management subsystems (Johnson, 1986). Data base subsystems are used to store factual data and can incorporate powerful data management and visualization functionality like those of geographic information systems (GIS). The model subsystem can be both procedural and declarative. Procedural models are typically numerical algorithms for analyzing data and predicting results based on statistics or deterministic relationships. Declarative subsystems attempt to represent heuristic knowledge about a specific problem domain by representing the domain by its premises and the relationships that link those premises. These relationships and premises are usually defined as rules (Geselbracht and Johnston, 1988). The dialog management subsystems are graphical user interfaces (GUI) that allow the user to enter and receive data and information from the computer.

Different DSS applications require different emphasis within these three subsystems. In some cases emphasis is on the use of simulation models. For example, the Tennessee Valley Authority (TVA) depends on a collection of compatible simulation models to analyze water-related programs for decision support purposes, including data collection, hydrodynamics of rivers and reservoirs, water quality of rivers and reservoirs, surface water hydrology, groundwater transport, transport of toxic material adhering to sediment, reservoir system operation, and



hydropower planning (Brown and Shelton, 1986). Another DSS was developed to aid in the management of the Tejo estuary in Portugal. This system was based on a chain of data bases and simulation models for dispersion, nonpoint pollution, and regional water-quality (Camara et al., 1990). Grenney (1992) developed a computer implementation of the water cost-allocation process for six regions of the Nile River Basin in Egypt. In order to maintain compatibility with existing software, he developed 45 linked spreadsheets that provided a data base of the physical and economical attributes for the regions. Procedural algorithms operate on the data to produce output files for graphical displays in a spreadsheet.

Complex simulation models are being enhanced by the application of rule-based pre- and post-processors. For example, software to expedite calibration of the widely used storm water management model (SWMM) relies heavily on rule-based subsystems to assist the user in selecting parameter values for the main simulation model (Baffaut and Delleur, 1989; Liong, Chan, and Lum, 1991). Simulation results are compared to observed values and useful adjustments in the values of significant parameters are provided.

Several investigators have taken steps to couple policy guidelines with simulation models in a DSS. Davis et al. (1991) described a prototype DSS that estimates the effects of potential land-use and land-management policies on the costs and quality of water supplied to the city of Adelaide, Australia. The DSS contains modules that allow the user to create a collection of policies in a rule-like formal

syntax. A watershed simulation model is also integrated into the system for the analyses of hydrologic data. Koch and Allen (1986) introduced a DSS to integrate data management with hydrologic models for use in local water management in areas where irrigation is the primary water use. Their program recognizes the priority of water rights throughout the operation along with need based on soil moisture levels.

### Decision Support Systems with Multimedia

Vanegas and Baker (1994) defined multimedia as the use of the computer as an integrator, combining audio and visual media along with text and data into a single digital document that can be directly accessed by the user through the computer. What sets multimedia technology apart from television and other technologies is the ability it provides the user to interact with the system (Narasimhalu, 1994). Multimedia applications are defined by the way the user interacts with the application as either passive or interactive. Passive multimedia sequentially presents the material to the user. Interactive multimedia consists of a nonlinear environment where users can navigate through the subject matter according to their own needs and pace using hypertext.

According to Narasimhalu (1994), multimedia technology can be incorporated into DSS to provide a means of visualizing interaction between the user and the DSS. However, he stated that the integration process is still in its infancy and will require considerable study before robust operational systems can be created which utilizes

multimedia's full capabilities. Maybury (1994) stated that decision support systems benefit from multimedia technology because it helps make user interaction a natural process. He explained that images are often used to identify or describe particular objects or events. Sequences of images can be used to explain situations or actions typically found in operation or design manuals. Video offers a continuous, dynamic medium that can also be used to effectively communicate information. However, when using multimedia technologies, appropriate media must be selected to effectively present the information and care must be taken to ensure consistency across multiple media. By integrating multimedia technologies into DSSs, information can be presented in the most efficient and effective manner, helping the user understand and interact with the system.



## FEATURES

Within this section, the features are described which were incorporated into the computer-based manual. Figures have been added to illustrate how each feature appears within the manual. The tools and procedures used to create each of these features will be discussed within the Authoring Process section of this report. As the project was completed, several UDOT professionals evaluated each feature. The Evaluations section will disclose how important they considered each of these features were for training, reference, and decision support.

### Hypertext and Hypergraphics

Hypertext and hypergraphics are “hot spots” that allow a user to maneuver within a document and access information quickly and efficiently. When a hot spot is selected (by position the cursor over the hot spot and clicking the left mouse button), predefined actions are performed. These actions are defined by the author and include moving the user from one point within a document to another predefined point, displaying a popup window, or executing a desired command (Microsoft Corporation, 1993).

Hypertext is a type of hot spot that can be placed anywhere within a manual and appears in the same form as standard text. Hypertext is generally distinguished from other text by its color. Microsoft Multimedia Viewer automatically creates the hypertext and sets the color to green. However, the author can change the color of the

hypertext used within the document. A light blue color was used for all hypertext within the Drainage Library. Hypertext activation occurs when the user positions the cursor over the hypertext and clicks the left mouse button. When hypertext is activated, the action or command associated with the hypertext is automatically performed. Hypertext is typically used to navigate the user through the document by moving the user to the position that the hypertext is linked to, known as the destination topic. Microsoft Multimedia Viewer allows the author to create hypertext very easily and quickly. An example of hypertext is shown in Figure 1(A).

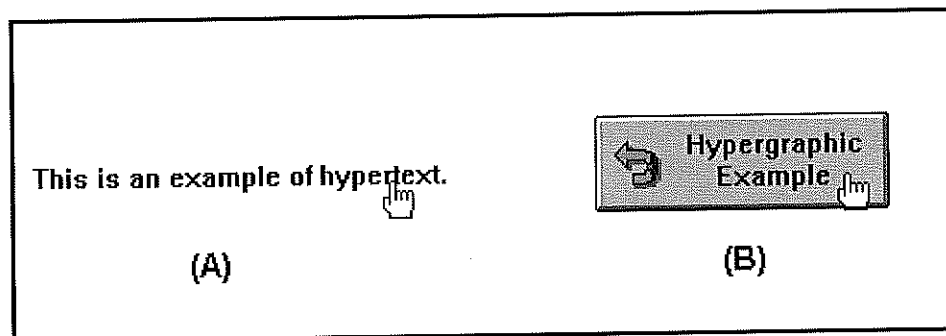


Figure 1. Examples of (A) hypertext and a (B) hypergraphic.


Hypergraphics operates in much the same manner hypertext does. Bitmaps (graphics) can be utilized as hot spots within a document to perform desired actions. An entire bitmap can be used as a hot spot. This is particularly useful for the creation of buttons, because commands can be associated with the button. Multiple hot spots can also be placed on a single bitmap. This feature is useful when defining different parts or regions found within a graphic. For example, a map of the different regions

within a state could be displayed as a bitmap. When users select a given region on the map, they could jump directly to a topic that gives desired information about the selected region.

As previously mentioned, hot spots are utilized by positioning the cursor over the hot spot and clicking the mouse button. When the cursor is located over a hot spot, the cursor changes to a small pointing hand as depicted in Figure 1. Figure 1(B) shows an example of a hypergraphic.

### Browser

The browser is an expandable table of contents that makes it possible for a user to hyperjump directly to a desired location within a computerized document. The browser, shown in Figure 2, contains headings for the topics of the manual arranged in a format similar to the table of contents of a book. However, the levels of information within the browser can be expanded and collapsed by the user to display as much or as little information as desired. When the manual is first accessed, the browser is automatically opened, and the user can choose the location he wishes to enter the manual. The browser can also be accessed from anywhere within the manual by the clicking the "Browser" button located on the button bar.

Any heading within the browser that has a small book () next to it can be expanded or collapsed by clicking the mouse button over the book or the text associated with the book. The headings can also be expanded and collapsed using the

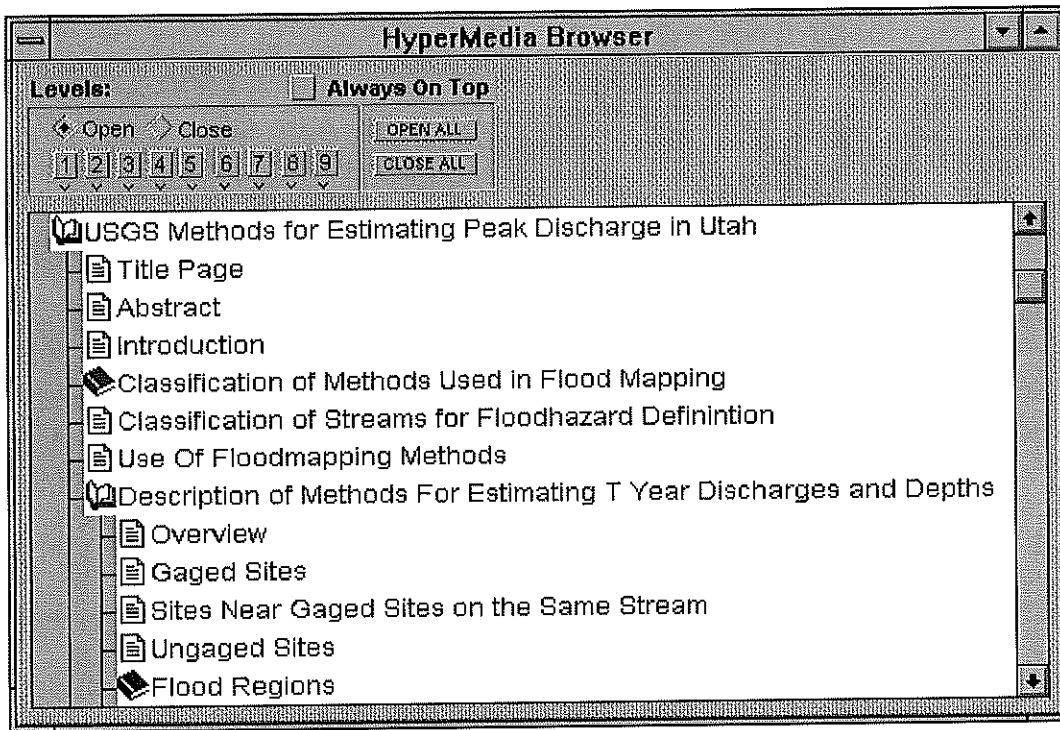


Figure 2. Browser interface.

“Levels” buttons located at the top of the browser window. By selecting a Levels button (0 through 9), the browser will automatically be opened or closed to the number of levels specified on the chosen button. The “close all” and “open all” buttons can also be used to manipulated the number of levels displayed by the browser. When the topic headings within the browser that contain small white pages (☐) next to them are chosen, the user is taken directly to that topic found within the manual. These jumps can only be performed from the headings that contain this white page next to them. The “always open” check box found at the top of the browser button can be selected to keep



the browser open and on top of all other open windows after a jump to the manual is performed. This can be particularly useful for navigation within the manual.

### Popup Windows

Popup windows are simplified windows displayed on top of the main topic window and are typically referred to simply as popups. Popups can be activated from hypertext or hypergraphics and are placed directly on top of the main topic window from which it was activated. Figure 3 shows an example of a popup window titled "Flood depth." The hypertext that was used to activate the popup window is also shown in the figure. Popups can be closed by pressing the escape key or clicking the mouse button outside the popup region (Microsoft Corporation, 1993). Text, graphics, movie clips, and multimedia controllers can be placed within popup windows. Hypertext and hypergraphics can also be activated from within these windows. As can be seen in Figure 3, popup windows do not have title bars, button bars, pulldown menus, or any of the other controls which windows normally contain. They also do not contain scroll bars. The amount of information that can be displayed in an individual popup is therefore limited to the size of the screen.

Popup windows are useful for displaying information that reinforces or clarifies information contained within the main topic. For example, a popup window can be used to define unknown words or terms found within the main topic of a document.

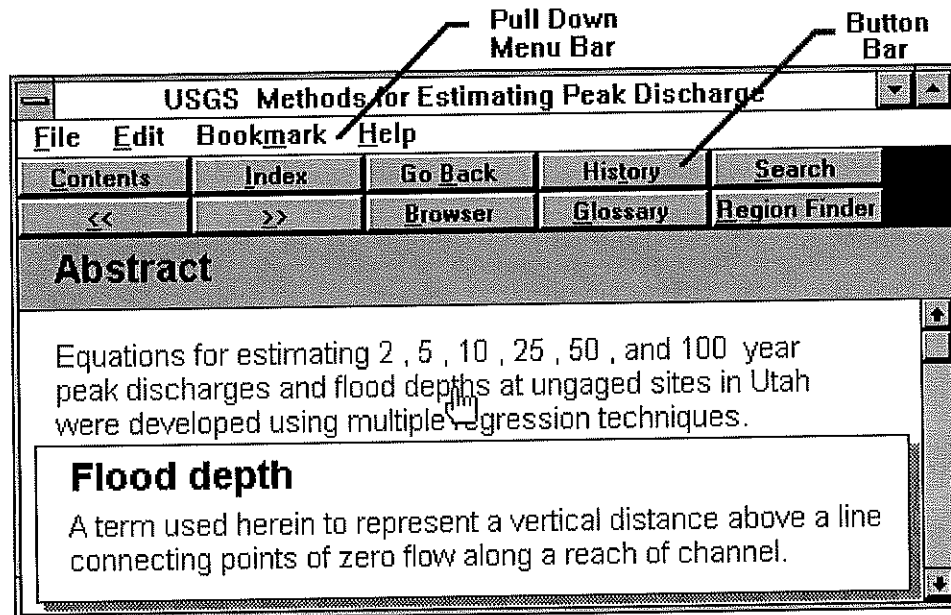


Figure 3. Popup window example.

### Button Bars and Pulldown Menus

Buttons and menus provide shortcuts for commonly used commands. The button bar and pulldown menu bars are shown in Figure 3.

The button bar consists of a row of buttons displayed at the top of each main topic of a document. Commands are assigned to each button on the button bar, so that when the user chooses a button, the command is executed. Button bars can be customized by the author to include commands that would be most useful for a given document.

Pulldown menus are located directly above the button bar and appear in the form of text. By placing the cursor over the pulldown menu text, a list of several

commands appears from which the user can choose. The items contained within the menu bar can also be customized by the author.

### Search

Multimedia Viewer contains full-text search capabilities that help the user quickly find desired information within a document. Searches can be performed within a document by selecting the “Search” button from the button bar. When this button is chosen, a search dialog box is opened, and the user is able to enter what he wishes to search for. Searches can be performed for words, phrases, numbers, or even special characters. The user can also identify which topic groups or sections he would like the searched to be performed within. For example, a computerized document might contain several chapters. However, the user might wish to search for a word within only one of the chapters. By selecting that chapter’s topic group, the search would be limited to only that chapter of the document. The author of the document can define these topic groups as he sees most beneficial and useful.

After a user defines what he would like to search for, the search results are displayed. These results display the number of topics that contain the item being search for and the title of each of these topics. The user can jump directly to any one of the topics where the item was found by clicking on the topic title in the search display window. Each occurrence of the item searched for is also highlighted within the topic.

### History List

The history list is a sequential list of the topics that a user has accessed within the manual. This list can be viewed by selecting the “History” button from the button bar. Any of the topics on this list can be directly jumped to by double clicking the topic title. This feature can be useful for accessing information previously found within the document and navigating through the document.

### Bookmarks

Bookmarks can be used to mark and find locations within a computerized document that are frequently referenced. Users can define their own bookmarks by selecting the pulldown menu titled “Bookmark.” Up to six bookmarks can be created within each document. Each bookmark can be given its own title so it can be quickly identified. Bookmarks can be accessed from anywhere within a document. When a bookmark is chosen, a jump is performed to the position previously defined within the document. Bookmarks are saved in a personalized file when a user exits a document but can be deleted by the user at any time.

### Annotation

Users can add personalized notes to any main topic using the annotation command. This is done by selecting “Annotate” from the “Edit” pulldown menu. The desired note or text can then be entered and saved. A small paper-clip icon is displayed

in the margin of the topic within which the note was created. Notes can be reaccessed by double clicking the mouse while the cursor is located over the paper-clip icon. Notes can also be changed and/or deleted. When the user exits the document, notes are saved in a \*.ANN file on the user's computer that can be reaccessed when the document is used again.

### External Program Links

The ability to run other applications and programs while using a computerized manual greatly enhances the usefulness of the manual. Program links can be created by the author of a computer-based manual that allow the user to run programs from within the manual. DOS and Windows-based programs were both accessed from within the Drainage Library. These programs were activated using buttons on the button bar, pulldown menus, and hypergraphics.

### HyperCalc

HyperCalc is an application that was developed to allow a user to quickly and accurately perform unit conversions. HyperCalc can be run directly from within the computerized manual by selecting the "HyperCalc" button located on the button bar. The version of HyperCalc (beta) linked to the manual contains general unit conversions as well as some commonly used drainage and highway design equations and conversions. The HyperCalc interface is shown in Figure 4. Check boxes located at

the top of the page allow the user to select which of the three types of unit conversions he wishes to perform. HyperCalc will convert values between units of the same system (English to English and metric to metric), as well as between units of the English and metric systems (metric to English and English to metric).

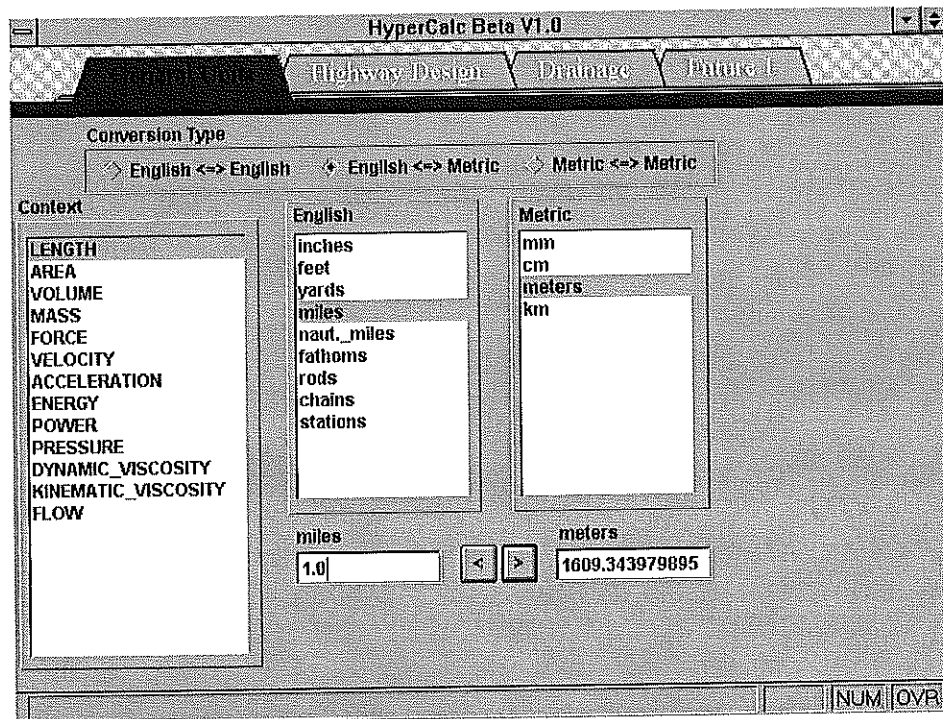


Figure 4. HyperCalc interface.

The user can choose what type of measurements he wishes to use from the context menu. He can then choose the units that he wishes to convert between from the next two menus. Two fields labeled with these units will then appear at the bottom of the page. By entering values in these fields, the user can convert between the respective units.

### Glossary

A glossary of terms found within the AASHTO Drainage Manual was created for the computerized document. This glossary can be quickly accessed using the “Glossary” button located on the button bar or the browser. Each of the definitions that are contained within the glossary is arranged in alphabetical order in the form of hypertext. When this hypertext is activated, the meaning or definition of the selected term is displayed in a popup window.

Where terms that were defined in the glossary occur within the main body of the manual, they were replaced with hypertext. This hypertext, when activated, displays a popup window with the definition of the given term.

### Multimedia

Graphics, audio, and movie clips can be used to enhance text that is contained within a document. These multimedia effects can be used to increase the informational content and clarify concepts contained within a document. Multimedia elements can be displayed and run within a Viewer document because Viewer supports the use of Media Control Interface (MCI) commands. These commands provide Viewer with the capability to control audio and video playback devices. MCI commands can be pasted anywhere within a topic, even within popup windows, using Viewers Topic Editor. Like all Viewer commands, MCI commands can be activated by the following situations (Microsoft Corporation, 1993):

- 1) When a topic is displayed or entered;
- 2) When a hot spot is chosen;
- 3) When a menu item or button is selected.

MCI commands can also be activated using the standard playback devices as shown in the figures in the following subsections.

Although multimedia elements can be used to greatly enhance a document, their use typically requires very large amounts of storage. Authors should therefore determine how a document will be distributed and what size constraints will be associated with its distribution to determine how to incorporate multimedia enhancements into a document.

### Audio

Audio can be used to help reinforce concepts and make information more available within a computer-based document. However, audio must be used effectively in order to be useful. Audio clips should be easy to understand, not too loud or soft, and of consistent quality throughout a document. Audio clips were incorporated into the Drainage Library within the glossary and at the startup of each document. These examples can be quickly found and accessed utilizing the "Multimedia" pulldown menu contained in the AASHTO Document.

MCI commands allow an author to customize the way that audio is played within a document. Several examples of different play back methods were implemented into the Drainage Library. For example, audio is played automatically



when the user enters a topic, when a hot spot is selected and from an audio control device within the Drainage Library. The author also has the ability to customize the controller device to fit desired needs. Figure 5 depicts a typical audio controller device. As can be seen in the figure, the playback device contains a play and stop button, as well as a sliding control bar. The sliding control bar can be moved by the user, allowing him to listen to any portion of the audio segment.



Figure 5. Audio control device.

Viewer has the capability to support many different audio formats, but the audio clips added to the Drainage Library were saved as waveform files (\*.wav) (Microsoft Corporation, 1993). These samples were digitized at 11.025 kilohertz (kHz) and took up approximately 1.5 megabytes of storage per minute. The quality of an audio sample depends on the frequency at which it is digitized. Higher quality audio segments can be obtained by digitizing at a higher frequency. However, the higher the audio quality, the more storage space that is required.

### Graphics

Graphics can be used to help clarify unclear concepts and ideas. Graphics can be saved as bitmaps (\*.bmp). Bitmaps are made up of sets of individual bits that contain information about the color and intensity of each pixel contained in an image.

Bitmaps can be obtained using paint software, digitizing slides, scanning in pictures or photographs, or digitizing video frames, or can even be purchased. Images can generally be scanned and or digitized to a desired size. Bitmaps can also be resized within most paint programs; however, sometimes resizing can distort the image. Paint programs are also often used to touch up and crop images. Bitmaps can be pasted directly into Viewer documents using embedded-pane statements.

The number of colors that the intended users' computers will support must be taken into consideration while authoring a system. Sixteen-color bitmaps are typically adequate for simple drawings and cartoons; however, most photographs and natural images require 256-color bitmaps. Viewer supports both 16 and 256-color bitmaps, however, when 256-color bitmaps are displayed on a system that only supports 16 colors, the images appear very distorted and are normally not distinguishable. This problem can be avoided by an author because Viewer has an option which when set determines how many colors a computer is setup for and uses the appropriate bitmap (the author enters two bitmaps, one for 16-color systems and one for 256-color systems). Viewer also provides a command which when set dithers 256-color bitmaps so that they can be viewed on a 16-color system. Dithering is a technique that represents an image using fewer colors than it originally had. This is accomplished by varying the pixel groups using subsets of the colors designed for the bitmap to recreate the effect of the lost colors (Microsoft Corporation, 1994).

The Drainage Library was authored for systems that support 256 colors. However, it contains examples of both 16- and 256-color bitmaps. These bitmaps can be easily located using the "Multimedia" pulldown menu. The size and the number of colors contained in bitmaps are directly related to the memory required to store an image. The majority of the bitmaps were approximately 3 by 5 inches or 180 by 280 pixels and took up approximately 50 kilobytes for 256-color bitmaps.

### Movie clips

Video or movie clips can be obtained from videotapes or using a video camera. The video clips shown in the Drainage Library were taken in Sardine Canyon near Logan, Utah using a home video camera and digitized into Audio Video Interleaved (\*.AVI) form.

Movie clips were incorporated into the glossary of the Drainage Library and can quickly be referenced using the "Multimedia" pulldown menu. Movie clips can be embedded in topics or activated from a hot spot. Several different interfaces and playback configurations can be obtained using the MCI command options. Figure 6 shows a typical movie playback device.

Video clip size is dependent on the quality (number of frames displayed per second), size of the display screen, and length of the clip. The clips contained within the Drainage Library were designed to be viewed on systems that support 256 colors and take up about 5 megabytes for 10 seconds of footage. Video clips typically require

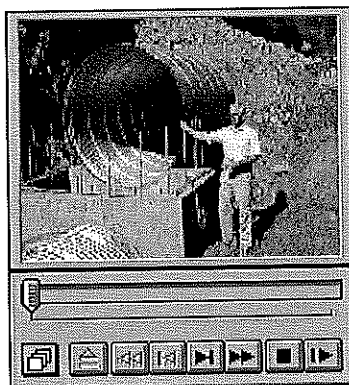


Figure 6. Movie playback device.

19.2 kilobytes per frame displayed plus the memory required for the audio associated with the video.

Special care must be taken when using movies within a document. Like bitmaps, problems will result when movies are viewed on a system that does not support as many colors as the movie was created to display. Differences in the speed of the central processing units (CPU) of computers can also cause problems when a movie is displayed. Movies, like most multimedia features, should be designed and authored for a given system.

### Equation Solvers

Equation solvers were incorporated into the Drainage Library to simplify calculations. Equation solvers can be found within the manual using the browser or the “Eq. Solver” pulldown menu. The equation solvers can be accessed by clicking on the hypergraphic or hypertext containing the equation name. When an equation solver is

accessed, a window is opened that contains the equation, variable definitions, and units and fields where values can be entered and displayed. Figure 7 shows an equation solver used to calculate friction loss within a culvert.

**EQ9-4B**

$$H_f = \frac{29 n^2 L}{R^{1.33}} \left[ \frac{V^2}{2g} \right]$$

\* **Hf = Friction loss (ft)**  
 \* **L = Length of culvert barrel (ft)**  
 \* **R = Hydraulic radius (ft)**  
 \* **V = Velocity (ft/sec)**  
 \* **n = Manning's Roughness Coefficient**

English Metric    Enter    Misc    ? Help    Done

Figure 7. Equation solver interface.

As can be seen in Figure 7, the represented equation is displayed at the top of the equation solver. The variables found in the equation and their respective units are defined below the equation. Values can be entered for each variable in the fields located next to the variable definitions. An unknown can be solved for by leaving its field empty, placing values in all of the other fields, and clicking the “Enter” button. As shown in the figure, the variable definitions contain an asterisk (\*) beside them. This asterisk denotes that the equation solver can be used to solve for that particular variable.

Since all of the variables in the equation solver shown in Figure 7 contain asterisks next to them, any of the variables can be solved for. Some equation solvers contain variables that cannot be solved for. These variables contain exclamation points (!) in place of the asterisk.

The units used within the equation solver can be changed to the English or metric system by selecting the appropriate check box. When this is done, the units displayed with the variable definitions are changed to the appropriate units and all calculations are performed using these units.

When the “Misc” button is selected, copy and paste buttons are displayed which enable the user to copy or paste to and from the clipboard. This allows the user to copy and paste values from the equation solver variable fields as well as to and from other applications (i.e., Quattro Pro, Excel, etc.). Up to five equation solvers can be open at one time, and users can copy and paste from one equation box to another.

Twelve equation solvers were added to the AASHTO Drainage Manual in section 9.5 where complicated design equations existed. Nine drainage equations were implemented in equation solver form into the HyperCalc application. Fifty-nine equation solvers were used to replace the design charts and nomographs found in Appendix D of the same manual. Six equation solvers were implemented into the USGS portion of the manual to aid the user in the calculation of peak discharges and flood depths. Appendix A contains a list of all of these equations.

### Decision Support System

Decision support systems can be linked directly to and executed from a hypertext document. Hypertext documents can also be accessed from a DSS. Accessing a hypertext document from a DSS can be a means of providing a user with necessary background information or definitions required to effectively use the DSS. Two separate decision support systems were incorporated into the Drainage Library.

The first decision support system steps the user through the process of designing a culvert. This program queries the user for the necessary design data and characteristics. It then selects the appropriate equations and relationships and performs the required calculations to determine the feasibility of the entered values. The design process used for this program is outlined in detail in sections 9.6 and 9.7 of the AASHTO Drainage manual. This decision support system can be accessed by clicking the "Culvert DSS" button located on the button bar of the Drainage Manual Document.

The second decision support system aids the user in determining the appropriate equations to use for the estimation of peak discharge and flood depths within Utah. Utah contains several regions which have different hydrological conditions. Regression equations were developed to represent each of these different regions. These equations can be found in The United States Geological Survey Water-Resources Investigations Report 83-4129, titled "Methods for Estimating Peak Discharge in Utah" (Blakemore and Lindskov, 1983). This system can be accessed by

clicking the “Region Finder” button found on the button bar of the USGS “Methods for Estimating Peak Discharge in Utah” computer-based document.



## DEVELOPMENT TOOLS

Powerful commercial authoring systems are available that greatly simplify the process of creating computerized documents. It is important to use an authoring system that fits the desired needs of a given project. Three separate authoring tools were used in the creation of the Drainage Library. These tools included a hypertext authoring tool, decision support system authoring tool, and the C programming language. These tools are evaluated and described in detail within this section.

### Hypertext Authoring Systems

Hypertext authoring systems can be used to organize information (text, graphics, audio, video, etc.) in a manner that is accessible and useful for a user. In order to determine which authoring system best fit our needs, an evaluation of Microsoft Multimedia Viewer 2.0 and Folio Views 3.0 was performed. After examining these products, it was decided that Microsoft Multimedia Viewer best fit our requirements for an authoring system. The results of this evaluation are summarized in the following section. As previously mentioned, Microsoft Multimedia Viewer 2.0 is referred to as Viewer. Folio Views 3.0 will simply be referred to as Folio throughout the remainder of this document.

Hypertext authoring tool evaluation

Both Folio and Viewer contain many features which make them powerful authoring tools and greatly enhance their usability. The features and capabilities of the Folio and Viewer authoring systems are listed and rated in Table 1. In Table 1, the features are rated as "Excellent," "Acceptable," "Poor," or "Not Available." Many of these features were discussed in detail in the Features section of this report. Brief explanations of those features not discussed within the Features section will be given within this section.

Table 1. Microsoft Multimedia Viewer 2.0 and Folio Views 3.0 evaluation summary

Feature	Microsoft Multimedia Viewer 2.0	Folio Views 3.0
Hypertext	Excellent	Excellent
Popup Windows	Excellent	Excellent
Annotation	Excellent	Excellent
Bookmarks	Excellent	Excellent
Highlighters	Not Available	Excellent
History Lists	Excellent	Excellent
Search Capabilities	Excellent	Excellent
External Program Links	Excellent	Excellent
Multimedia Support	Excellent	Excellent
Printing	Acceptable	Excellent
Text Support (Post & Subscripts)	Acceptable	Excellent
Table of Contents (Browser)	Acceptable	Acceptable
Controllability by External Programs	Excellent	Not Available
Custom Button and Menu Bars	Excellent	Poor
Security	Acceptable	Excellent
Updateability	Excellent	Excellent
Royalty Costs	Excellent	Poor
Overall Ease of Use	Acceptable	Excellent

Both Viewer and Folio can be used to create documents that contain hypertext, popup windows, annotation, bookmarks, history lists, search capabilities, external program links, and multimedia enhancements. These features were all rated "Excellent" for both authoring systems.

The "Highlighter," "Printing," "Text Support," and "Table of Contents (Browser)" features were rated higher for Folio than for Viewer. Highlighters are supported under Folio, but are not an option under Viewer. Highlighters can be used to quickly mark text in the hypertext document much as one might mark a textbook. Folio and Viewer both allow the user to print information contained in the hypertext document. However, the print options supported by Folio offer the user more versatility. Under Folio, the user can add headers and footers and specify exactly what he wants to print. On the other hand, Viewer only allows the user to print complete topics and does not support headers or footers. Folio supports the use of superscripts and subscripts within the hypertext document, whereas Viewer does not. In order to display superscripts and subscripts using viewer, the author must either save the characters as bitmaps and paste them into the document or create a custom superscript/subscript font that includes characters that sit above or below the baseline.

The ability to control a hypertext document from an external program greatly increases the usability of an application. Folio 2.0 does not provide the necessary hooks to allow a hypertext document to be externally controlled. On the other hand, Viewer provides these hooks that allow its documents to be controlled from external

applications. The application programming interfaces (APIs) provided by Viewer let an author start or run Viewer applications from within another application or program. These APIs can also specify which topic of a document should be opened when an instance of Viewer is started. This ability to externally control hypertext documents is extremely useful. The browser that was incorporated into this project made use of these APIs to link the browser to the individual topics contained within the document. These APIs can also be useful for accessing and displaying information found within a document from a decision support system (Microsoft Corporation, 1993).

“Custom Button and Menu Bars” was rated “Poor” for Folio and “Excellent” for Viewer. Both Viewer and Folio allow the author to choose from several standard buttons and design a button bar. However, Viewer allows the author the option of creating his own buttons and placing them along with the standard buttons on the button bar. Folio does not allow the author the ability to customize pulldown menus, whereas Viewer does.

Security is an important feature which prohibits individual users from making changes to the hypertext document. It is important to be able to make necessary changes and updates to a hypertext document, but problems could occur if everyone had the ability to make these changes. Both Folio and Viewer provide systems that allow the author of a document to make changes and updates as well as limit those who have the ability to do so.

Viewer does not have any royalty cost associated with its use and distribution, whereas Folio does. The importance of royalty costs is dependent on the uses and expected number of users of the application.

Folio was rated as having a higher "Ease of Use" than Viewer. Folio has a much shorter learning curve than Viewer. The process of authoring jumps, popups, and multimedia enhancements is also much simpler using Folio.

After reviewing the use of Microsoft Multimedia Viewer 2.0 and Folio Views 3.0, it was decided that Viewer best fit the expected needs of this project. The main reason being the desire to access the hypertext document from external locations (the decision support systems). The ability to customize buttons and menus and the related royalty costs also weighed heavily in this decision.

#### Viewer authoring process

The text for a document created using Viewer is stored in topic files. Topic files can be created from text files and word-processed documents by saving them in rich-text format (RTF). This can be accomplished using Microsoft Word or Word Perfect. Once a document has been converted to rich-text format, hypertext and other desirable features can be added. These procedures are outlined in detail in the Viewer Authoring Guide (Microsoft Corporation, 1993).

It was found that the conversion of the data to topic files (RTF) can be performed automatically using RoboHELP, another commercial hypertext authoring tool. RoboHELP utilizes Microsoft Word to create RTF files, but has a much more

user friendly convention for the authoring of hypertext, popups, etc. than Viewer does. RoboHELP does not support many of the features previously mentioned within this report. However, RoboHELP can be used to quickly create RTF files that contain topics, hypertext, and popup windows. These RTF files can be utilized by Viewer to create documents that contain the previously mentioned features and enhancements.

### Browser

As previously mentioned, Viewer does not support or generate a table of contents. The browser utility was therefore developed to provide documents authored with Viewer table of contents functionality. The browser utility was developed at the C-BIT laboratory using the C++ programming language. This utility contains an executable (BROWSER.EXE) that provides hierarchical structure functionality. This structure is based on data that are placed in a data file (HIERLIST.DAT). This data file supplies the executable with the hierarchical outline and the macros that are called from the browser when a selection is made. In this case, the APIs previously mentioned are utilized to open specific topics within Viewer.

### Decision Support System

The term decision support system (DSS) was coined in the early 1970s to represent the use of information systems to support complex decision-making processes (Davis, 1988). Since that time, various interpretations and definitions

have been applied to this term. Turban (1990) cited Scott-Morton (1971) and Keen and Scott-Morton (1978) to emphasize that interactive computer-based systems help decision makers utilize data and models to solve unstructured problems. He continued to define DSSs as follows:

Decision support systems couple the intellectual resources of individuals with the capabilities of the computer to improve the quality of decisions. It is a computer-based support system for management decision makers who deal with semi-structured problems. (Turban, 1990, p. 9)

Although there is not a universally accepted definition for a DSS, it is generally accepted that DSSs are implemented to improve the quality of the information on which a decision is based (Turban, 1990). This is accomplished as DSSs provide a range of alternative solutions for a problem under a given range of circumstances and data. This allows the user to better understand the problem and possible solutions, leading to better informed decisions.

Decision support systems are typically implemented using either a declarative or a procedural approach. The declarative scheme involves the use of nonsequential data that are typically organized using a rule-based approach. Procedural programming involves the use of sequential data that can be implemented through conventional programming. Some applications are more suitable to a rule-based design and others to the procedural design (Grenney, 1992).

### Knowledge encapsulation

As previously mentioned, knowledge is typically represented within an expert system through declarative or procedural schemes. This report will focus on the declarative approach and rule-based systems.

Most commercial DSSs are rule-based and consist of rules, variables, and activities or similarly named terms.

Rules. Rules are conditional statements, typically consisting of two parts, that analyze the state of variables within a rule base. The first part, the antecedent, contains one or more IF clauses. The IF clauses are used to test the values of variables. After the state of a rule, the IF clause, is tested, the second part of the rule is implemented. This portion executes the consequence, THEN and ELSE clauses, of the rule. The THEN and ELSE clauses typically execute the activities of the rule base.

Variables. Variables are objects that can change state. The state of a variable can be assigned and/or tested within a rule base. These values can be previously set or input by the user or another device such as a remote sensor or an external program.

Activities. Activities are actions or consequences that are performed when the described conditions of an associated rule are met. Activities are typically used to display text, graphics, or other multimedia enhancements, or perform a defined operation or command.



### Inference engine

Rules, variables, and actions are selected, tested, and executed by the inference engine. The inference engine is the brain of a decision support system. It is also referred to as the control structure or the rule interpreter. The inference engine selects the rules to be tested, tests the rules, and determines when a rule has met the defined conditions. Because all of the information of a DSS is contained in the rule, variable, and activity files, the inference engine does not need to be reprogrammed in order to change the application. An application can be modified by simply changing the rules, variables, or activities.

Rules are selected on one of two approaches which control inference for rule-based DSS: forward or backward chaining. The type of chaining which is most useful for a given application depends on the characteristics of that application. In some instances, both types of chaining can be implemented within one DSS. Forward and backward chaining are described in the following paragraphs.

Forward chaining. Forward chaining is a data-driven approach. The program starts from available information or a basic idea and then tries to draw conclusions. This is accomplished by analyzing the variables or facts that match the antecedent portion of a rule. As each rule is tested in this manner, the computer works its way towards a conclusion (Turban, 1990). In forward chaining, rules are simply tested in the order that they appear within the rule base. When an unknown variable is encountered, a query is performed to determine the variable's value (EXSYS, 1988).

Backward chaining. Backward chaining is a goal-driven approach. The computer starts from an expectation of what is going to happen and seeks evidence to support or contradict that expectation. The program starts with a goal to be proven either true or false and then looks for rules that contain that goal in their conclusion (THEN or ELSE clauses). It then checks the variables contained in that rule to see whether the defined conditions are met to reach that goal or conclusion. If these conditions are not met, the program looks for another rule that contains the desired goal and tests it in a similar manner. This process continues until all possible rules are checked or until the goal is satisfied (Turban, 1990). When backward chaining is used, the order of the rules does not matter. If a new rule relevant to the decision process is needed, it can be added anywhere within the rule base.

#### Object-oriented programming

Object-oriented programming (OOP) is a useful way to represent knowledge, and its use in DSSs is increasing rapidly. OOP is a method of programming based on the use of objects that communicate with one another using messages. Each object has rules and procedures associated with it. When an object receives a message, the object processes the message and carries out its commands. OOP can therefore be highly modular, can perform local actions, and can both receive information from and send information to other objects. Objects represent knowledge that can be used over and over again. Objects can be reused within the

same application or even within other applications. OOP also provides a simple method for unifying rules, variables, actions, and other data (Hu, 1989).

Encapsulation, inheritance, and polymorphisms are the three main properties that characterize OOP (Borland International, 1990). These concepts will be explained in the following paragraphs.

Encapsulation. Encapsulation is a process that is used to combine objects with functions that use and manipulate information contained within the object. It can simply be viewed as a process that welds code and data together into a single object. This process makes objects easier to work with because it reduces the chance for error and creates a more controlled environment that is easier to perceive.

Inheritance. Inheritance is the process by which an object assumes the characteristics and properties of another object of the same class. Inheritance allows objects to contain common features, yet be as specialized as needed. For example, insects could be classified as shown in Figure 8 and each ellipse could be considered an object. The “insects” object would be defined to contain characteristics and features that all insects have. The “winged” and “wingless” objects could each be created containing the same characteristics defined for the “insects” object as well as specialized characteristics. The “bees,” “moths,” and “flies” objects each contain the properties associated with the “winged” object, yet each of these objects has a set of very distinct properties that define its object (Borland International, 1990).

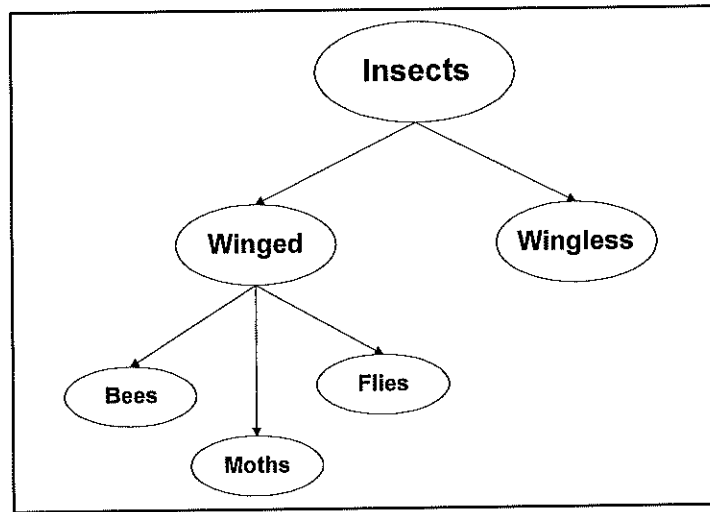


Figure 8. Inheritance example.

Once the characteristics of an object are defined, all of the objects beneath that object include those characteristics. Higher levels contain the general characteristics, and lower levels contain more specific ones.

Polymorphism. Polymorphism is a Greek word meaning “having many shapes.” It involves the ability of an entity to refer to instances of various classes at run-time (Gonzalez and Dankel, 1993). This is accomplished using virtual functions that allow many versions of the same function to be used throughout a class hierarchy, with the version to be implemented determined at run time.

#### Flexpert implementation

The decision support systems implemented into the Drainage Library were authored using Flexpert. Flexpert is an expert system/decision support system

authoring tool that is under development by the Computer-based Intelligent Technology (C++BIT) Laboratory at Utah State University.

Flexpert is a rule-based, object-oriented system that provides a friendly environment for the development of decision support systems. Figure 9 depicts the layout of a decision support system authored using Flexpert. The domain expert or author creates the rule base by preparing "Activities," "Variables," "Rules," and "Resources." Individual activity, variable, and rule files can be created very easily using the interfaces provided by Flexpert.

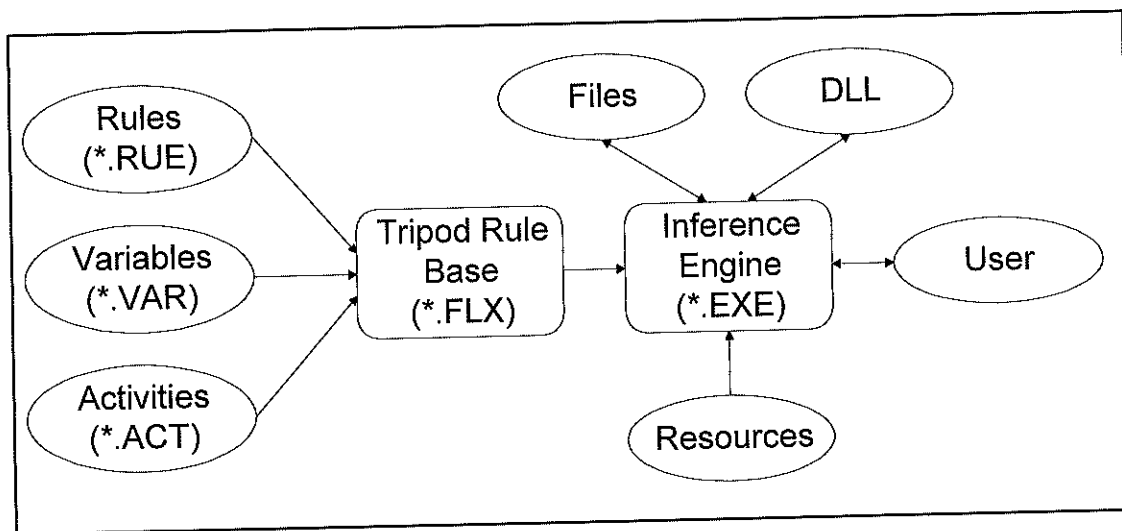


Figure 9. Flexpert application process.

Rule file. Variables can be tested, and actions performed depending on the value of a variable using rules. Rule files are identified by the extension \*.RUE. Rules can be formed within Flexpert using the "Rule Guru," which is shown in Figure 10.

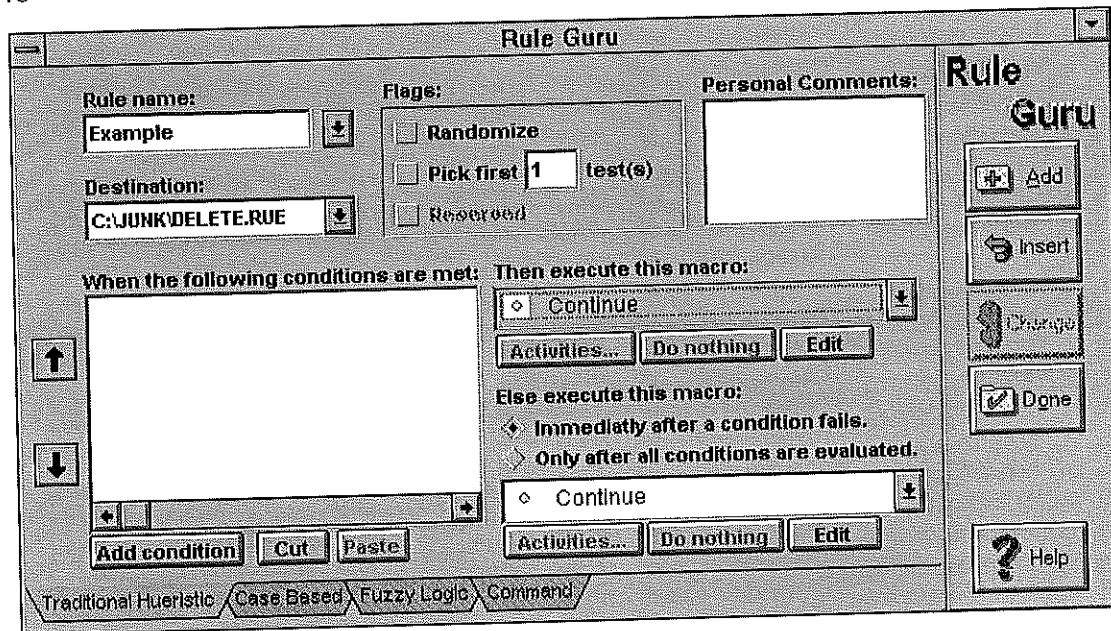


Figure 10. "Rule Guru."

The antecedent portion of a rule is made up of "conditions" that are defined using the "Rule Guru." These conditions are tested within the rule, and the consequences of a rule are fired depending on the results of these tests. The author can define single actions or activities to be performed when these tests are determined to be either true or false.

Flexpert also allows rules to be tested by other rules. Individual rules are arranged within the rule file to form a useful decision support system. The rule file developed for the culvert design decision support system that was implemented into the Drainage Library is displayed in Appendix B.

Variable file. The variable file is identified by the extension \*.VAR. Variables can be created very simply using the "Variable Guru." The "Variable Guru" is shown in Figure 11.

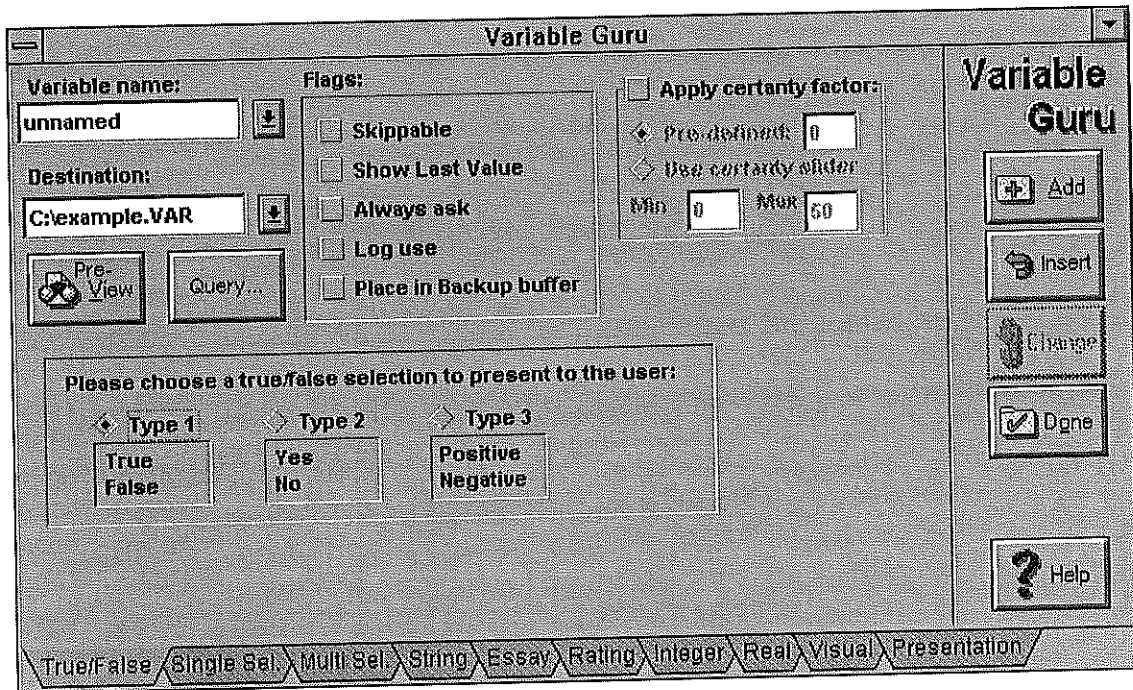


Figure 11. "Variable Guru."

The "Variable Guru" allows for the creation of several different and distinct types of variables. The tabs along the bottom of the guru, Figure 11, show all of the types of variables that can be created using Flexpert. As can be seen in the figure, Flexpert allows for the creation of variables that can contain a true or false value, a single or multiple selection (menu), text (string or essay), ratings, real and integer numbers, or even visual variables. Visual variables can be used to display a bitmap and have the user select a region or feature within the bitmap. Flexpert also allows for the creation of presentation variables. Presentation variables are useful for displaying information or results within a DSS.

By selecting the "preview" button, the author accesses an interface that allows him to enter text, usually in the form of a question, to invoke a response from

the user for each variable. This interface also allows the author to create buttons and add multimedia enhancements to customize the individual variables.

Appendix C contains the variable file utilized in the development of the culvert design decision support system.

Activity file. Activity files typically consist of more than one action or command that an author desires to perform within the decision support system. An individual action can be executed from a rule, but if more than one action needs to be fired at a given instance, an activity is used. Activities can be reused and the same activity can even be executed from several different locations within a rule base. The activity file is identified by the extension \*.ACT. Activities can be created very simply using the "Activity Guru." The "Activity Guru" is shown in Figure 12.

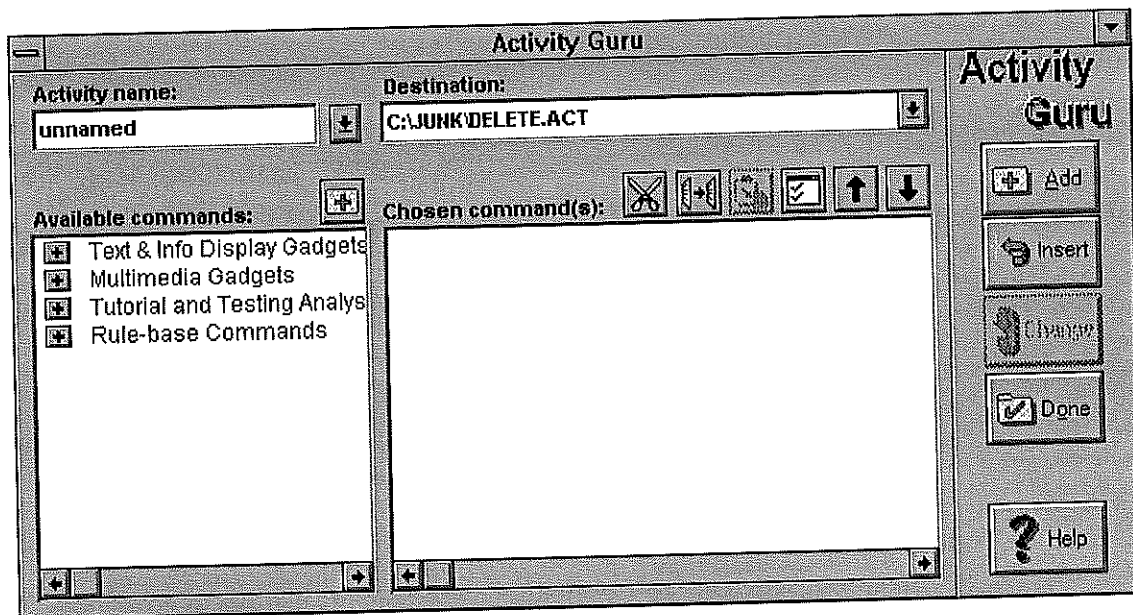


Figure 12. "Activity Guru."



The “Activity Guru” allows the user to create activities that display text and multimedia enhancements or perform any of a large number of rule-based commands.

Appendix D contains the activity file utilized in the development of the culvert design decision support system.

Tripod rule base. Variable, rule, and activity files are created and compiled using the Flexpert Integrated Development Environment (Flexpert IDE). The Flexpert IDE component contains an executable (FM.EXE). This executable creates a single knowledge-base file known as the tripod rule base from the variable, rule, and activity files. The tripod rule base has extension \*.FLX and can be utilized by the Flexpert inference engine along with resources and files to interact with the user.

Inference engine. The Flexpert inference engine contains a run-time module (Flexpert.EXE) which utilizes the tripod rule base (\*.FLX) created using Flexpert IDE.

The Flexpert inference engine has the ability to perform both forward and backward chaining. This is a unique feature in that most decision support system authoring tools support only one of the two types of chaining. When using Flexpert, the author can decide what type of chaining best fits the given application and can even use both types of chaining within the same rule base.

Forward chaining is automatically implemented unless the author specifies otherwise. Forward chaining is a procedural approach because rules are fired in the

order that they appear within the rule base and any variables that do not contain a value are immediately queried.

Backward chaining can be implemented on a per variable basis within Flexpert. A check box titled "back chain" can be checked on the "Variable Guru," Figure 11, to make a variable backward chaining. This allows the author the option to make the entire rule base or just an individual variable backward chaining. If the back chain box is not checked, Flexpert assumes that the variable is forward chaining. Since backward chaining is implemented on a per variable basis, a given rule could contain variables that are both forward and backward chaining. When backward chaining is implemented for a variable, Flexpert tests rules that contain the given variable in their consequence portion. If Flexpert is unable to obtain a value for the variable in this manner, after it has tested all of the rules that contain the variable in their consequence, it will directly query the variable.

Resources. Resources typically consist of pictures, movie clips, audio segments, and hypertext documents that can be directly incorporated into the decision support system. Text and data files can also be read from or written to by the inference engine.

User-defined dynamic link library. Figure 9 shows a dynamic link library (DLL) interacting with the inference engine. The mathematical calculations required for the culvert design decision support system were performed in the wrapper using a DLL. A DLL is a collection of functions, data, and resources. Flexpert is capable

of calling and sending necessary parameters and data, as well as retrieving calculated output from a DLL. The references of a DLL are resolved each time the application is run, rather than when it is originally compiled (Borland International, 1993).

Encapsulation. The variable, rule, and activity files created by Flexpert are objects that incorporate the concept of encapsulation. Activities encapsulate Flexpert commands, including text and information display, control gadgets, multimedia enhancements, and rule-based commands that allow the author to change the state of the rule base. Activities are encapsulated by rules because they can be reused and the same activity can be fired by more than one rule. Variables and rules can also be encapsulated by rules.

Inheritance. Flexpert implements the inheritance process through its use of templates. Flexperts' template menu is shown in Figure 13. Flexpert allows for two different types of templates to be implemented: base-knowledge objects and cover layers. Base-knowledge objects, also known as master layers, define components that the author wishes to keep uniform from one variable to the next. Variable characteristics such as background bitmaps, control buttons, and other gadgets are typically defined for variables within a rule base using base-knowledge objects. Flexpert allows the author to define more than one base-knowledge object, and specify when each is to be executed within the rule base. By changing the characteristics of a base-knowledge object, the characteristics of all of the variables associated with that object are also be changed.

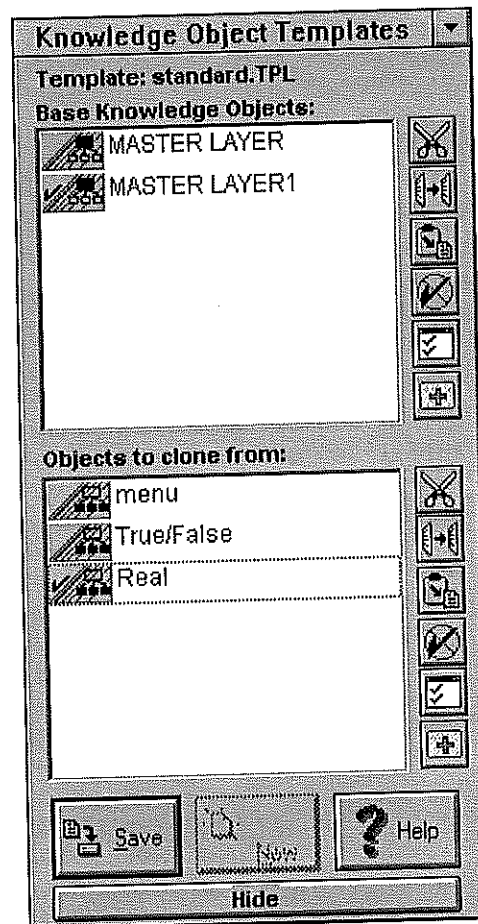


Figure 13. Flexpert object templates menu.

Cover layers, also known as cloneable objects, demonstrate restricted inheritance properties. Cover layers can be used as templates for the creation of variables. Typically a cover layer would be made which describes the characteristics of each type of variable utilized within the rule base. For example, Figure 13 shows three cover layers that have been defined for the creation of menu, true/false, and real variables. Cover layers can be defined and changed by the author. When he creates a variable, he would use the appropriate cover layer as a template. The characteristics defined on the cover layer would be inherited by each variable created

from it. However, once a variable is saved, its inherited characteristics are saved and cannot be changed by changing the cover layer.

### C Programming Language

The C programming language was used to create numerical modules which were incorporated into the Drainage Library. These modules include custom equation solvers, the HyperCalc utility, and other DLLs which were accessed from the DSS to perform mathematical calculations. These applications were described in detail within the Features section of this report.

### Equation Solvers

Equation solvers were developed using a DLL that was written in the C programming language. The portion of this DLL which generates the equation solver displayed in Figure 7 is displayed in Appendix E. This DLL is accessed by Viewer and passed the necessary parameters. Viewer passes a variable to the DLL that identifies which instance (equation) of the equation solver is being called. These variables are associated with the hypergraphic or hypertext that are used to start the equation solver application. Viewer actually loads a DLL at the exact moment that is used within a document, not when the document is first entered. In order for Viewer to load and access a DLL, it must also be registered (using the Register Routine Command) and its exact location specified within Viewer. The equation solver DLL was developed to support multiple instances, so that it can be accessed

numerous times from within the same Viewer application. This allows the user to use the same equation solver over and over again as well as have several equation solvers open and running at the same time.

### HyperCalc

HyperCalc is a stand-alone application that was developed at the C-BIT Laboratory at Utah State University. It was created using the C programming language and called from within Viewer using an external program link. The DLL that was developed for the equation solvers was used within HyperCalc to perform the drainage equation calculations.

### Integration of Applications

Several individual programs or modules can be incorporated into a single application. Three different types of models were created for the Drainage Library using the authoring tools discussed within this section. Figure 14 shows how these individual modules interact with each other and how they can be linked together to form a single application.

As can be seen in Figure 14, the user can directly access each of the three different types of modules. The user can also access any of these modules from within another module. These modules interact with each other through application programming interfaces (APIs). They could also interact with any other program or application which conforms to API specifications.

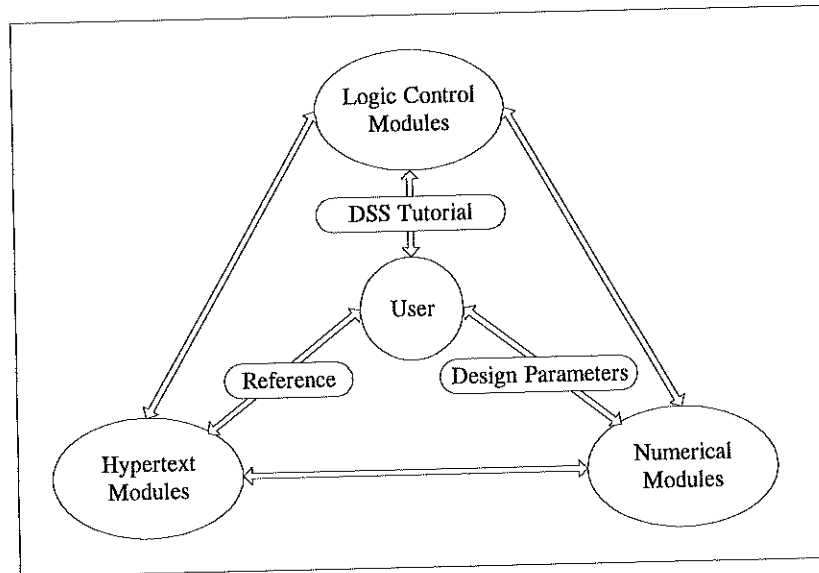


Figure 14. Hypertext, logic, and numerical modules interaction.

### Hypertext module

The hypertext module typically contains individual topics that consist of text, graphics (bitmaps), wave files, video clips, etc. The hypertext module incorporated into the Drainage Library was authored using Viewer. Both logic control and numerical modules are accessible from hypertext modules. Logic control modules are used to help a user perform design procedures or processes that are outlined within the hypertext module. Numerical modules are utilized from hypertext modules to help the user perform calculations of necessary design parameters.

### Logic control module

Logic control modules consist of knowledge bases or DSSs. The logic control module that was implemented into the Drainage Library was a DSS that was created using the Flexpert authoring tool. Both hypertext modules and numerical

modules can be accessed from logic control modules. Hypertext modules are typically accessed from DSSs to supply the user with definitions and background information to make necessary decisions. For example, the culvert design DSS of the Drainage Library contains a "More Info" button that, when selected, links the user with a Viewer topic. Numerical modules can be executed from a DSS to perform necessary mathematical calculations. For example, the culvert design DSS accesses a numerical module in the form of a DLL that is used to perform the essential calculations for the design of a culvert.

#### Numerical module

Numerical modules are programs which are utilized to perform mathematical calculations. Equation solvers and HyperCalc are examples of numerical modules which were implemented into the Drainage Library. These modules were created using the C programming language. Numerical modules can be used to access both logic control and hypertext modules. Numerical values were passed back and forth between the logic control and numerical modules within the Drainage Library. Hypertext modules are typically executed from a numerical module to provide help or background information for the user. For example, the HyperCalc application contains a "Help" button. When this button is selected, a hypertext topic is displayed which contains context-sensitive help.



## AUTHORING PROCESS

The process of authoring a computerized document can be very time consuming and requires careful planning and preparation. In order to take full advantage of the computer's potential power, more needs to be done than simply transferring the material contained within a book to a computer-based document. Within this section, the design approach followed in the creation of the Drainage Library will be outlined and discussed.

Figure 15 shows a flow chart of the authoring process that was followed in the creation of the Drainage Library. This figure will be explained in detail in the following portions of this sections.

### User Needs Analysis

The first step in the process of developing a computer-based reference manual is to evaluate the expected needs of the users of the manual. This can be accomplished by interviewing the potential user group and discussing what they would like the application to accomplish. It is also important to involve the advisory committee in this process so that their goals and long-range objectives are understood. Continually involving the user group and the advisory committee in the authoring process will help ensure that their needs and wants are understood and met.

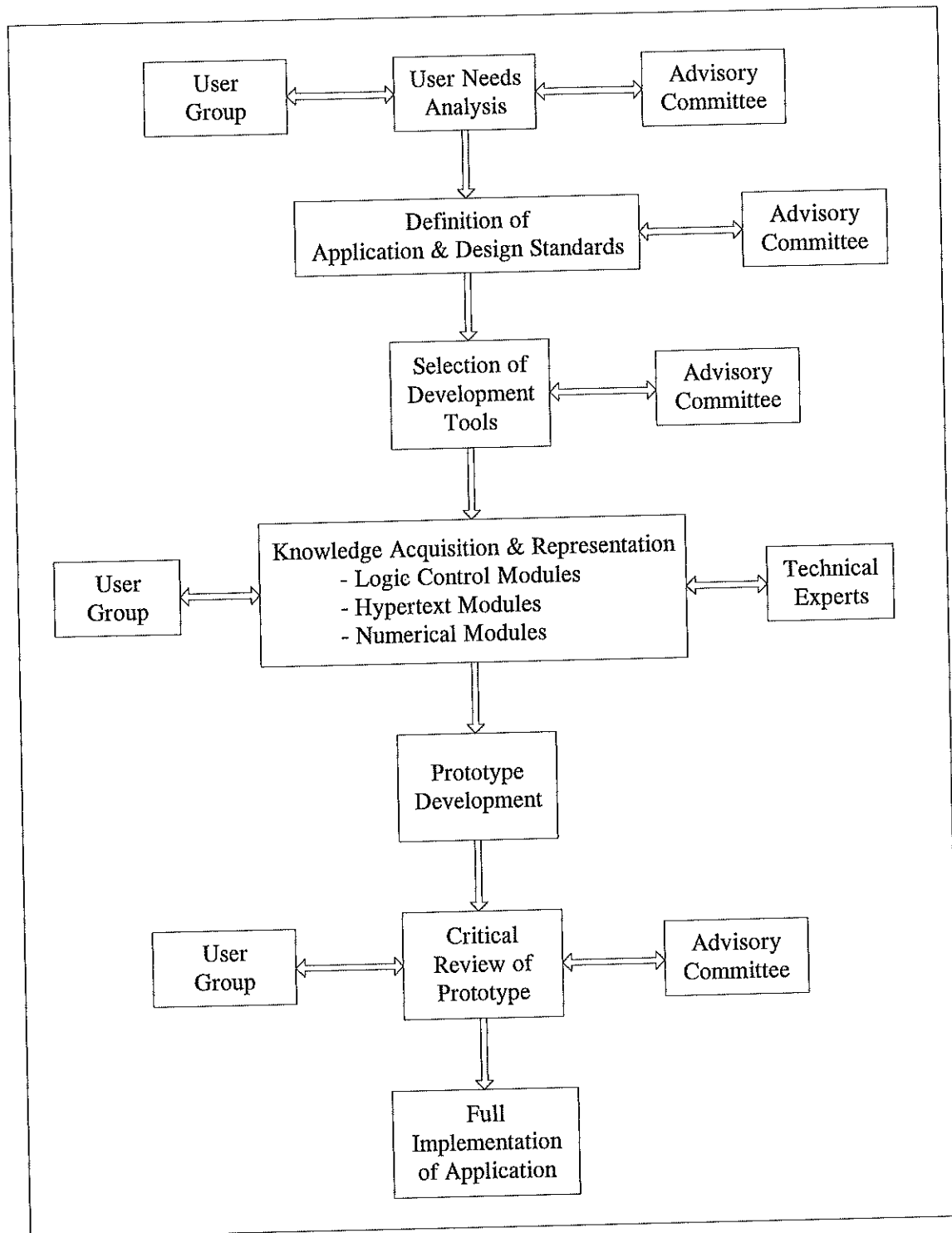


Figure 15. Computer-based manual authoring process flow chart.

### Definition of Application and Design Standards

The usefulness of a computer-based manual can be substantially increased when information is effectively organized. The establishment of an application definition and design standards can simplify the authoring process and increase the value of the information contained in an application.

#### Application definition

After the expected needs of the user have been determined, the author should meet with the advisory committee to define the application. At this point, major objectives and goals should be set. Effectively defining the application can simplify the authoring process as well as ensure that a useful product is created.

#### Design standards

Establishing consistent design standards helps keep a computerized document uniform and easy to understand. Within a computer-based document, it is as important as within a book or other printed document that a certain style and look be maintained. Fonts, font sizes, color, topic sizes, margins, and spacing should be kept consistent throughout the entire document. Topics should be arranged so that they are consistent and easy for the user to understand. Viewer can control which topic or subtopic a given jump can access, but cannot jump to specified locations within a topic. Topics should therefore be organized small enough to allow jumps to

access desired information found within the topic. When information is effectively linked within a document, the user can find desired information more quickly. Related topics can be cross referenced, allowing the user to quickly access related information.

Multimedia elements should also be presented in a uniform manner that fits in well with the rest of the document. Inconsistent use of these elements can detract from the informational content of the document and can be confusing for the user (Microsoft Corporation, 1994).

Large amounts of time and effort can be saved if design standards are identified at the beginning of the authoring process and strictly followed throughout the entire process.

### Selection of Development Tools

After the users' needs have been analyzed and the application has been defined, development tools must be evaluated and selected. These development tools include both the software and hardware the author will use to create the application. Several factors play an important role in this selection process. Tools must be selected which are capable of providing the functionality defined by the application objectives. Cost, and software and hardware requirements, as well as ease of use, also play an important role in this decision. The authoring tools used to develop the Drainage Library are described within the Development Tools section of this report.

The hardware and software used to develop the Drainage Library are outlined in the following paragraphs.

The hardware and software requirements of authoring systems often differ from those of the system a user can use to view a document. The Drainage Library was authored to run on an IBM compatible microcomputer using the Microsoft Windows platform. In order to efficiently run this system, a computer should contain at least the following hardware and software (Microsoft Corporation, 1993):

- 80386 or better processor running at a minimum of 33 MHz
- 6 MB RAM
- 30 MB hard-disk storage
- VGA+ (256 color) monitor
- 16- or 32-bit digital audio device (Sound Card)
- Speakers and/or Headphones
- Windows 3.1 or later

The Drainage Library was authored using a computer that contained the following hardware and software:

- 80486 based computer running at 66 MHz
- 16 MB RAM
- 430 MB hard-disk storage
- VGA+ (256 color) monitor
- 16- or 32-bit digital audio device (Sound Card)
- Video Capture Board (Video Blaster or Targa Board)
- CD-ROM Drive
- Speakers and/or Headphones
- Microsoft Windows 3.11
- Microsoft Multimedia Viewer
- Microsoft Word for Windows (6.0)
- RoboHELP (2.6)
- Flexpert
- Borland C++ 4.0
- Paintbrush, Corel Draw, and WinGif
- Digitizing Software

Applications that do not contain audio segments do not require a sound card or speakers. The Drainage Library could be run on a system that does not contain a sound card or speakers, but the user would obviously not be able to hear the sound associated with the audio and movie clips. Other hardware that was used to author the Drainage Library is shown below:

- Color Scanner
- Home Video Camera (SVHS or Hi8)
- Zap Shot Camera
- Slide Scanning Device
- Microphone

The author must take into account how the application he creates will appear on the computer that the user will actually be using. Authors often have superior equipment that make certain aspects of an application appear better on his system than they may on the user's.

### Knowledge Acquisition and Representation

Acquiring the knowledge to place in a computer-based document can be one of the most important, difficult, and time consuming phases in the development process. Knowledge can be acquired from many sources, the most common being experts on the subject and published reference materials. The user group often contains experts on the given subject, and can be very useful in the knowledge acquisition process.

Knowledge in the Drainage Library application was represented using logic control, hypertext, and numerical modules as previously defined. It is important that knowledge be represented in a manner that is easy for the user to access and understand. Effective use of multimedia elements can greatly facilitate this process.

Use of a previously defined structural outline can help the author organize material consistently so that it is easy for the user to understand. The structural outline used for the development of the Drainage Library is outlined in Table 2. Terminology was adopted to represent the levels of information of the structural outline for the hypertext modules. Since written documents are a familiar medium through which information is conveyed, this terminology was compared to that of a textbook. This terminology is summarized in Table 2.

Table 2. Structural outline and terminology used for the development of computer-based manuals

Level	Computer-based Terminology	Printed Copy Equivalent
1	Library	Group of books or manuals
2	Document	Book or manual
3	Segment	Chapter
4	Topic	Page or section
5	Subtopic	Subsection
6	Element	Paragraph, table, figure, definition, bitmap, video clip, etc.

A "library" is a body of information about a subject like a group of books or manuals related to a specific topic. The application developed for this project could be called a library since two separate manuals, The AASHTO Drainage Manual and The Methods for Estimating Peak Discharge and Flood Boundaries of Streams in Utah Report, are incorporated into the computerized document. Volumes of information equivalent to a book, manual, or report will be referred to as a "document." Documents may be divided into "segments," equivalent to chapters of a book. A "topic" would be similar to a page within a chapter and can be further divided into "subtopics." Subtopics contain a single concept or operation that can be represented at one time on the computer screen. "Elements" are used to create subtopics and consist of paragraphs of text, tables, graphics, video clips, etc.

Consistent use of a defined structural outline (Table 2) will aid the author in organizing the information contained in a document and will also help the user effectively navigate within the document. Topics should be arranged so that they are consistent and easy for the user to understand. Viewer can control which topic or subtopic a given jump can access, but cannot jump to specified locations within a topic. Topics should therefore be organized small enough to allow jumps to access desired information found within the topic. When information is effectively linked within a document, the user can find desired information quicker. Related topics can be cross referenced, allowing the user to quickly access related information.



### Prototype Development

Before an entire application is developed, a prototype should be constructed. A critical review should be performed on the prototype by the advisory committee and the user group. The purpose of this evaluation should be to determine any changes or modifications which should be made to the authoring process before final implementation occurs. This helps to ensure that the users are provided with a useful and effective tool.

### Debugging Process

After an application has been fully implemented and developed, debugging must be performed. Figure 16 shows a flow chart of the debugging process that was followed for the Drainage Library. The advisory committee and user group were provided with the alpha version of the application and asked to test it for any errors. Upon their recommendation, problems and errors were corrected and they were presented with the beta version of the application. Debugging continued until they were satisfied with the application. After this debugging process, final distribution of the application took place. The entire Drainage Library Application took up approximately 20 megabytes of space.

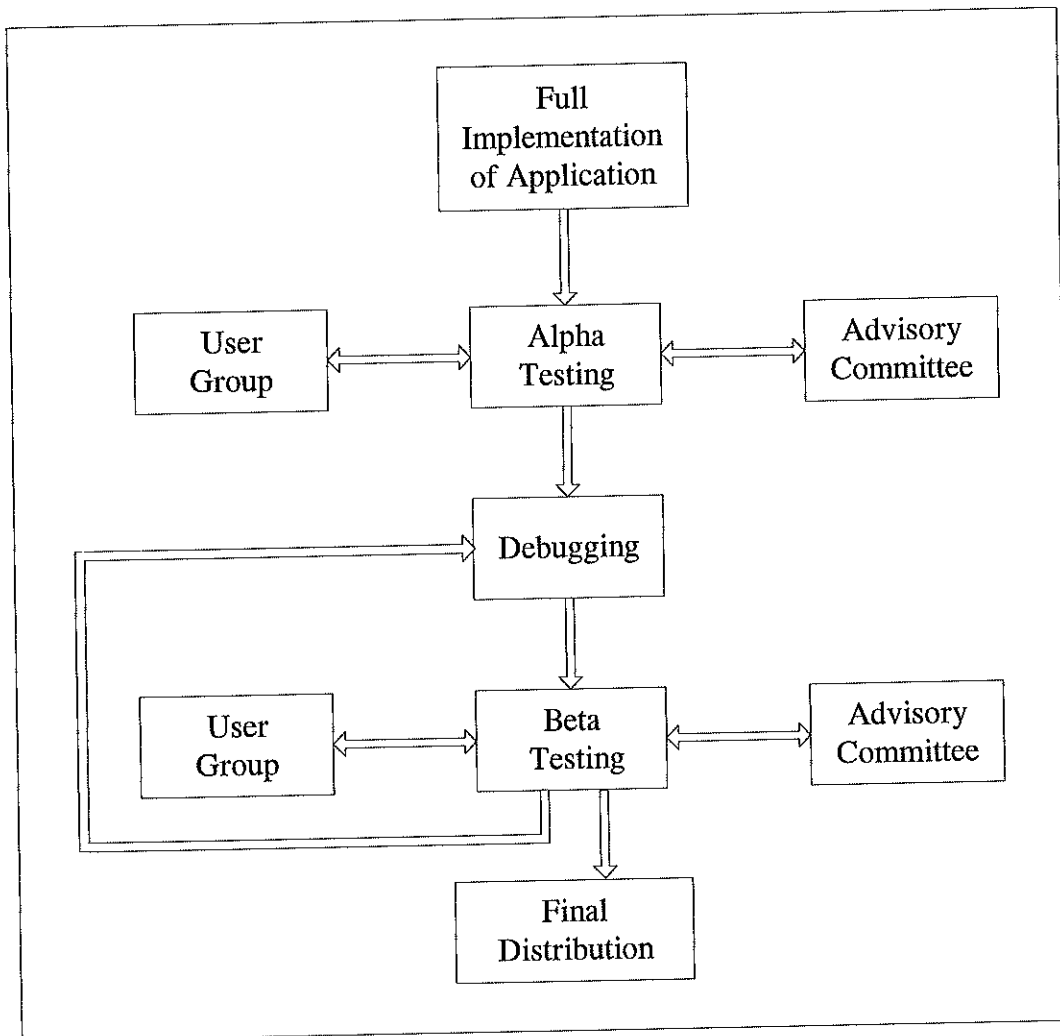


Figure 16. Debugging process.

### Updating of Manuals

After final implementation of an application occurs, updates and changes must often be made. As design procedures, policies, regulations, and other information contained within manuals of instruction change, they must be updated. This process can be very time consuming and expensive for written manuals.

Security can also be an important issue because individual users should be prohibited from making changes to existing manuals. After updates are made, it is also important that the revised version be distributed so that everyone is using the corrected version. Computer-based manuals can be updated quickly and relatively inexpensively, can provide the necessary security, and can be redistributed to individual users quickly and effectively.

Updates can be made to hypertext modules by making the desired changes to the RTF files and recompiling the document using the Viewer Compiler. These changes can only be performed by an individual who has the original source files (RTF files) and the necessary software. After this person makes the desired changes to the manual, he would then compile the files. Changes cannot be made to a compiled Viewer file by the individual users. The person with the source files and Viewer software would be the only person with the capabilities of making changes to and updating a Viewer file. Individual users can, however, add bookmarks and annotations, as well as print topics.

After desired changes and updates are made, the network manager could replace the version of the manual on the network with the revised version. This would ensure that everyone was using the same updated version of the manual and could be performed very easily and quickly.



## EVALUATIONS

An evaluation was performed to compare the effectiveness of the Drainage Library computer-based manual to the printed version of the same manual. This evaluation included two phases. The first phase consisted of a user evaluation that was performed by several Utah Department of Transportation professionals. The second phase of the evaluation was performed by the authors of the Drainage Library. The results of both phases of this evaluation are discussed within this section.

### User Evaluation

A demonstration of the Drainage Library and the features it contains was given to fourteen professionals at the Utah Department of Transportation (UDOT). The application was also installed on several computers to which they had access. Each of the evaluators was each asked to fill out an evaluation form which was designed by the author with the approval of the UDOT advisory committee. A copy of this form is shown in Appendix F. Since the purpose of this project was to evaluate the use of computer-based manuals, the evaluators were asked to use the Drainage Library as an example and base their evaluations on the overall concept of computerized manuals.

The first question on the questionnaire asked the evaluators to rate the importance of each of the features of a computer-based manual and the usefulness of

each feature for training, reference, and decision support. Training would include the use of the computer-based manual to teach or instruct a user that is not proficient in the subject matter. Reference describes the process that a user, already knowledgeable about the subject, goes through to look up values, data, procedures, or other desirable forms of information. Decision support characterizes the ability of the computerized manual to aid the user in the process of making a decision. The features evaluated for each of these three criteria included: the browser, hypertext, popup windows, search capabilities, glossary, history list, bookmarks, annotation, multimedia effects, equation solvers, external program links, HyperCalc, and decision support system. Those evaluating the system were asked to rate the importance of each feature between 0 and 3 as defined below.

- 0 = not important
- 1 = little importance
- 2 = important
- 3 = very important

The replies to this question are displayed in Appendix G along with confidence intervals calculated for each rating. Table 3 contains a summary of these results, the average response for each category and feature. Figures 17 and 18 show graphical representations of these data.

Table 3. User importance ratings of the computer-based manual features (0 = not important, 1 = little importance, 2 = important, 3 = very important)

Feature	Training	Reference	Decision Support	Average
Browser	2.64	2.79	1.54	2.32
Hypertext	2.86	2.79	1.85	2.50
Popup Windows	2.38	2.62	2.17	2.39
Search	2.43	2.93	2.15	2.50
Glossary	2.79	2.50	1.54	2.27
History List	1.86	1.79	1.46	1.70
Bookmarks	2.21	2.50	1.62	2.11
Annotation	2.15	2.54	1.92	2.20
Multimedia	2.57	2.14	1.15	1.96
Equation Solvers	2.36	2.29	2.85	2.50
External Program Links	2.14	2.21	2.62	2.32
HyperCalc	2.21	2.21	2.46	2.30
Decision Support System	2.14	2.21	3.00	2.45
Average	2.37	2.42	2.02	2.27

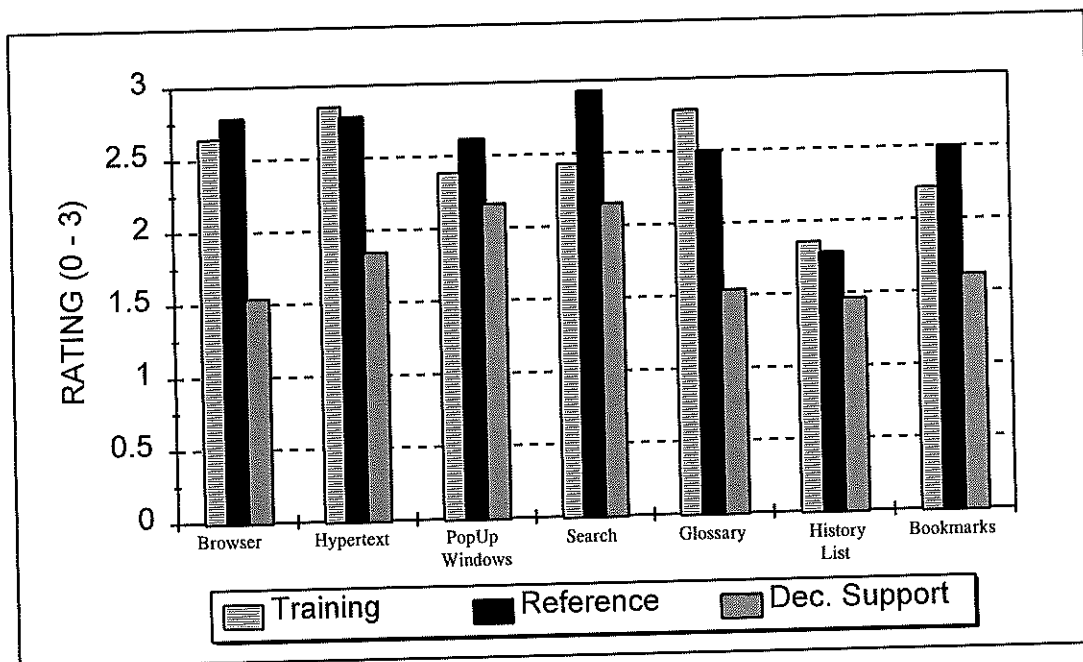


Figure 17. Users evaluation of computer-based manuals features (0 = not important, 1 = little importance, 2 = important, 3 = very important).

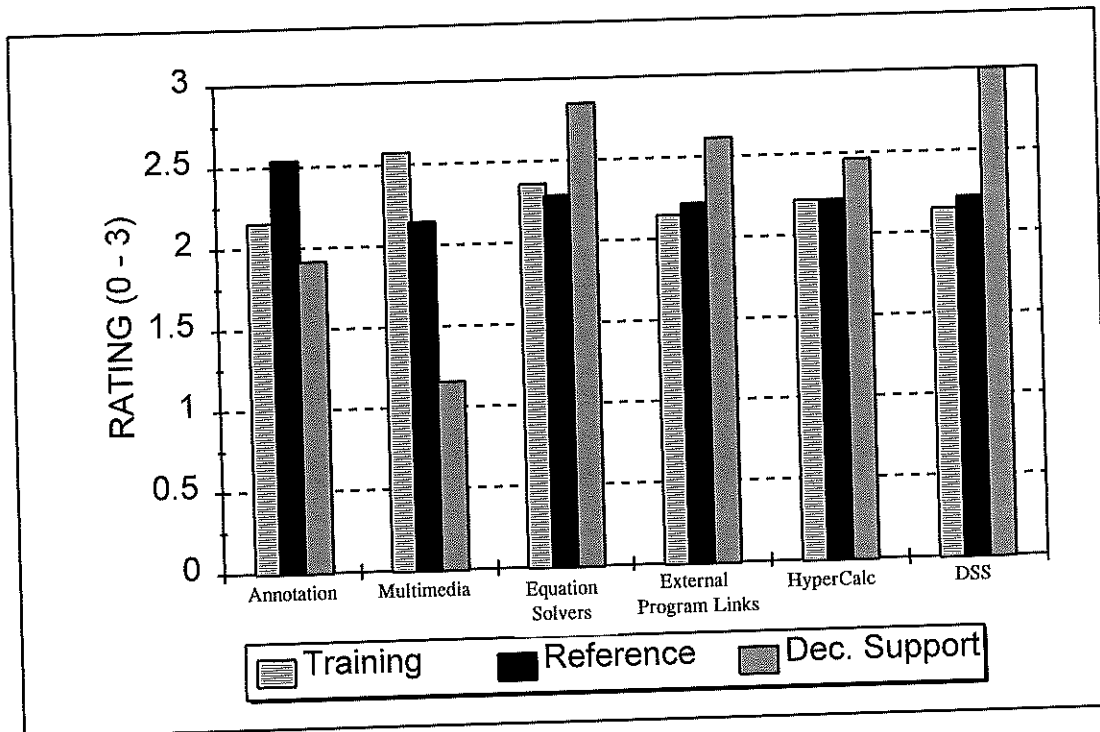


Figure 18. Users evaluation of computer-based manuals features continued (0 = not important, 1 = little importance, 2 = important, 3 = very important).

It can be noted in Table 3 and Figures 17 and 18 that all of the features evaluated were thought to be important in at least one of the three categories. It is also apparent from the wide range of replies received from the evaluators (Appendix G) that the features will most likely have varying degrees of importance to different users. Depending on what the purpose of a computer-based manual is (training, reference, or decision support), the author should focus his efforts on the features determined to be most important in that category.

The second question of the questionnaire asked the user to compare the computer-based Drainage Library to the printed version of the AASHTO Drainage



Manual. The evaluators were asked to compare these two forms of the manual for each of the following aspects: speed, accuracy, navigation (ability to maneuver and find information), overall ease of use, level of information, comprehension of the information, and unit conversions (using conversion table and a calculator versus HyperCalc and Equation Solvers). To accomplish this, the evaluators were asked to decide which version (computer-based or printed) they thought was better and rate how much better they felt it was using the following scale:

- 0 = about the same
- 1 = slightly better
- 2 = significantly better
- 3 = much better

The evaluators felt that the computer-based version of the manual was superior in every one of the aspects. These results are displayed in Table 4, which contains the average response, the standard deviation, and the 95 percent confidence interval for each aspect. Figure 19 shows a graphical representation of the data.

Table 4. User comparison of computer-based manual to printed version (0 = about the same, 1 = computer version slightly better, 2 = computer version significantly better, 3 = computer version much better).

	Average	Standard Deviation	95% Confidence Interval
Speed	2.78	0.44	3.03 - 2.53
Accuracy	2.33	0.87	2.82 - 1.84
Navigation	2.44	0.73	2.86 - 2.03
Ease of Use	1.89	1.17	2.55 - 1.23
Information Level	2.33	0.87	2.82 - 1.84
Information Comprehension	1.56	1.33	2.31 - 0.80
Unit Conversions	2.78	0.44	3.03 - 2.53

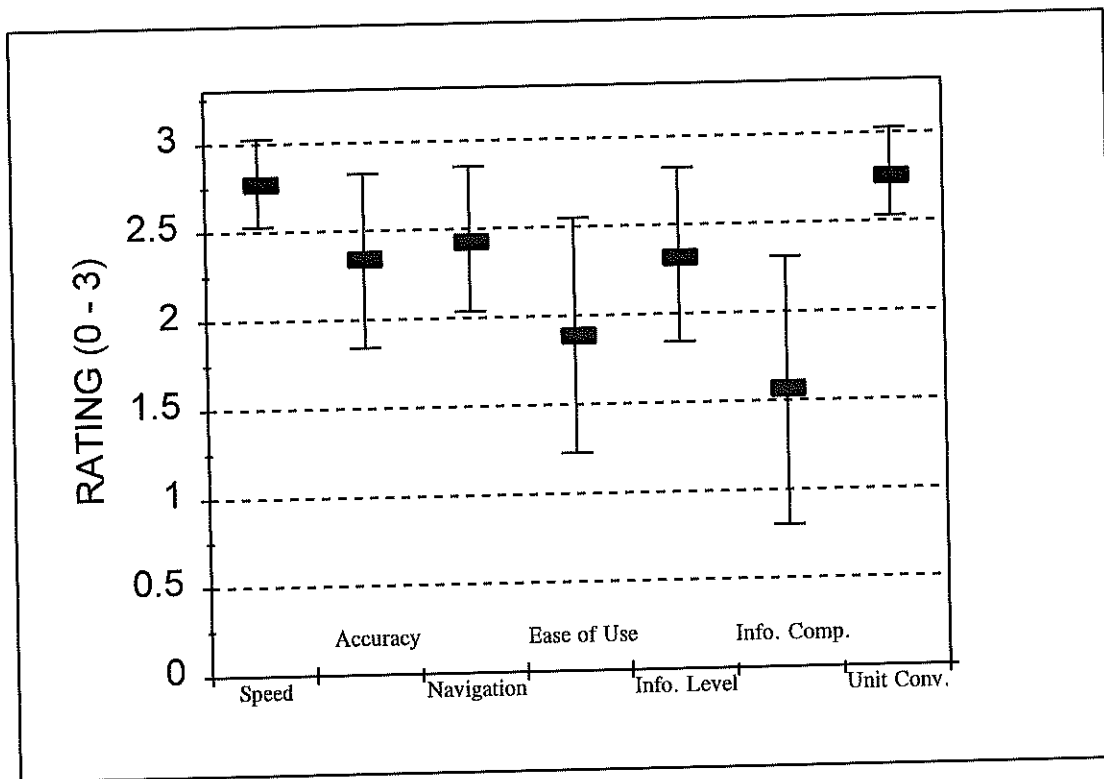


Figure 19. User comparison of computer-based and printed manuals (0 = about the same, 1 = computer version slightly better, 2 = computer version significantly better, 3 = computer version much better), average reply and confidence intervals are shown.

The evaluators were also asked to list what they felt the advantages and disadvantages of the computer-based manual were. Table 5 summarizes their responses to these questions. They generally agreed that the computer-based manuals let users access more information much more quickly than conventional manuals would allow. The evaluators also thought that the computerized manual offered more flexibility and would be easier to update and maintain. The major disadvantages that the evaluators mentioned were cost, both development and

hardware costs, and the need for computer knowledge to be able to use the system effectively.

Table 5. Advantages and disadvantages of the computer-based manual perceived by the evaluators

Advantages	Disadvantages
Easy to update	Cost of development
Offers interesting learning environment	Cost of required hardware
Potential to achieve quick solutions	Required computer knowledge
Better use of resources	Learning curve
Faster for obtaining information	Accessibility - can carry a book with you, but won't always have a computer
Multimedia useful for explaining concepts	
Easy to find desired data	
Search capabilities	
Flexibility	
Manual, equations, and software packages can all be bound together	

The evaluators suggested that a training session or tutorial be prepared to instruct users not familiar with the Windows platform on how to use the manual. They also felt that a help file would be useful and that all of the features and enhancements within the manual should be authored to perform in the same manner as all Windows applications. For example, they suggested that a user should be able to access the features using the mouse or through keystroke combinations.

### Author Evaluation

The author of the Drainage Library used the same evaluation questionnaire as the UDOT professionals to evaluate the effectiveness of the Drainage Library and computer-based manuals in general. The results obtained from that questionnaire are contained within this section.

Figures 20 and 21 show the authors' response to the first question of the questionnaire. This question asked the evaluator to rate the importance of each of the features of the computer-based manual for training, reference, and decision support purposes.

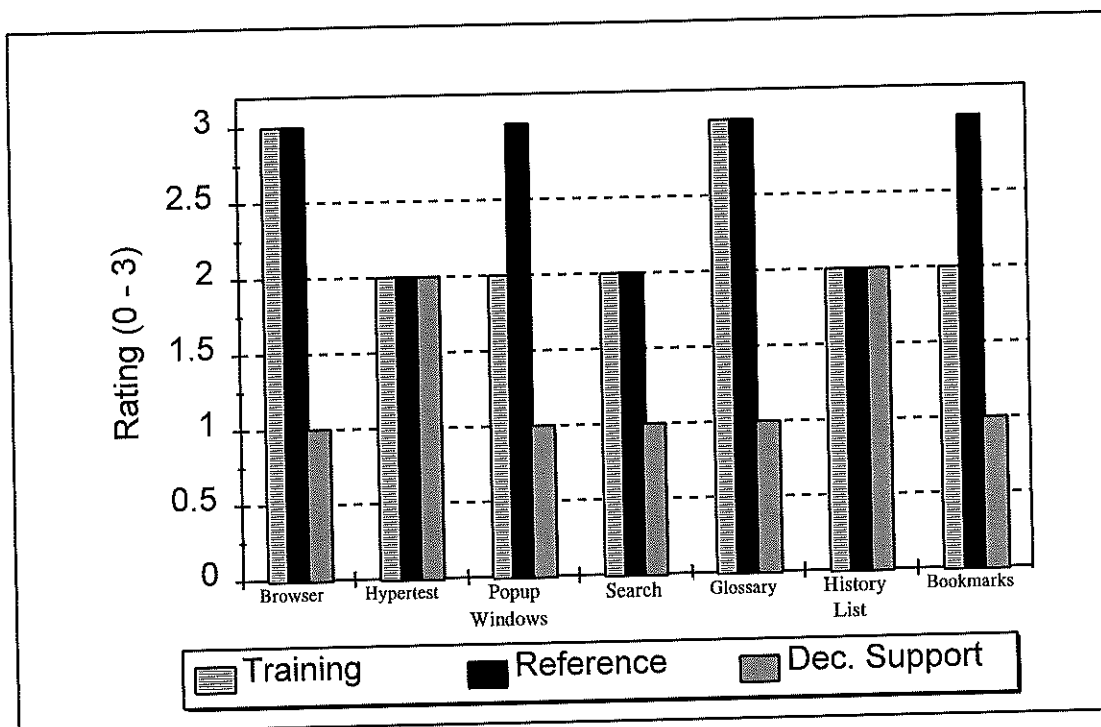


Figure 20. Author evaluation of computer-based manuals features (0 = not important, 1 = little importance, 2 = important, 3 = very important).

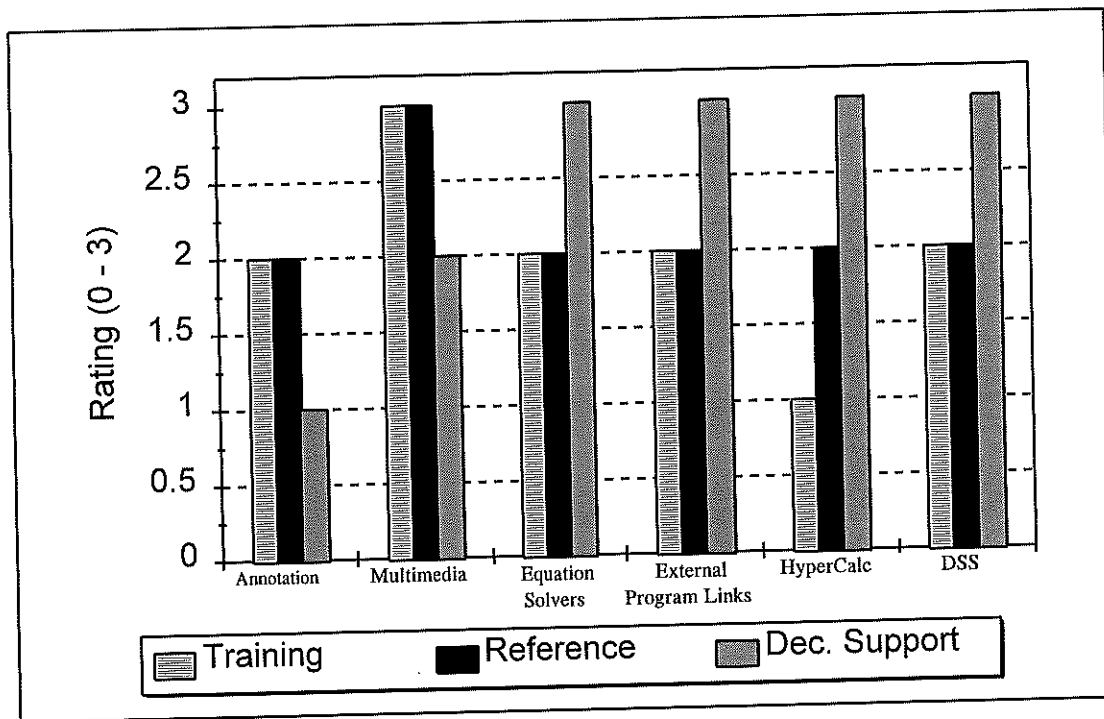


Figure 21. Author evaluation of computer-based manuals features continued (0 = not important, 1 = little importance, 2 = important, 3 = very important).

As can be seen in Figures 20 and 21, the authors' response was quite similar to that of the UDOT professionals. It is also apparent that features built using the hypertext authoring system (hypertext, popup windows, searches, bookmarks, annotation, etc.) rank higher in the training and reference categories than the decision support category. On the other hand, the equation solvers, external program links, HyperCalc, and decision support systems ranked very high in the decision support category. These trends were also apparent in the user evaluation.

The second question of the questionnaire asked the evaluator to compare the computer-based version of the Drainage Library to the printed version of the same

manual. Figure 22 shows the author's response to this question. The author also felt that the computer-based version of the manual was superior in each one of the aspects evaluated.

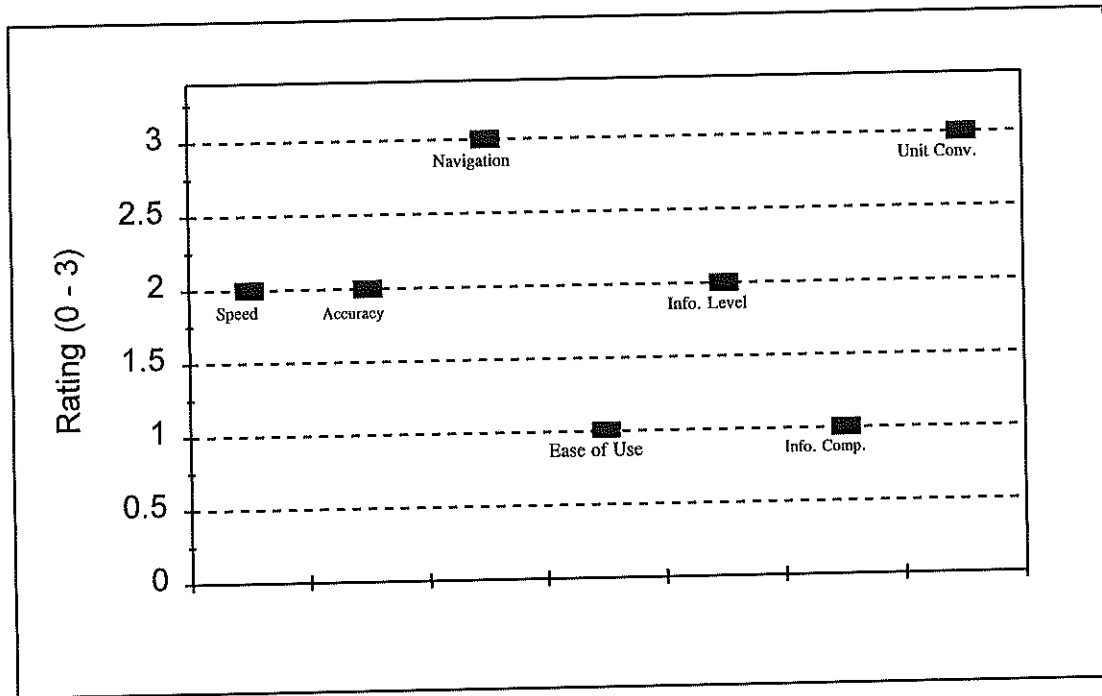


Figure 22. Author comparison of computer-based and printed manuals (0 = about the same, 1 = computer version slightly better, 2 = computer version significantly better, 3 = computer version much better).

The author felt that using the computer-based version of the manual was quicker. Equation solvers, search capabilities, the browser, and hypertext allow calculations to be performed and information accessed very quickly. He also felt that the accuracy of the computer-based manual was superior because the equation solvers, HyperCalc, and the decision support systems perform exact calculations in

place of estimations using charts and nomographs. Hypertext, bookmarks, the history list, and search capabilities make navigation in the computer-based manual preferable to that of the printed version. These same features also make the computerized manual easier to use. The addition of multimedia enhancements and external programs increases the amount of information contained within a manual and the level a user can comprehend. The HyperCalc applications greatly simplifies and increases the accuracy of performing unit conversions.





## SUMMARY AND CONCLUSIONS

Technical reference manuals are important engineering tools that provide information necessary to perform design procedures and calculations correctly and accurately. Reference materials can be easier to access and become more useful when presented in a computer-based format. Computer-based reference manuals can be developed using tools intended for the development of rule-based decision support systems (DSS). By incorporating decision support systems, hypertext, multimedia, and other features into computer-based design manuals, very powerful and useful tools can be authored. These tools allow users to access more information more quickly than conventional hard copy manuals. They also offer the user the versatility of performing searches for desired information, running related programs or models, and performing important design calculations.

Decision support system development techniques were implemented in this project to create a computer-based manual. Chapter 9 of the American Association of State Highway and Transportation Officials (AASHTO) Model Drainage Manual titled "Culverts" and the United States Geological Survey Water-Resources Investigations Report 83-4129 titled "Methods for Estimating Peak Discharge and Flood Boundaries of Streams in Utah" were incorporated into this computerized manual referred to as the Drainage Library. The Drainage Library was implemented on IBM compatible microcomputers operating under Microsoft Windows. The following tasks were accomplished during the development of the Drainage Library:

- 1) A decision support system was developed for the design of culverts using the procedures and policies outlined in the AASHTO manual. This DSS was complemented with another DSS for estimating peak discharges and flood depths based on the methods described in the USGS manual.
- 2) Commercial hypertext authoring tools were analyzed to determine which tool best met desired needs for organizing the contents of the manual. Microsoft Multimedia Viewer was chosen and subsequently used to create the hypertext of the computer-based manual. Over 200 pages of text were converted to hypertext topics and integrated with the DSS.
- 3) The manual was directly linked with other computer programs, allowing the user to access these programs from within the drainage manual application. The Drainage Library was linked with the decision support systems, Federal Highway Administration drainage software, and commercial software products such as WordPerfect, Quattro Pro, and Auto Cad.
- 4) Multimedia enhancements were implemented into the manual. Pictures, video, and audio were added to the text of the manual to clarify concepts and increase the informational content of the manual.
- 5) Equation solvers were developed which simplify the performance of necessary calculations and allow the user to perform them directly within the manual. Fifteen equation solvers were added to section 9.5, "Design Equations," of the AASHTO manual. Fifty-nine equation solvers were used

to replace the design charts and nomographs found in Appendix D of the same manual. Six equation solvers were implemented into the USGS portion of the manual to aid the user in the calculation of peak discharges and flood depths.

- 6) A table of contents (browser) was created with hypertext jumps to provide easy access of desired information within the manual.
- 7) Search capabilities were added to allow the user to find desired topics or words and quickly access related information.
- 8) A unit conversion package, HyperCalc, was developed and linked with the Drainage Library to allow a user to effortlessly perform unit conversions and design calculations.
- 9) An easy-access glossary was organized which can be used to define unknown terms. Popup windows were incorporated within the main body of the manual to explain or define unclear concepts and terms.
- 10) Tables and graphs were displayed quickly and concisely.

The accomplishment of these tasks proves that computer-based manuals can be effectively authored to contain features and enhancements that make them very powerful reference materials and design tools. This was also confirmed by the evaluations performed by the Utah Department of Transportation professionals and the author of the Drainage Library. The tools and procedures used to develop the

Drainage Library can also be used to create tutorials and a wide variety of other useful applications.

## REFERENCES

- Baffaut, C., and J.W. Delleur. 1989. Expert system for calibrating SWMM. American Society of Civil Engineering, Journal of Water-Resources Planning and Management 1153:78-298.
- Blakemore, E.T., and K.L. Lindskov. 1983. Methods for estimating peak discharge and flood boundaries of streams in Utah. Water-Resources Investigations Report 83-4129. U.S. Geological Survey. Salt Lake City, Utah. 77 p.
- Borland International. 1990. Turbo C++: Getting started. Borland International, Inc., Scotts Valley, California. 268 p.
- Borland International. 1993. Borland object windows for C++: Programmer's guide. Borland International, Inc., Scotts Valley, California. 418 p.
- Brown, B.W., and R.A. Shelton. 1986. TVA's use of computers in water resource management. American Society of Civil Engineering, Journal of Water-Resources Planning and Management 112(3):409-418.
- Camara, A.S., C. da Silva, A. C. Rodrigues, J.M. Remedio, P.P. Castro, M.J. Soares de Oliverira, and T.F. Fernandes. 1990. Decision support systems for estuarine water-quality management. American Society of Civil Engineering, Journal of Water-Resources Planning and Management 116(3):417-430.
- Davis, Michael W. 1988. Applied decision support. Prentice Hall, Inc., Englewood Cliffs, New Jersey. 256 p.
- Davis, R.J., P.M. Nanninga, J. Biggins, and P. Laut. 1991. Prototype decision support system for analyzing impact of catchment policies. American Society of Civil Engineering, Journal of Water-Resources Planning and Management 117(4):399-414.
- EXSYS. 1988. EXSYS professional demo: Building an expert system. EXSYS Inc. 38 p.
- Geselbracht, J.J., and D.M. Johnston. 1988. Issues in rule base development. American Society of Civil Engineering, Journal of Water-Resources Planning and Management, 114(4):457-468.

- Gonzalez, A.J., and D.D. Dankel. 1993. The engineering of knowledge-based systems: Theory and practice. Prentice-Hall, Inc., Englewood Cliffs, New Jersey. 523 p.
- Grenney, W.J. 1992. Computer implementation of the cost allocation process for the Nile River Basin, Egypt. Final report, Irrigation support project for Asia and the Near East, U.S. Agency for International Development. Utah Water Research Laboratory, Logan, Utah. 172 p.
- Grenney, W.J., W.W. Wallace, and T. Senti. 1993. A decision support system to assist stakeholders evaluate a water resource project, p. 145-151. In knowledge based systems for civil and structural engineering. Civil-Comp Press. Edinburgh, Scotland.
- Hu, David. 1989. C/C++ for expert systems: Unleashes the power of artificial intelligence. Management Information Source, Inc., Portland, Oregon. 565 p.
- Johnson, L.E. 1986. Water resource management decision support systems. American Society of Civil Engineering, Journal of Water-Resources Planning and Management, 112(3):308-324.
- Keen, P.G.W., and M.S. Scott-Morton. 1978. Decision support systems, an organizational perspective. Addison-Wesley, Reading, Massachusetts. 264 p.
- Koch, R.W., and R.L. Allen. 1986. Decision support system for local water management. American Society of Civil Engineering, Journal of Water-Resources Planning and Management 112(3):527-540.
- Labadie, J.W., and C.H. Sullivan. 1986. Computerized decision support systems for water managers. American Society of Civil Engineering, Journal of Water-Resources Planning and Management 112(3):299-307.
- Liong, S.Y., W.T. Chan, and L.H. Lum. 1991. Knowledge-based system for SWMM runoff component calibration. American Society of Civil Engineering, Journal of Water-Resources Planning and Management 117(5):507-524.
- Maybury, M. T. 1994. Knowledge-based multimedia: The future of expert systems and multimedia. Expert Systems With Applications 7(3):387-396.
- Microsoft Corporation. 1993. Microsoft multimedia viewer: Authoring guide. Microsoft Corporation. 400 p.

- Microsoft Corporation. 1994. Microsoft developer network: Development library. Microsoft Corporation. 312 p.
- Narasimhalu, A.D. 1994. A framework for the integration of expert systems with multimedia technologies. *Expert Systems With Applications*, 17(3), No. 3: 427-439.
- Scott-Morton, M.S. 1971. Management decision systems: Computer-based support for decision making. Division of Research, Harvard University, Cambridge, Massachusetts. 216 p.
- Turban, Efraim 1990. Decision support and expert systems: Management support systems. Macmillan Publishing Company, New York. 846 p.
- Vanegas, J.A., and N.C. Baker. 1994. Multimedia in civil engineering. *Civil Engineering*. May:71-73.





## APPENDICES



Appendix A.  
Equations Implemented In  
Equation Solver



The equations, nomographs, charts and relationships which were implemented into equation solvers are listed below.

AASHTO Drainage Manual section 9.5

Total Head Loss (Equation 9.1)  
 Continuity (Equation 9.2)  
 Velocity Head (Equation 9.3)  
 Entrance Loss (Equation 9.4a)  
 Friction Loss (Equation 9.4b)  
 Exit Loss (Equation 9.4c)  
 Barrel Losses (Equation 9.5)  
 Total Energy (Equation 9.6)  
 Head Water - Flowing Full (Equation 9.7)  
 Head Water - Partly Full Flow (Equation 9.8)  
 Overtopping Flow Rate (Equation 9.9)  
 Culvert Length Formula (Equation 9.10)

HyperCalc

Continuity  
 Mannings  
 Hazen-Williams  
 Darcy-Weisbach  
 Hydraulic Radius  
 Friction Factor for Laminar Flow  
 Friction Factor for Turbulent Flow  
 Rational Method  
 Reynold's Number

AASHTO Drainage Manual Design Charts (Headwater Depth)

Chart #	Shape	Control Section	Material	Type
1	Circular Inlet		Concrete	
2	Circular	Inlet	Metal	
3	Circular	Inlet	Metal	Beveled Ring Control
Chart #	Shape	Control Section	Material	Type

4	Circular	Critical Depth		
5	Circular	Outlet	Concrete	n = 0.012
6	Circular	Outlet	Metal	n = 0.024
7	Circular	Outlet	Metal	n = 0.0328 - 0.0302
8	Box	Inlet	Concrete	
9	Box	Inlet	Concrete	Wingwalls 18° - 33.7°, 45°
10	Box	Inlet	Concrete	90° Headwall, Beveled Edges
11	Box	Inlet	Concrete	Skewed Headwalls, Beveled Edges
12	Box	Inlet	Concrete	Flared Wingwalls Normal and Skewed
13	Box	Inlet	Concrete	Offset Flared Wingwalls, Beveled Edges
14	Box	Critical Depth		
15	Box	Outlet	Concrete	n = 0.012
16	Box	Inlet	Metal	Rise/Span < 0.3
17	Box	Inlet	Metal	0.3 ≤ Rise/Span < 0.4
18	Box	Inlet	Metal	0.4 ≤ Rise/Span < 0.5
19	Box	Inlet	Metal	Rise/Span ≥ 0.5
20	Box	Critical Depth		Concrete Bottom
21	Box	Outlet	Metal	Concrete Bottom, Rise/Span < 0.3
22	Box	Outlet	Metal	Concrete Bottom, 0.3 ≤ Rise/Span < 0.4
23	Box	Outlet	Metal	Concrete Bottom, 0.4 ≤ Rise/Span < 0.5
24	Box	Outlet	Metal	Concrete Bottom, Rise/Span ≥ 0.5
25	Box	Outlet	Metal	Metal Bottom, Rise/Span < 0.3
26	Box	Outlet	Metal	Metal Bottom, 0.3 ≤ Rise/Span < 0.4
27	Box	Outlet	Metal	Metal Bottom, 0.3 ≤ Rise/Span < 0.4
28	Box	Outlet	Metal	Metal Bottom, 0.4 ≤ Rise/Span < 0.5
29	Elliptical	Inlet	Concrete	Horizontal
30	Elliptical	Inlet	Concrete	Vertical
31	Elliptical	Critical Depth	Concrete	Horizontal
32	Elliptical	Critical Depth	Concrete	Vertical
33	Elliptical	Outlet	Concrete	Horizontal and Vertical
34	Pipe Arch	Inlet	Metal	
35	Pipe Arch	Inlet	Metal	18 in. Corner Radius
36	Pipe Arch	Inlet	Metal	31 in. Corner Radius
Chart #	Shape	Control Section	Material	Type

37	Pipe Arch	Critical Depth		Standard
38	Pipe Arch	Critical Depth		Structural Plate
39	Pipe Arch	Outlet	Metal	$n = 0.024$
40	Pipe Arch	Outlet	Metal	18 in. Corner Radius
41	Arch	Inlet	Metal	$0.3 \leq \text{Rise/Span} < 0.4$
42	Arch	Inlet	Metal	$0.4 \leq \text{Rise/Span} < 0.5$
43	Arch	Inlet	Metal	$\text{Rise/Span} \geq 0.5$
44	Arch	Critical Depth		
45	Arch	Outlet	Metal	Concrete Bottom, $0.3 \leq \text{Rise/Span} < 0.4$
46	Arch	Outlet	Metal	Concrete Bottom, $0.4 \leq \text{Rise/Span} < 0.5$
47	Arch	Outlet	Metal	Concrete Bottom, $\text{Rise/Span} \geq 0.5$
48	Arch	Outlet	Metal	Earth Bottom, $0.3 \leq \text{Rise/Span} < 0.4$
49	Arch	Outlet	Metal	Earth Bottom, $0.4 \leq \text{Rise/Span} < 0.5$
50	Arch	Outlet	Metal	Earth Bottom, $\text{Rise/Span} \geq 0.5$
51	Long Span	Inlet	Metal	Circular or Elliptical
52	Long Span	Inlet	Metal	High and Low Profile Arch
53	Long Span	Critical Depth	Metal	Circular or Elliptical
54	Long Span	Critical Depth	Metal	High and Low Profile Arch
55	Circular	Inlet		Throat Control, Side Tapered
56	Circular	Inlet		Face Control, Side Tapered
57	Box	Inlet	Concrete	Throat Control, Side Tapered
58	Box	Inlet	Concrete	Face Control, Side Tapered
59	Box	Inlet	Concrete	Face Control, Slope Tapered

USGS Survey Methods for Estimating Peak Discharge and Flood Boundaries of  
Streams in Utah Peak Discharge and Flood Depths for the following regions

Northern Mountains High Elevation  
 Northern Mountains Low Elevation  
 Uinta Basin  
 High Plateaus  
 Low Plateaus  
 Great Basin High Elevation

Appendix B.

Culvert Design Decision Support

System Rule File





## [Title] COMMAND

Set Master Layer: MASTER LAYER

Query Variable: DUMMY

Set Master Layer: MASTER LAYER1

Set a variables value: Ok To Change = False

Set a variables value: Change Q = False

## [Units] CASE BASED

Conditions: Unit Check

Case 0: Set a variables value: Units = 1

Case 1: Set a variables value: Units = 2

## [Check Reset Q] HUERISTIC

Conditions: Ok To Change

THEN: Continue

ELSE: Goto: FEMA

## [Reset Q] COMMAND

Set Variable Flags: Flow Check

Set Variable Flags: Enter/Calc

Set Variable Flags: Flow Enter

Set Variable Flags: Flood Region

Set Variable Flags: Map

Set Variable Flags: North Mount

Set Variable Flags: Great Basin North

Set Variable Flags: Great Basin South

Set Variable Flags: No Region

Set Variable Flags: Flow

Set Variable Flags: Region Elev.

Set Variable Flags: Region Area

Set Variable Flags: FlowDLL

Set Variable Flags: Flow Print

Set Variable Flags: Qd1

Set variable flags: Head Inlet

Set variable flags: Head Outlet

Set variable flags: Tail Water

Set variable flags: Critical Depth

Set variable flags: Outlet Velocity

Set variable flags: Flow Area

## [FEMA] HUERISTIC

Conditions: FEMA

THEN: Set a variables value: Qd = 100

ELSE: Continue:

[Road Type] CASE BASED

Conditions: Road Type

Case 0: Set a variables value:  $Qd = 50$

Case 1: Set a variables value:  $Qd = 5$

Case 2: Goto: Qd2

Case 3: Set a variables value:  $Qd = 25$

Case 4: Set a variables value:  $Qd = 10$

[Qd Reset] HUERISTIC

Conditions: FEMA

THEN: Set a variables value:  $Qd = 100$

ELSE: Goto: Flow Check

[FEMA Flow Check] HUERISTIC

Conditions: FEMA, FEMA Flow Check

THEN: Goto: Enter/Calc

ELSE: Goto: Qd2

[Flow Check] HUERISTIC

Conditions: Flow Check

THEN: Goto: Enter/Calc

ELSE: Continue

[Qd2] CASE BASED

Conditions: Qd1

Case 0: Set a variables value:  $Qd = 10$

Case 1: Set a variables value:  $Qd = 100$

Case 2: Set a variables value:  $Qd = 2$

Case 3: Set a variables value:  $Qd = 25$

Case 4: Set a variables value:  $Qd = 5$

Case 5: Set a variables value:  $Qd = 50$

[Enter/Calc] HUERISTIC

Conditions: Enter/Calc

THEN: Continue

ELSE: Goto: USGS

[USGS] CASE BASED

Conditions: Flood Region

Case 0: Execute Activity: GBH Region

Case 1: Execute Activity: GBL Region

Case 2: Execute Activity: HP Region  
 Case 3: Execute Activity: LP Region  
 Case 4: Continue  
 Case 5: Execute Activity: NMH Region  
 Case 6: Execute Activity: NML Region  
 Case 7: Execute Activity: U Region

[Map] CASE BASED

Conditions: Map

Case 0: Continue

Case 1: Execute Activity: U Region  
 Case 2: Execute Activity: HP Region  
 Case 3: Execute Activity: LP Region  
 Case 4: Execute Activity: LP Region  
 Case 5: Execute Activity: LP Region  
 Case 6: Goto: GBN Region  
 Case 7: Goto: GBN Region  
 Case 8: Goto: GBS Region  
 Case 9: Continue

[North Mnt Det.] CASE BASED

Conditions: North Mount

Case 0: Execute Activity: NMH Region  
 Case 1: Execute Activity: NML Region

[GBN Region] HUERISTIC

Conditions: Great Basin North

THEN: Execute Activity: GBH Region  
 ELSE: Goto: No Region 6

[GBS Region] HUERISTIC

Conditions: Great Basin South

THEN: Execute Activity: GBH Region  
 ELSE: Continue

[No Region 6] CASE BASED

Conditions: No Region

Case 0: Continue  
 Case 1: Execute Activity: LP Region  
 Case 2: Quit

[Flow Enter2] HUERISTIC

Conditions: Flow

104

THEN: Goto: Flow Print  
ELSE: Goto: Flow Print

[Get Area/Elev] HUERISTIC  
Conditions: Region Elev, Region Area  
THEN: Continue  
ELSE: Continue

[Get Flow DLL] HUERISTIC  
Conditions: FlowDLL  
THEN: Set variables equal: Flow = FlowDLL  
ELSE: Set variables equal: Flow = FlowDLL

[Flow Print] CASE BASED  
Conditions: Flow Print  
Case 0: Goto: Check Reset Emb  
Case 1: Goto: Reset Q  
Case 2: Continue:

[Round] HUERISTIC  
Conditions: Round  
THEN: Set variables equal: Flow = Round  
ELSE: Set variables equal: Flow = Round

[Reset Emb] COMMAND  
Set Variable Flags: Elev1  
Set Variable Flags: Length  
Set Variable Flags: Slope  
Set variable flags: Tail Water  
Set variable flags: Cover  
Set variable flags: Overflow  
Set variable flags: OverRoad  
Set variable flags: Head Inlet  
Set variable flags: Head Outlet  
Set variable flags: Outlet Velocity  
Set variable flags: Control  
Set variable flags: Check  
Set variable flags: Ht

[Emb Data] HUERISTIC  
Conditions: Elev1  
THEN: Continue:  
ELSE: Continue:

[US Data] HUERISTIC

Conditions: Length

THEN: Continue

ELSE: Continue

[Negative Slope] HUERISTIC

Conditions: Slope

THEN: Continue

ELSE: Goto: Check Reset Channel

[Negative Slope1] CASE BASED Conditions: Negative Slope Case

Case 0: Quit

Case 1: Goto: Reset Emb

[Check Reset Channel] HUERISTIC

Conditions: Ok To Change, Change

THEN: Continue

ELSE: Goto: Channel Data

[Reset Channel] COMMAND

Set variable flags: EL1

Set variable flags: Channel n

Set variable flags: Channel Slope

Set variable flags: Fix Tail Water

Set variable flags: Tail Water

Set variable flags: Outlet Velocity

Set variable flags: Control

Set variable flags: Check

Set variable flags: Overflow

Set variable flags: OverRoad

Set variable flags: Ht

[Channel Data] HUERISTIC

Conditions: EL1

THEN: Continue:

ELSE: Continue:

[Channel n and S] HUERISTIC

Conditions: Channel n, Channel Slope

THEN: Continue

ELSE: Continue

[Tail Water Check] HUERISTIC  
 Conditions: Tail Water  
 THEN: Continue  
 ELSE: Goto: Check Reset General

[Overflow] CASE BASED  
 Conditions: Fix Tail Water  
 Case 0: Goto: Reset Channel  
 Case 1: Quit

[Check Reset General] HUERISTIC  
 Conditions: Ok To Change, Change  
 THEN: Continue  
 ELSE: Goto: Check Reset General1

[Reset General] COMMAND  
 Set variable flags: Material Test  
 Set variable flags: Critical Depth  
 Set variable flags: Head Inlet  
 Set variable flags: Head Outlet  
 Set variable flags: Critical Depth  
 Set variable flags: Cover  
 Set variable flags: Outlet Velocity  
 Set variable flags: Control  
 Set variable flags: Check  
 Set variable flags: Outlet Velocity  
 Set variable flags: Control  
 Set variable flags: Check  
 Set variable flags: Overflow  
 Set variable flags: OverRoad  
 Set variable flags: Ht  
 Set variable flags: Diameter  
 Set variable flags: Material

[Check Reset General1] HUERISTIC  
 Conditions: Ok To Change, Change  
 THEN: Continue  
 ELSE: Goto: Culvert Charact.

[Reset General1] COMMAND  
 Set variable flags: Entrance Loss  
 Set variable flags: CM Inlet Type  
 Set variable flags: Conc Inlet Type

Set variable flags: Check n  
 Set variable flags: Critical Depth  
 Set variable flags: Enter n  
 Set variable flags: Inlet Type  
 Set variable flags: Head Inlet  
 Set variable flags: Head Outlet  
 Set variable flags: Critical Depth  
 Set variable flags: Cover  
 Set variable flags: Outlet Velocity  
 Set variable flags: Control  
 Set variable flags: Check  
 Set variable flags: Outlet Velocity  
 Set variable flags: Control  
 Set variable flags: Check  
 Set variable flags: Overflow  
 Set variable flags: OverRoad  
 Set variable flags: Ht

[Culvert Charact.] HUERISTIC  
 Conditions: Diameter, Entrance Loss  
 THEN: Continue:  
 ELSE: Continue:

[Charact Cont.] HUERISTIC  
 Conditions: Material Test  
 THEN: Execute Activity: Concrete  
 ELSE: Execute Activity: CM

[CM Inlet Name] CASE BASED  
 Conditions: CM Inlet Type  
 Case 0: Set a variables value: Inlet Name Print = Headwall  
 Case 1: Set a variables value: Inlet Type Print = Mitered to Conform to Slope  
 Case 2: Set a variables value: Inlet Type Print = Projecting

[CM Culvert Char] CASE BASED  
 Conditions: CM Inlet Type  
 Case 0: Execute Activity: Inlet1  
 Case 1: Execute Activity: Inlet2  
 Case 2: Execute Activity: Inlet3

[Conc Inlet Name] CASE BASED  
 Conditions: Conc Inlet Type  
 Case 0: Set a variables value: Inlet Type Print = Square Edge with Headwall



Case 1: Set a variables value: Inlet Type Print = Groove End with Headwall  
Case 2: Set a variables value: Inlet Type Print = Groove End Projecting

[Conc Culvert Char] CASE BASED

Conditions: Conc Inlet Type

Case 0: Execute Activity: Inlet1

Case 1: Execute Activity: Inlet2

Case 2: Execute Activity: Inlet3

[Critical Depth] HUERISTIC

Conditions: Critical Depth

THEN: Continue

ELSE: Continue

[Head Inlet Data] HUERISTIC

Conditions: Head Inlet

THEN: Continue

ELSE: Continue

[Head Outlet Data] HUERISTIC

Conditions: Head Outlet

THEN: Continue

ELSE: Continue

[Control] HUERISTIC

Conditions: Control

THEN: Set a variables value: Control Name = INLET Control

ELSE: Set a variables value: Control Name = OUTLET Control

[Check1] HUERISTIC

Conditions: Check

THEN: Message Box

ELSE: Continue

[Check2] HUERISTIC

Conditions: Check

THEN: Message Box

ELSE: Continue

[Overflow Check] HUERISTIC

Conditions: Overflow

THEN: Goto: Outlet Velocity

ELSE: Continue

[Surface Type] CASE BASED  
 Conditions: Surface Check  
 Case 0: Set a variables value: Surface = 2  
 Case 1: Set a variables value: Surface = 1

[Ht] HUERISTIC  
 Conditions: Ht  
 THEN: Continue  
 ELSE: Continue

[Overflow Calculation] HUERISTIC  
 Conditions: OverRoad  
 THEN: Set variables equal: Overflow = OverRoad  
 ELSE: Set variables equal: Overflow = OverRoad

[Outlet Velocity] HUERISTIC  
 Conditions: Outlet Velocity, Flow Area  
 THEN: Continue  
 ELSE: Continue

[Cover] HUERISTIC  
 Conditions: Cover  
 THEN: Continue  
 ELSE: Continue

[Reset Change] COMMAND  
 Set variable flags: Change  
 Set variable flags: Print English  
 Set variable flags: Print Metric

C [Print Unit Check] HUERISTIC  
 Conditions: Units  
 THEN: Continue:  
 ELSE: Goto: Print Metric

[Print English] COMMAND  
 Set variable flags: Print English  
 Query Variable: Print English  
 Goto: Change

[Print Metric] COMMAND  
 Set variable flags: Print Metric  
 Query Variable: Print Metric

110

[Change] CASE BASED

Conditions: Change

Case 0: Quit:

Case 1: Continue

Case 2: Continue

[Reset Print] COMMAND

Set variable flags: Cover Print

Set variable flags: Head Print

Set a variables value: OK To Change = True

[Change1] CASE BASED

Conditions: Change

Case 0: Goto: Reset Q

Case 1: Goto: Reset Emb

Case 2: Goto: Reset Channel

Case 3: Goto: Reset General

Case 4: Goto: Reset General1

Case 5: Continue

[Junk] HUERISTIC

Conditions: Unit Check

THEN: Execute Activity:

ELSE: Continue

[Startup] COMMAND

Set Master Layer: MASTER LAYER

Appendix C.  
Culvert Design Decision Support  
System Variable File



**[DUMMY] PRESENTATION**

Query Method: Flexpert Query Engine (Default)

**[Elev1] REAL**

Query Method: User Defined Method

Function: Query Variable, DLL: culvert.DLL

: Embankment Cross Sectional Data

**[Elev2] REAL**

Query Method: Flexpert Query Engine (Default)

: Embankment Cross Sectional Data

**[Elev3] REAL**

Query Method: Flexpert Query Engine (Default)

: Embankment Cross Sectional Data

**[Elev4] REAL**

Query Method: Flexpert Query Engine (Default)

: Embankment Cross Sectional Data

**[Sta1] REAL**

Query Method: Flexpert Query Engine (Default)

: Embankment Cross Sectional Data

**[Sta2] REAL**

Query Method: Flexpert Query Engine (Default)

: Embankment Cross Sectional Data

**[Sta3] REAL**

Query Method: Flexpert Query Engine (Default)

: Embankment Cross Sectional Data

**[Sta4] REAL**

Query Method: Flexpert Query Engine (Default)

: Embankment Cross Sectional Data

**[ST1] REAL**

Query Method: Flexpert Query Engine (Default)

: Downstream Channel Cross Sectional Data

**[ST2] REAL**

Query Method: Flexpert Query Engine (Default)

: Downstream Channel Cross Sectional Data

[ST3] REAL

Query Method: Flexpert Query Engine (Default)  
: Downstream Channel Cross Sectional Data

[ST4] REAL

Query Method: Flexpert Query Engine (Default)  
: Downstream Channel Cross Sectional Data

[EL4] REAL

Query Method: Flexpert Query Engine (Default)  
: Downstream Channel Cross Sectional Data

[EL3] REAL

Query Method: Flexpert Query Engine (Default)  
: Downstream Channel Cross Sectional Data

[EL2] REAL

Query Method: Flexpert Query Engine (Default)  
: Downstream Channel Cross Sectional Data

[EL1] REAL

Query Method: User Defined Method - Function: Query Variable, DLL: culvert  
DLL  
: Downstream Channel Cross Sectional Data

[Ok To Change] LOGICAL

Query Method: Flexpert Query Engine (Default)  
Static Text: Ok To Change?

[Units] INTEGER

Query Method: Flexpert Query Engine (Default)  
: Variable Units used in DLL's

[Unit Check] SINGLE

Query Method: Flexpert Query Engine (Default)  
Static Text: Would you like to work in English or Metric units?  
Gray Shade

[FEMA] LOGICAL

Query Method: Flexpert Query Engine (Default)  
Static Text: Is the area where you wish to place the culvert regulated by FEMA (The  
Federal Emergency Management Agency)?  
Gray Shade

[Data Check] MULTI

Query Method: Flexpert Query Engine (Default)

Static Text: The following data is necessary for the culvert design process.

Choose one of the selections. navycSOhna5dter] LOGICAL

Query Method: Flexpert Query Engine (Default)

Static Text: Have all of the environmental constraints outlined in the AASHTO

Model Drainage Manual been met?

[Flood Region] SINGLE

Query Method: Flexpert Query Engine (Default)

Which USGS region is the culvert going to be placed within?

Gray Shade

[Map] VISUAL

Query Method: Flexpert Query Engine (Default)

Static Text: (To scroll within the map, simply hold down the left mouse button and move the mouse )

: Place the mouse pointer over the approximate location where the culvert will be placed and double click the left mouse button.

[Great Basin North] LOGICAL

Query Method: Flexpert Query Engine (Default)

Static Text: Is the mean basin elevation greater than 6000 feet and the study site datum greater than 5000 feet?

Gray Shade

[Great Basin South] LOGICAL

Query Method: Flexpert Query Engine (Default)

Static Text: Is the mean basin elevation greater than 8000 feet and the study site datum greater than 7000 feet?

Gray Shade

[North Mount] SINGLE

Query Method: Flexpert Query Engine (Default)

Static Text: The approximate elevation of the site is:

Gray Shade

[No Region] SINGLE

Query Method: Flexpert Query Engine (Default)

Regression equations were not developed for the Great Basin Low Elevation Subregion (GBLER) do to the extremely variable flood characteristics



[Region Elev] REAL

Query Method: Flexpert Query Engine (Default)

Static Text: Please enter the mean basin elevation of the site (in thousands of feet or meters)

Gray Shade

[Round] REAL

Query Method: Flexpert Query Engine (Default)

Static Text w/ Embedded Values: The calculated [Qd] year design flow is [Flow] for the [Region Name] Region. What would you like to round this value to?

[Region Area]REAL

Query Method: Flexpert Query Engine (Default)

Static Text: Please enter the drainage area (in square miles or kilometers).

Gray Shade

[FlowDLL] REAL

Query Method: User Defined Method - Function: Query Variable, DLL:  
culvert.DLL

: Get USGS Flow Calculations from DLL

[Tail Water] REAL

Query Method: User Defined Method - Function: Query Variable, DLL:  
CULVERT.DLL

: Determine tail water - My DLL

[Fix Tail Water] SINGLE

Query Method: Flexpert Query Engine (Default)

: The design flow used in this problem will exceed the banks of the downstream channel which was entered. Choose one of the following options.

[HWoi] REAL

Query Method: User Defined Method - Function: Query Variable, DLL: Culvert  
DLL

: Contains the value calculated for HWoi in the DLL

[Control]INTEGER

Query Method: User Defined Method - Function: Query Variable, DLL: Culvert  
DLL

: Contains value which tells whether inlet or outlet is the controlling headwater.

1 = Inlet

[Control Name] STRING

Query Method: User Defined Method - Function: Query Variable, DLL: Culvert  
DLL

: Tells whether inlet or outlet is the controlling headwater.

[Head Control] REAL

Query Method: User Defined Method - Function: Query Variable, DLL: Culvert  
DLL

: Controlling Headwater Value - My DLL

[Head Inlet] REAL

Query Method: User Defined Method - Function: Query Variable, DLL: Culvert  
DLL

: Determine inlet control headwater depth - My DLL

[Slope] REAL

Query Method: User Defined Method - Function: Query Variable, DLL: Culvert  
DLL

: Determine slope

[Length] REAL

Query Method: User Defined Method - Function: Query Variable, DLL: Culvert  
DLL

: Determine Length

[Negative Slope] SINGLE

Query Method: Flexpert Query Engine (Default)

Static Text w/ Embedded Values: A slope of [Slope] was calculated for the culvert using the cross sectional embankment entered. The slope of the Gray Shade

[Material] INTEGER

Query Method: Flexpert Query Engine (Default)

: Material

1 = Concrete

2 = C.M.

.

[Flow Area] REAL

Query Method: User Defined Method - Function: Query Variable, DLL:  
culvert.DLL

Static Text: Flow Area Calculation - culvert DLL.

**[Check] INTEGER**

Query Method: User Defined Method - Function: Query Variable, DLL:

culvert.DLL

: Checks to make sure assumptions used to determine controlling headwater are met

**[Cover] REAL**

Query Method: User Defined Method - Function: Query Variable, DLL:

culvert.DLL

Static Text: Cover Calculation - culvert.DLL.

**[Outlet Velocity] REAL**

Query Method: User Defined Method - Function: Query Variable, DLL:

culvert.DLL

Static Text: Outlet Velocity Calculated - culvert.DLL

**[Overflow] REAL**

Query Method: User Defined Method - Function: Query Variable, DLL: culvert

DLL

Static Text: Determines whether flow over the roadway (Qr) exists. - Culvert DLL

**[OverRoad] REAL**

Query Method: User Defined Method - Function: Query Variable, DLL:

culvert.DLL

Static Text: Over the road flow value-Culvert DLL

**[Surface] INTEGER**

Query Method: Flexpert Query Engine (Default)

: Roadway Material

1 = Paved

2 = Gravel

**[Surface Check] SINGLE**

Query Method: Flexpert Query Engine (Default)

: In order to compute the discharge over the roadway (Qr), the roadway surface must be known.

**[Hwr] REAL**

Query Method: Flexpert Query Engine (Default)

**[Ht] REAL**

Query Method: Flexpert Query Engine (Default)

Static Text: t

Static Text: h

Gray Shade:  
 Bitmap: CHART60.BMP  
 : Please enter the value of ht.  
 (HW<sub>r</sub> = [HW<sub>r</sub>])  
 Gray Shade:

[Inlet Type Print] STRING  
 Query Method: Flexpert Query Engine (Default) Static Text  
 : Inlet type name, too be printed.

[Material Test] SINGLE  
 Query Method: Flexpert Query Engine (Default)  
 What type of material do you wish to design the culvert for?

[Flow Print] SINGLE  
 Query Method: Flexpert Query Engine (Default)  
 Static Text w/ Embedded Values: The calculated [Qd] year design flow is [Flow] for  
 the [Region Name] Region.

[Print Metric] PRESENTATION  
 Query Method: Flexpert Query Engine (Default)  
 Static Text: Output:  
 Static Text w/ Embedded Values: [Qd]  
 $Q = [\text{Flow}] \text{ cms}$   
 $K_e = [\text{Entrance Loss}]$   
 Manning's n = [Mannings]  
 Culvert Diameter = [Diameter] m  
 Culvert Length = [Length] m  
 Inlet Type = Static Text w/ Embedded Values: Tail Water = [Tail Water] m  
 Headwater:  
     Inlet Control (HW<sub>i</sub>) = [Head Inlet] m  
     Outlet Control (HW<sub>oi</sub>) = {Head Outlet} m  
     Controlling Headwater = [Head Control]

[Print English] PRESENTATION  
 Query Method: Flexpert Query Engine (Default)  
 Static Text w/ Embedded Values: [Qd]  
 Static Text w/ Embedded Values: Outlet Velocity = [Outlet Velocity] ft/s  
 $\text{Flow Area} = [\text{Flow Area}] \text{ ft}^2$   
 Cover = [Cover] ft  
 Overflow = [Overflow]  
 $Q = [\text{Flow}] \text{ cfs}$   
 $K_e = [\text{Entrance Loss}]$

120

Manning's n = [Mannings]

Culvert Diameter = [Diameter] ft

Appendix D.  
Culvert Design Decision Support  
System Activity File



**FEMA1**

Set a variables value:  $QD = 100$

Goto: Check Reset Q

**FEMA**

Message Box: Backwater due to the constriction must be held to 1 foot for the 100 year Q Backwater surface profiles can be determined using the Wespro

**No Channel Data**

Message Box: The downstream channel cross section must be obtained before the culvert can be designed.

**No Emb Data**

Message Box: The cross sectional embankment data must be obtained before continuing with the design process.

Quit

**No Emb/Chan Check**

Message Box: The down stream channel cross sectional data must be obtained before continuing with the design process.

Message Box: The roadway cross sectional embankment data must be obtained before continuing with the design process.

Quit

**LP Region**

Set a variables value: Region = 4

Set a variables value: Region Name = Low Plateaus

Goto: Get Area/Elev

**HP Region**

Set a variables value: Region = 3

Set a variables value: Region Name = High Plateaus

Goto: Get Area/Elev

**U Region**

Set a variables value: Region = 2

Set a variables value: Region Name = Uinta Basin

Goto: Get Area/Elev

**NMH Region**

Set a variables value: Region = 0

Set a variables value: Region Name = Northern Mountains High Elev.

Goto: Get Area/Elev



124

NML Region

Set a variables value: Region = 1

Set a variables value: Region Name = Northern Mountains Low Elev.

Goto: Get Area/Elev

GBL Region

Set a variables value: Region = 6

Set a variables value: Region Name = Great Basin Low Elev.

Goto: No Region 6

GBH Region

Set a variables value: Region = 5

Set a variables value: Region Name = Great Basin High Elev.

Goto: Get Area/Elev

Concrete

Set a variables value: Material = 1

Set a variables value: Mannings = 0 012

Goto: Conc Inlet Name

Overflow

Message Box: The calculated tailwater exceeds the banks of the entered downstream channel.

Quit

CM

Set a variables value: Mannings = 0 024

Set a variables value: Material = 2

Continue

Inlet1

Set a variables value: Inlet Type = 1

Goto: Critical Depth

Inlet2

Set a variables value: Inlet Type = 2

Goto: Critical Depth

Inlet3

Set a variables value: Inlet Type = 3

Goto: Critical Depth

BackUp

Backup: 1 variable(s)

Set Master Layer: MASTER LAYER1



Appendix E.  
Equation Solver C Code



EQUATN.CPP

```

//-----
// ObjectWindows - (C) Copyright 1991, 1993 by Borland International
//-----
#include <owl\owlpch.h>
#pragma hdrstop
#include <owl\window.h>
#include "equatn.h"
#include <stdlib.h>

#include "eqat.rh"           // Definition of all resources.
#include "nfuncdlg.h"

#include "charts.h"
#include "eqcalc.h"
#include "eqdefs.cpp"

#define MaxDlg 5

static TModule *pM, *MainMod;
static TWindow *pW;
static NFuncDlg *Dlg[MaxDlg];
static int LDlg = 0;

int far pascal LibMain(HINSTANCE ins,WORD ,WORD ,LPSTR CmdLine){
    if (!pM) {
        pM = new TModule(0,ins,CmdLine);
#ifdef __WIN32__
        MainMod = new TModule("BWCC32.DLL", TRUE);
#else
        MainMod = new TModule("BWCC.DLL", TRUE);
#endif
        BOOL FAR PASCAL(*bwccIntlInit)(UINT);
        (FARPROC)bwccIntlInit = MainMod->GetProcAddress("BWCCIntlInit");
        if (bwccIntlInit)
            bwccIntlInit(0);
        BOOL FAR PASCAL(*bwccRegister)(HINSTANCE);
        (FARPROC)bwccRegister = MainMod-
> GetProcAddress("BWCCRegister");
        if (bwccRegister)
            bwccRegister(pM->GetInstance());
    }
    return 1;
}

```

```

void pascal far _export EQUATIONDLG(HWND pHW, char far *who){
    if(!pW){
        TWindow *pW = GetWindowPtr(pHW);// check if an OWL window
        if (!pW)
            pW = new TWindow(pHW, pM);
    }
    if(Dlg[LDlg]){
        Dlg[LDlg]->CloseWindow();
        delete Dlg[LDlg];
    }
    if(who[0] == 'C' || who[0] == 'c'){
        char tmp[10];
        for(int i = 5,j = 0; who[i] != '\0';i++,j++){
            tmp[j] = who[i];
        }
        tmp[j] = '\0';
        i = atoi(tmp);
        switch(i){
            case(1): SHOWCHART1(pM, Dlg[LDlg], pW); break;
            case(2): SHOWCHART2(pM, Dlg[LDlg], pW); break;
            case(3): SHOWCHART3(pM, Dlg[LDlg], pW); break;
            case(4): SHOWCHART4(pM, Dlg[LDlg], pW); break;
            case(5): SHOWCHART5(pM, Dlg[LDlg], pW); break;
            case(6): SHOWCHART6(pM, Dlg[LDlg], pW); break;
            case(7): SHOWCHART7(pM, Dlg[LDlg], pW); break;
            case(8): SHOWCHART8(pM, Dlg[LDlg], pW); break;
            case(9): SHOWCHART9(pM, Dlg[LDlg], pW); break;
            case(10): SHOWCHART10(pM, Dlg[LDlg], pW); break;
            case(11): SHOWCHART11(pM, Dlg[LDlg], pW); break;
            case(12): SHOWCHART12(pM, Dlg[LDlg], pW); break;
            case(13): SHOWCHART13(pM, Dlg[LDlg], pW); break;
            case(14): SHOWCHART14(pM, Dlg[LDlg], pW); break;
            case(15): SHOWCHART15(pM, Dlg[LDlg], pW); break;
            case(16): SHOWCHART16(pM, Dlg[LDlg], pW); break;
            case(17): SHOWCHART17(pM, Dlg[LDlg], pW); break;
            case(18): SHOWCHART18(pM, Dlg[LDlg], pW); break;
            case(19): SHOWCHART19(pM, Dlg[LDlg], pW); break;
            case(20): SHOWCHART20(pM, Dlg[LDlg], pW); break;
            case(29): SHOWCHART29(pM, Dlg[LDlg], pW); break;
            case(30): SHOWCHART30(pM, Dlg[LDlg], pW); break;
            case(31): SHOWCHART31(pM, Dlg[LDlg], pW); break;
            case(32): SHOWCHART32(pM, Dlg[LDlg], pW); break;
            case(34): SHOWCHART34(pM, Dlg[LDlg], pW); break;
            case(35): SHOWCHART35(pM, Dlg[LDlg], pW); break;
            case(36): SHOWCHART36(pM, Dlg[LDlg], pW); break;
            case(37): SHOWCHART37(pM, Dlg[LDlg], pW); break;
            case(38): SHOWCHART38(pM, Dlg[LDlg], pW); break;
            case(41): SHOWCHART41(pM, Dlg[LDlg], pW); break;
        }
    }
}

```

```

        case(42): SHOWCHART42(pM, Dlg[LDlg], pW); break;
        case(43): SHOWCHART43(pM, Dlg[LDlg], pW); break;
        case(44): SHOWCHART44(pM, Dlg[LDlg], pW); break;
        case(51): SHOWCHART51(pM, Dlg[LDlg], pW); break;
        case(52): SHOWCHART52(pM, Dlg[LDlg], pW); break;
        case(53): SHOWCHART53(pM, Dlg[LDlg], pW); break;
        case(54): SHOWCHART54(pM, Dlg[LDlg], pW); break;
        case(55): SHOWCHART55(pM, Dlg[LDlg], pW); break;
        case(56): SHOWCHART56(pM, Dlg[LDlg], pW); break;
        case(57): SHOWCHART57(pM, Dlg[LDlg], pW); break;
        case(58): SHOWCHART58(pM, Dlg[LDlg], pW); break;
        case(59): SHOWCHART59(pM, Dlg[LDlg], pW); break;
        default: SHOWCHART21(pM, Dlg[LDlg], pW,i);
    }
} else if(who[0] == 'E' || who[0] == 'e'){
    char tmp[10];
    for(int i = 2,j = 0; who[i] != '\0' && who[i] >= '0' && who[i] <=
'9';i++,j++){
        tmp[j] = who[i];
        tmp[j] = '\0';
        int chpt = atoi(tmp);
        for(i++,j = 0; who[i] != '\0' && who[i] >= '0' && who[i] <=
'9';i++,j++){
            tmp[j] = who[i];
            tmp[j] = '\0';
            int eqn = atoi(tmp);
            if(who[i] != '\0')
                i = toupper(who[i])-'A';
            if(chpt == 9)
                switch(eqn){
                    case(1): SHOWEQ9_1(pM, Dlg[LDlg], pW); break;
                    case(2): SHOWEQ9_2(pM, Dlg[LDlg], pW); break;
                    case(3): SHOWEQ9_3(pM, Dlg[LDlg], pW); break;
                    case(4): switch(i){

                        case(0): SHOWEQ9_4A(pM, Dlg[LDlg], pW); break;
                        case(1): SHOWEQ9_4B(pM, Dlg[LDlg], pW); break;
                        case(2): SHOWEQ9_4C(pM, Dlg[LDlg], pW); break;
                        default: SHOWEQ9_4D(pM, Dlg[LDlg], pW); break;
                    } break;

                    case(5): SHOWEQ9_5(pM, Dlg[LDlg], pW); break;
                    case(6): SHOWEQ9_6(pM, Dlg[LDlg], pW); break;
                    case(7): SHOWEQ9_7(pM, Dlg[LDlg], pW); break;
                    case(8): SHOWEQ9_8(pM, Dlg[LDlg], pW); break;
                    case(9): SHOWEQ9_9(pM, Dlg[LDlg], pW); break;
                    default: SHOWEQ9_10(pM, Dlg[LDlg], pW); break;
                }
            }

```



132

```
    }else if(who[0] == 'F' || who[0] == 'f'){
        char tmp[10];
        for(int i = 4,j = 0; who[i] != '\0' && who[i] >= '0' && who[i] <=
'9';i++,j++)
            tmp[j] = who[i];
        tmp[j] = '\0';
        i = atoi(tmp);
        SHOWFLOW(pM, Dlg[LDlg], pW,i);
    }
    if(++LDlg >= MaxDlg)
        LDlg = 0;
}
```

EQDEFS.CPP

```

#include "eqat.rh"          // Definition of all resources.
#include "nfuncdlg.h"

#include "charts.h"
#include "eqcalc.h"

void SHOWEQ9_4B(TModule *pM, NFuncDlg *Dlg, TWindow *pA){
    Dlg = new NFuncDlg(pA,IDD_BASE,pM);
    Dlg->SetCaption("EQ9-4B");
    Dlg->setFunc(calcE9_4B);
    Dlg->AddVar(0,0,5,505);
    Dlg->AddVar("Hf = Friction loss (ft)","Hf = Friction loss (m)");
    Dlg->AddVar("L = Length of culvert barrel (ft)","L = Length of culvert barrel
(m)");
    Dlg->AddVar("R = Hydraulic radius (ft)","R = Hydraulic radius (m)");
    Dlg->AddVar("V = Velocity (ft/sec)","V = Velocity (m/sec)");
    Dlg->AddVar("n = Manning's Rougness Coefficient","n = Manning's Rougness
Coefficient",4);
    Dlg->AddCombo("Concrete Pipe      Smooth walls      0.020-0.13",0.0115);
    Dlg->AddCombo("Concrete Boxes     Smooth walls      0.012-0.15",0.0135);
    Dlg->AddCombo("Corrugated Metal    2 2/3 by 1/2\"      0.022-
0.027",0.0245);
    Dlg->AddCombo("Pipes and Boxes     2 2/3 by 1/2\"      0.022-
0.027",0.0245);
    Dlg->AddCombo("Annular or Helical  6x1\" corrugation    0.022-
0.026",0.0235);
    Dlg->AddCombo(" pipes and boxes    5x1\" corrugation    0.027-
0.028",0.0255);
    Dlg->AddCombo("(n varies by barrel 3x1\" corrugation    0.027-
0.028",0.0275);
    Dlg->AddCombo(" size)             6x2\" corrugation    0.033-0.035",0.034);
    Dlg->AddCombo("                   9x2\" corrugation    0.033-0.037",0.035);
    Dlg->AddCombo("Corrugated Metal    2 2/3x1/2\" corrugation 0.012-
0.024",0.018);
    Dlg->AddCombo("Spiral Rib Metal    Smooth walls      0.012-
0.013",0.0125);
    Dlg->Make();
}

```

NFUNC.DLG.CPP

```
/* Project test
```

```
    Copyright © 1993. All Rights Reserved.
```

```
    SUBSYSTEM:  test.exe Application
    FILE:       nfuncDlg.cpp
    AUTHOR:     Newell Crookston
```

## OVERVIEW

```
    =====
```

```
    Source file for implementation of NFuncDlg (TDialog).
```

```
*/
```

```
#include <owl\owlpch.h>
#pragma hdrstop
```

```
#include "nfuncdlg.h"
#include "Ctl3d.cpp"
```

```
// Vstatic
```

```
//-----
```

```
void VStatic::DispText(float f){
    size_t Len = Text.GetItemsInContainer();
    for(size_t i = 0; i < Len; i++)
        if(f >= Value1[i] && f <= Value2[i])
            SetText(Text[i].c_str());
}
```

```
void VStatic::AddText(const char far *t, float V1, float Vh){
    Text.Add(t);
    Value1.Add(V1);
    Value2.Add(Vh);
}
```

```
void VStatic::SetValue(int pos,const char far *t, float V1, float Vh){
    if(pos < Text.GetItemsInContainer()){
        Text[pos] = t;
        Value1[pos] = V1;
        Value2[pos] = Vh;
    }
}
```

```
// NumList function defs.
```

```

// * * * * *
NumList *NumList::addNum(NumList *L, float f){
    for(NumList *t = L; t->next != 0; t = t->next); // finds the last link
    t->next = new NumList(f, t->which+1); // adds a new link
    return t->next; // returns the new link
}

int NumList::getSize(){ // returns the number stored in the link
    int num = 0;
    for(NumList *t = next; t != 0; t = t->next, num++); // goes thru the links
    return num;
}

float NumList::getNum(int who){ // returns the number stored in the link
    if(which == who) //checks if the first link has the number
        return data;
    for(NumList *t = next; t != 0; t = t->next) // goes thru the links
        if(t->which == who) //checks if this link has the needed number
            return t->data;
    return 0;
}

void NumList::setNum(int who, float f){ //Puts a number into the given link
    if(which == who){
        data = f;
        return;
    }
    for (NumList *t = next; t != 0; t = t->next)
        if(t->which == who){
            t->data = f;
            return;
        }
}

//
// GBoxes function defs.
// * * * * *
GBoxes *GBoxes::addBox(GBoxes *GL, TWindow *p, TModule *m, int Id, const char far *t =
0, int x = -1, int y = 1, int w = -1, int h = -1){
    GBoxes *v, *nw = new GBoxes();
    if(x == -1)
        nw->GB = new NGroupBox(p, Id, m);
    else {
        nw->GB = new NGroupBox(p, Id, t, x, y, w, h, m);
        nw->setSize(x, y, x+w, y+h);
    }
}

```

```

        if(GL){
            for(v = GL; v->next != 0; v = v->next);
            v->next = nw;
        } else
            GL = nw;
    return nw;
}

NGroupBox *GBoxes::getGrp(int i){
    if(i == 1)
        return GB;
    GBoxes *v = next;
    for(int j = 2; j < i && v != 0; j++, v = v->next);
    if(v)
        return v->GB;
    return 0;
}

void GBoxes::setSize(int who, int x, int y, int w, int h){
    if(who == 1)
        setSize(x,y,x+w,y+h);
    else {
        GBoxes *v = next;
        for(int j = 2; j < who && v != 0; j++, v = v->next);
        if(v)
            v->setSize(x,y,x+w,y+h);
    }
}

void GBoxes::setBox(){
    if(A)
        GB->MoveWindow(*A,TRUE);
    if(next)
        next->setBox();
}

//
// Va ~ bl function defs.
// * * * * *
Varbl:: ~ Varbl(){ //cleans up after a variable is done
//     if(Eng) delete Eng;
//     if(Metric) delete Metric;
//     if(Btext) delete Btext;
//     if(val) delete val;
//     if(ClearB) delete ClearB;
//     if(type == 3 || type == 4)

```

```

        delete Ecombo;
    else if (type > 10)
        delete Rbut;
    else if (type == 5)
        delete Bitmap;
    else
        delete Efield;
    if (next) delete next;
}

void Varbl::setIdClr(TDialog *p, TModule *m, int b, int ofs, int sp = 0){
    if (sp)
        ClearB = new NButton(p, b, m);
    else
        ClearB = new NButton(p, b, "", 4, ofs + 3, 12, 18, m);
}

void Varbl::setIdBit(TDialog *p, TModule *m, int b, int ofs, int wide, int sp = 0){
    IdEdit = b;
    if (sp)
        ClearB = new NButton(p, b, m);
    else
        ClearB = new NButton(p, b, "", (DLGW-wide)/2, ofs + 3, 12, 18, m);
    ClearB->Attr.Style |= BBS_BITMAP;
    ClearB->Attr.Style ^= WS_VISIBLE;
}

void Varbl::setIdBitmap(TDialog *p, TModule *m, int b, int ofs, int wide, int sp = 0){
    IdEdit = b;
    if (sp)
        Bitmap = new NButton(p, b, m);
    else
        Bitmap = new NButton(p, b, "", (DLGW-wide)/2, ofs + 3, 12, 18, m);
    Bitmap->Attr.Style |= BBS_BITMAP;
}

void Varbl::setIdEdit(TDialog *p, TModule *m, int b, int ofs, int sp = 0){
    IdEdit = b;
    if (sp)
        Efield = new TEdit(p, IdEdit, 9, m);
    else
        Efield = new TEdit(p, IdEdit, "", 31, ofs, 80, 25, 10, FALSE, m);
    Efield->SetValidator (new TNumValidator);
}

void Varbl::setIdRadB(TDialog *p, TModule *m, int b, int ofs, NGroupBox *G, int sp = 0){
    IdEdit = b;

```

138

```

        if(sp)
            Rbut = new NRadioButton(p, IdEdit, G,m);
        else
            Rbut = new NRadioButton(p, IdEdit, "",10,ofs+5,DLGW-30,18, G,m);
    }

void Varbl::setIdCombo(TDialog *p,TModule *m, int b, int ofs, int/* sp = 0*/) {
    IdEdit = b;
    Ecombo = new NCombo(p,b,"",4,ofs,107,25,m,DLGW-126,120,new
TNumValidator);
}

void Varbl::setIdText(TDialog *p,TModule *m, int b, int ofs, int sp = 0, int t = 0){
    if(sp)
        Btext = new VStatic(p, b, 20,m);
    else if(t == 6){
        Btext = new VStatic(p, b, "",5,ofs+3,DLGW-25,18,0,m);
        Btext->Attr.Style |= SS_CENTER;
    }else
        Btext = new VStatic(p, b, "",116,ofs+3,DLGW-130,18,0,m);
}

/*char far */ void Varbl::Disp(int w) { //Displays the info about the variable
/*    char *t;
    if(w)
        t = Metric;
    else
        t = Eng;
*/
    if(Btext)
        Btext->ShowText(w);
    if(type == 5 && varnum)
        if(w){
            Bitmap->Show(SW_HIDE);
            ClearB->Show(SW_SHOW);
        }else{
            ClearB->Show(SW_HIDE);
            Bitmap->Show(SW_SHOW);
        }
    //    return t;
} // returns the info to display next to the var box.

void Varbl::setValidate(char far *s){ //Not used at this time!!!!
    if(val)
        delete val;
    val = new char[strlen(s)+1];
    strcpy(val,s);
}

```

```

    }

void Varbl::setEnglish(char far *s){ //Enters the english info about the variable
/*
    if(Eng)
        delete Eng;
    Eng = new char[strlen(s)+1];
    strcpy(Eng,s);
*/
    Btext->SetValue(0,s,0,0);
}

void Varbl::setMetric(char far *s){ //Enters the metric info about the variable
/*
    if(Metric)
        delete Metric;
    Metric = new char[strlen(s)+1];
    strcpy(Metric,s);
*/
    Btext->SetValue(1,s,0,0);
}

Varbl *Varbl::addVarbl(Varbl *v){ //Create a new variable and add it to the list
    for(Varbl *t = v; t->next != 0; t = t->next);
    t->next = new Varbl;
    return t->next;
}

int Varbl::GetInput(){ //Gets the input from the edit fields and converts it to a float
    if(type == 3)
        data = Ecombo->GetData(Ecombo->GetSelIndex());
    else {
        char t[20];
        getText(t,10);
        if(t[0] == '\0')
            return 0;
        data = atof(t);
    }
    return 1;
}

int Varbl::SetInput(){ //Converts a float to a string then puts it in the edit field
    if(type == 7)
        Btext->DispText(data);
    else{
        char t[20];

```



140

```

        sprintf(t,"%f", data);
        t[9] = '\0';
        setText(t);
    }

    return 1;
}

// END of Varhl function defs.

//
// Funcdef function defs.
// * * * * *
Funcdef::Funcdef(){
    VarList = 0;
    DlgBox = 0;
    Modg = 0;
    Grps = 0;
    form = varSolv = NoVar = IsPar = rNum = 0;
    rPos = 5;
    Ltype = 0; //Last type.
    Grpos = 0; //The top row of the groupbox.
    numG = 0; //The number of vars in a group.
    Lset = 0;
}

int Funcdef::AddVar(char far *e, char far *m, int Id = 1, int vn = 0, float d = 0, int t = 0,
int spl = 0, long Type = 0){ //Adds a variable to the list and its attributs
    Varbl *vtmp;

    if(!VarList)
        VarList = vtmp = new Varbl;
    else
        if((vtmp = VarList->addVarbl(VarList)) == 0){
            MessageBox((HWND)NULL,"Out of
memory","Error",MB_SYSTEMMODAL);
            exit(-1);
        }
    vtmp->setType(t);
    vtmp->setVarNumber(vn);
    vtmp->setData(d);
    if(t > 10){
        GBoxs *Gt;
        if(t != Ltype)
            if(spl)
                if(Grps)
                    Gt = Grps->addBox(Grps,
DlgBox,Modg,IDC_P2G1+t-11);

```

```

else
    Gt = Grps = Grps->addBox(Grps,
DlgBox,Modg,IDC_P2G1+t-11);
else
    if(Grps){
        Grps->setSize(Ltype-10,5,Grpos,DLGW-
19,20*numG+7);
        rPos += 25;
        Gt = Grps->addBox(Grps,
DlgBox,Modg,IDC_P2G1+t-11,"",5,(Grpos = rPos),DLGW-19,10);
        numG = 0;
    }else
        Gt = Grps = Grps->addBox(Grps,
DlgBox,Modg,IDC_P2G1+t-11,"",5,(Grpos = rPos),DLGW-19,10);
        vtmp->setIdRadB(DlgBox,Modg, MinRad+rNum, rPos,Gt->GBptr(), spl);
        vtmp->setIdText(DlgBox,Modg, MinRad+rNum++, rPos,1);
        vtmp->AddText(e,0,0);
        vtmp->AddText(m,0,0);
        rPos += 20;
        numG++;
    }else{
        if(Ltype > 10)
            if(!spl){
                Grps->setSize(Ltype-10,5,Grpos,DLGW-19,20*numG+7);
                rPos += 30;
            }
            if(e && m){
                vtmp->setIdText(DlgBox,Modg, MinText-201+Id, rPos, spl, t);
                char Tdisp = 0;
if(Type & 32)
                if(t == 0)
                    Tdisp = '*';
                else if(t == 1)
                    Tdisp = '!';
                if(Tdisp){
                    char T1[200],T2[200];
                    T1[0] = Tdisp; T1[1] = ' '; T1[2] = '\0';
                    T2[0] = Tdisp; T2[1] = ' '; T2[2] = '\0';
                    strcat(T1,e);
                    strcat(T2,m);
                    vtmp->AddText(T1,0,0);
                    vtmp->AddText(T2,0,0);
                } else {
                    vtmp->AddText(e,0,0);
                    vtmp->AddText(m,0,0);
                }
            }
        }
    }
}

```

142

```

    if(t == 7 && !vtmp->GetText())
        vtmp->setIdText(DlgBox,Modg, MinText-201+Id, rPos, spl, t);
    if(t == 3 || t == 4)
        vtmp->setIdCombo(DlgBox,Modg,MinText+399+Id,rPos,spl);
    else if(t == 5){
        int hi = 0;
        try{
            TDib *bitm = new TDib(Modg->GetInstance(),Id+1000);
            vtmp->setIdBitmap(DlgBox,Modg,Id,rPos,bitm-
> Width(),spl);

            hi = bitm->Height();
            delete bitm;
        }
        catch(TGdiBase::TXGdi) {
            return (Grps != 0);
        }

        if(vn){
            try{
                TDib *bitm = new TDib(Modg->GetInstance(),vn+1000);
                vtmp->setIdBit(DlgBox,Modg,vn,rPos,bitm->Width(),spl);
                if(bitm->Height() > hi)
                    hi = bitm->Height();
                delete bitm;
            }
            catch(TGdiBase::TXGdi) {
                return (Grps != 0);
            }
        }
        rPos += hi - 15;
        }else if(t != 6 && t != 7){
            vtmp->setIdEdit(DlgBox,Modg, MinText-101+Id, rPos, spl);
        if(t != 2)
            vtmp->setIdClr(DlgBox,Modg, MinText-1+Id, rPos, spl);
            }
            rPos += 30;
        }
        if(t == 2)
            NoVar = 1;
        Ltype = t;
        return (Grps != 0);
    }

void Funcdef::addCombo(char far *s, float d){
    for(Varbl *v = VarList; v->n() != 0; v = v->n());
    if(!v->addStrCombo(s, d))
        ErrorBox("Can't add string, must have Combo Box",DlgBox);
}

```

```

void Funcdef::addText(char far *s, float V1, float Vh){
    for(Varbl *v = VarList; v->n() != 0; v = v->n()){
        v->AddText(s, V1, Vh);
    }
}

void Funcdef::setRad(){
    int Ltype = 0;
    if(Grps){
        Grps->setBox();
        for(Varbl *v = VarList; v != 0; v = v->n())
            if(v->Type() > 10)
                if(v->Type() != Ltype){
                    v->ChkRad();
                    Ltype = v->Type();
                }
    }
}

void Funcdef::Display(){//Displays the info about all the variables in the function list
    for(Varbl *v = VarList; v != 0; v = v->n())
        if(v->Type() != 7)
            v->Disp(form);
}

int Funcdef::ChkVars(){//Checks the variable to see if the data has been entered corectly
    char msg[256];
    int slvnum = 0;
    varSolv = 0;
    for(Varbl *v = VarList; v != 0; v = v->n())
        if(v->Type() < 11 && v->Type() != 2 && !(v->Type() >= 5 && v->Type() <= 7)){
            slvnum++;
            if(!v->GetInput())
                if(v->Type() == 1){
                    sprintf(msg,"You must enter a value for:\n%s",v->GetDispText());
                    ErrorBox(msg,DlgBox);
                    return 0;
                }else if(varSolv){
                    ErrorBox("You can only Solve for one variable",DlgBox);
                    return 0;
                }else
                    varSolv = slvnum;
        }
    if(!NoVar && !varSolv){

```

```

        MessageBox("You must leave one variable cell empty",DlgBox);
        return 0;
    }
    return 1;
}

int Funcdef::UpdateVars(){//Updates the edit fields for all the variables in the function
    for(Varbl *v = VarList; v != 0; v = v->n())
        if(v->Type() < 11 && v->Type() != 5 && v->Type() != 6 && v-
>Type() != 3)
            v->SetInput();
    return 1;
}

void Funcdef::Clear(int IdE){ //Clears all the edit fields of the variables in the function
    IdE = IdE - 100;
    for(Varbl *v = VarList; v != 0; v = v->n())
        if(IdE == v->GetIdEdit())
            v->Clear();
    }

int Funcdef::Comput(){ //Solves the function using the variable entered
    NumList *n = new NumList(DlgBox);
    for(Varbl *v = VarList; v != 0; v = v->n())
        if(v->Type() < 11 && v->Type() != 5 && v->Type() != 6)
            n->addNum(n,v->Data());

    if(Grps){ // Checks to see if there is a special set of radio buttons.
        int ltype = 0, i;
        for(Varbl *v = VarList; v != 0; v = v->n())
            if(v->Type() > 10){
                if(ltype != v->Type()){
                    i = 1;
                    ltype = v->Type();
                }
                if(v->CKD())
                    n->addNum(n,(float)i); // prepares info from the
radio buttons to be sent.
                i++;
            }
    }

    if(IsPar)
        Calc(form+1, sPar, n);
    else
        Calc(form+1, varSolv, n);
}

```

```

int i;
for(v = VarList, i = 0; v != 0; v = v->n())
    if(v->Type() < 11 && v->Type() != 5 && v->Type() != 6)
        v->setData(n->getNum(i++));

delete n;
return 1;
}

void Funcdef::Make3d(){
    TControl *c;
    for(Varbl *v = VarList; v != 0; v = v->n()){
        if((c = v->GetEdit()) != 0)
            ForceControl3d(c->HWindow);
        if((c = v->GetBut()) != 0)
            ForceControl3d(c->HWindow);
        if((c = v->GetText()) != 0)
            ForceControl3d(c->HWindow);
    }
}

char *Funcdef::GetWorkText(){
    for(Varbl *v = VarList; v->GetIdEdit() != LastId && v != 0; v = v->n());
    if(v != 0){
        char t[20];
        v->getText(t,10);
        return t;
    }
    return 0;
}

void Funcdef::SetWorkText(char *t){
    for(Varbl *v = VarList; v->GetIdEdit() != LastId && v != 0; v = v->n());
    if(v->GetIdEdit() == LastId)
        v->setText(t);
}

// end of Funcdef's function definitions.

//
// IsValidInput function Def for Validate Number
// * * * * *

BOOL TNumValidator::IsValidInput(char far* str, BOOL){ //Checks the data entered to see
if it is a valid float
    BOOL t = TRUE;

```

146

```

        for(int i = 0, p = 0, l = strlen(str); i < l; i++)
            if(i != 0 || str[i] != '-')
                if(str[i] == '.'){
                    if(p)
                        t = FALSE;
                    p++;
                }else if(str[i] < '0' || str[i] > '9')
                    t = FALSE;

    return t;
}

//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(NFuncDlg, TDialog)
//{{NFuncDlgRSP_TBL_BEGIN}}
    EV_BN_CLICKED(IDC_COMP, BNCalc),
        EV_BN_CLICKED(IDC_DONE, BNContinue),
        EV_BN_CLICKED(IDC_HELP, BNhelp),
        EV_BN_CLICKED(IDC_PUSHBMisc, BNMisc),
        EV_BN_CLICKED(IDC_RADIOBUTTONONE, BNEnglish),
        EV_BN_CLICKED(IDC_RADIOBUTTONM, BNMetric),
//{{NFuncDlgRSP_TBL_END}}
END_RESPONSE_TABLE;

//{{NFuncDlg Implementation}}

NFuncDlg::NFuncDlg (TWindow* parent, TResId resId, TModule* module, char far *who,
long FuncType):
    TDialog(parent, resId, module){

// Constructor code here.
    Func.setDlg(this,module);
    nVars = 0;
    RBm = 0;
    Bdone = 0;
    Bhelp = 0;
    Bmisc = 0;
//    Gbox = new TGroupBox(this,IDC_GBOX,module);
//    BEnglish = new NRadioButton(this,IDC_RADIOBUTTONONE,Gbox,module);
//    BCompute = new NButton(this,IDC_COMP,module);
    Gbox = 0;
    BEnglish = 0;
    BCompute = 0;

```

```

        X = 5; Y = 5;
        Name = who;
        Type = FuncType;
        HelpFile = 0;
        HelpID = 0;
    }

NFuncDlg::~NFuncDlg(){

    Destroy();

// INSERT>> Your destructor code here.
    if(BEnglish) delete BEnglish;
    if(BCompute) delete BCompute;
    if(Bdone) delete Bdone;
    if(Bhelp) delete Bhelp;
    if(Bmisc) delete Bmisc;
    if(RBm) delete RBm;
    if(Gbox) delete Gbox;
    if(HelpFile) delete HelpFile;
}

void NFuncDlg::SetupWindow(){
    TDialog::SetupWindow();
    int R,B;
    if(sDlg){
        TRect Da = GetClientRect();
        R = Da.Width();
        B = Da.Height()+35;
    }else{
        R = DLGW;
        B = Func.rowInc(77);
    }
    if(Type & 16){
        if(Parent){
            TRect Pa = Parent->GetWindowRect();
            X += Pa.left;
            Y += Pa.top;
        }
    }else{
        X = (GetSystemMetrics(SM_CXSCREEN) - R)/2 +4;
        Y = (GetSystemMetrics(SM_CYSCREEN) - B)/2;
    }
    MoveWindow(X,Y,R,B,TRUE);
    for(int i = 0; i < nVars; i++)
        Func.Clear(MinText+i);
    BEnglish->SetCheck(BF_CHECKED);
}

```



```

        Func.setRad();
        Func.Display();
        Func.Make3d();
        SetWindowPos(HWND_TOPMOST,10,10,10,10,SWP_NOSIZE |
SWP_NOMOVE);
    }

void NFuncDlg::BNContinue (){ // There are done with the dialog.
    CloseWindow();
}

void NFuncDlg::BNhelp(){ //Someone wants help
    WinHelp(HelpFile,HELP_CONTEXT,HelpID);
}

void NFuncDlg::BNEnglish (){ // English Units were chosen.
    Func.setForm(0);
    Func.Display();
}

void NFuncDlg::BNMetric (){ // Metric units were chosen.
    Func.setForm(1);
    Func.Display();
}

void NFuncDlg::BNCalc (){ // ~he botton for the calculation was pushed.
    if(!Func.ChkVars())
        return;

    Func.Comput();
    Func.UpdataVars();
}

void NFuncDlg::Make(){
    if(!sDlg){
        if(BEnglish)
            delete BEnglish;
        if(BCompute)
            delete BCompute;
        if(Gbox)
            delete Gbox;
        int pos = (DLGW - 4*70 - 80)/5;
        Gbox = new TGroupBox(this,IDC_GBOX,"",pos/2-
2,Func.rowInc(15),80,39,Func.GetModl());
        ForceControl3d(Gbox->HWindow);
    }
}

```

```

        BEnglish = new
NRadioButton(this, IDC_RADIOBUTTON1, "En&glish", pos/2+1, Func.rowPos()+1, 70, 17,
Gbox, Func.GetModl());
        ForceControl3d(BEnglish->HWindow);
        RBm = new
NRadioButton(this, IDC_RADIOBUTTON2, "&Metric", pos/2+1, Func.rowPos()+18, 70, 17,
Gbox, Func.GetModl());
        RBm->Attr.Style = BS_AUTORADIOBUTTON | WS_CHILD |
WS_VISIBLE;
        ForceControl3d(RBm->HWindow);
        if(!(Type & 8)){
            BCompute = new
NButton(this, IDC_COMP, "&E", 1.5*pos+80, Func.rowPos(), 50, 35, Func.GetModl(), TRUE)
;
            ForceControl3d(BCompute->HWindow);
        }
        if(!(Type & 4)){
            Bmisc = new
NButton(this, IDC_PUSHBMisc, "Mi&sc", 2.5*pos+150, Func.rowPos(), 50, 35, Func.GetMod
l());
            ForceControl3d(Bmisc->HWindow);
        }
        if(!(Type & 2)){
            Bhelph = new
NButton(this, IDC_HELP, "&h", 3.5*pos+220, Func.rowPos(), 50, 35, Func.GetModl());
            ForceControl3d(Bhelph->HWindow);
        }
        if(!(Type & 1)){
            Bdone = new
NButton(this, IDC_DONE, "&d", 4.5*pos+290, Func.rowPos(), 50, 35, Func.GetModl());
            ForceControl3d(Bdone->HWindow);
        }
    }
    Create();
}

void NFuncDlg::BNMisc () { // ~ he botton for the calculation was pushed.
    CopyDlg Cdlg(Bmisc, 26, Func.GetModl());
    Cdlg.Execute();
}

```

```

LRESULT CopyDlg::DefWindowProc (UINT msg, WPARAM wParam, LPARAM
lParam){

```

```

    LRESULT result;
    result = TDialog::DefWindowProc(msg, wParam, lParam);

```

```

    // INSERT>> Your code here.

```

150

```
        if(wParam == 107){
            Parent->Parent->PostMessage(msg,wParam,lParam);
            CloseWindow();
        }else if(wParam == 106){
            CloseWindow();
            Parent->Parent->PostMessage(msg,wParam,lParam);
        }

        return result;
    }

void CopyDlg::SetupWindow(){
    TRect pR = Parent->GetWindowRect();
    int Lh = 93;
    int y = pR.bottom;
    if(y+Lh > GetSystemMetrics(SM_CYSCREEN))
        y = pR.top-Lh;
    MoveWindow(pR.left,y,75,Lh,TRUE);
}

BOOL NWCopy(char *Ctext,HWND winHan){
    void *lptstrCopy;
    HGLOBAL hglbCopy;

    OpenClipboard(winHan);
    if (EmptyClipboard()) {
        /* Allocate a global memory object for the text. */
        hglbCopy = GlobalAlloc(GMEM_DDESHARE, sizeof(Ctext));
        if (hglbCopy == NULL) {
            CloseClipboard();
            return FALSE;
        }
        /* Lock the handle and copy the text to the buffer. */
        lptstrCopy = GlobalLock(hglbCopy);
        int lng = strlen(Ctext)+1;
        memcpy(lptstrCopy, Ctext, lng);

        GlobalUnlock(hglbCopy);

        /* Place the handle on the clipboard. */
        SetClipboardData(CF_TEXT, hglbCopy);
        CloseClipboard();
        return TRUE;
    }
    return FALSE;
}
```

```

void NWCpaste(char *Ctext,HWND winHan){
    HGLOBAL hglb;
    void *lptstr;

    //    TClipboard& Cbrd;
    OpenClipboard(winHan);
    if(IsClipboardFormatAvailable(CF_TEXT)){
        hglb = GetClipboardData(CF_TEXT);
        if (hglb != NULL) {
            lptstr = GlobalLock(hglb);
            if (lptstr != NULL){
                memcpy(Ctext, lptstr, 10);
                GlobalUnlock(hglb);
            }
        }
        CloseClipboard();
    }
}

LRESULT NFuncDlg::DefWindowProc (UINT msg, WPARAM wParam, LPARAM
lParam){
    LRESULT result;
    result = TDialog::DefWindowProc(msg, wParam, lParam);

        // INSERT>> Your code here.
    if(wParam >= MinText-101 && wParam <= MaxText-101){//
        Func.SetLID(wParam);
    }
    else if(wParam >= MinText+399 && wParam <= MaxText+399){//
        Func.SetLID(wParam);
    }
    else //if(msg == BN_CLICKED){
        if(wParam >= MinText && wParam <= MaxText) // Checks if a clear button was
        pushed
            Func.Clear(wParam);
        else if(wParam == 107){//Copy Item
            char ctmp[24];
            strcpy(ctmp,Func.GetWorkText());
            NWCcopy(ctmp,this->HWindow);
        }else if(wParam == 106){//Paste item
            char ctmp[24];
            NWCpaste(ctmp,this->HWindow);
            TNumValidator NVald;
            if(NVald.IsValidInput(ctmp, TRUE))
                Func.SetWorkText(ctmp);
        }
    }
    // }
    return result;
}

```

```

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
float calcE9_4B(int i, //English or Metric
                int j, NumList *n){//the list of data
to transfer
    int wN = 0;
    float fHf = n->getNum(wN++); // Discharge (cfs or cms)
    float fL = n->getNum(wN++); // length of culvert barrel (ft or m)
    float fR = n->getNum(wN++); // hydraulic radius (ft or m)
    float fV = n->getNum(wN++); // velocity (ft/sec or m/sec)
    float fn = n->getNum(wN++); //Manning's Rougness Coefficient

float fg = 32.2, fTemp;
if (i==2)
{
    fHf = fHf * 3.28084;
    fL = fL * 3.28084;
    fR = fR * 3.28084;
    fV = fV * 3.28084;
}
switch(j)
{
    case 1:
        if(fR)
            fHf = ((29 * (pow(fn,2)) * fL * (pow(fV,2))) / ((pow(fR,1.33)) * 2 * fg));
        else
            MessageBox("A value of 0 has been entered for a variable in the
denominator!",n->GetWnd());
        break;
    case 2:
        if(fn*fV)
            fL = (fHf * (pow(fR,1.33)) * 2 * fg) / (29 * (pow(fn,2)) * (pow(fV,2)));
        else
            MessageBox("A value of 0 has been entered for a variable in the
denominator!",n->GetWnd());
        break;
    case 3:
        if(fg*fHf)
            fR = pow(((29 * (pow(fn,2)) * fL * (pow(fV,2))) / (2 * fg * fHf)),0.75188);
        else
            MessageBox("A value of 0 has been entered for a variable in the
denominator!",n->GetWnd());
        break;
    case 4:

```

```

        if(fn*fL)
        {
            fTemp = (fHf * 2 * fg * (pow(fR,1.33))) / (29 * (pow(fn,2)) * fL);
            fV = pow(fTemp,0.5);
        }
        else
            MsgBox("A value of 0 has been entered for a variable in the
denominator!",n->GetWnd());
        break;
    default:
        MsgBox("BUG !!!",n->GetWnd());
    }
    if (i==2)
    {
        fHf = fHf / 3.28084;
        fL = fL / 3.28084;
        fR = fR / 3.28084;
        fV = fV / 3.28084;
    }

    wN = 0;
    n->setNum(wN++, fHf);// Discharge (cfs or cms)
    n->setNum(wN++, fL);// length of culvert barrel (ft or m)
    n->setNum(wN++, fR);// hydraulic radius (ft or m)
    n->setNum(wN++, fV);// velocity (ft/sec or m/sec)
    n->setNum(wN++, fn);//Manning's Rougness Coefficient
    return 1;
}

```



Appendix F.

Computer-based Manual Evaluation Form





---

# Computer-based Manuals

## Evaluation

Utah Department of Transportation  
Utah Transportation Center  
Utah State University

September 1994

How skilled are you at using Windows based computer programs (check one)?

Very skilled	Average	Below average	Have never used Windows before
--------------	---------	---------------	-----------------------------------

*Thank you for your help!!!*

1) Rate the importance of each of the following features for each of the categories.

**3 = Very Important    2 = Important    1 = Little Importance    0 = Not Important**

Feature	Training (0-3)	Reference (0-3)	Decision Support (0-3)	Comments
Browser				
Hypertext				
PopUp Windows				
Search				
Glossary				
History List				
Bookmarks				
Annotation				
Multimedia (Video, bitmaps)				
Equation Solvers				
External Program Links				
HyperCalc				
Decision Support System				

- 2) Compare the printed version to the computer-based version of the AASHTO Drainage Manual for each of the following categories. Mark which version of the manual is better in each category and then rate it using the following scale:

**3 = Much Better    2 = Significantly Better    1 = Slightly Better    0 = About the Same**

	<b>Printed Version</b>	<b>Computer Version</b>	<b>Rating (0-3)</b>	<b>Comments</b>
Speed				
Accuracy				
Navigation				
Overall Ease of Use				
Level of Information				
Comprehension of Information				
Metric Conversions (Hypercalc vs. Manual)				

- 3) What do you think the advantages and disadvantages of the computer-based design manual are?
- 4) What did you most like or dislike about the computerized manual?
- 5) What could be done to make the computer-based manual easier to use?
- 6) Other comments:



## Appendix G

### Computer-based Manual Evaluation Results



Table C-1. Results from the Utah Department of Transportation evaluation of the computer-based manual features(0 = not important, 1 = little importance, 2 = important, 3 = very important)

Evaluator	Browser			Hypertext			PopUp Windows			Search		
	Train.	Ref.	D.S.	Train.	Ref.	D.S.	Train.	Ref.	D.S.	Train.	Ref.	D.S.
1	3	3	3	3	3	3	3	3	3	3	3	3
2	3	3	1	3	3	1	3	3	1	3	3	1
3	1	3	3	3	3	2	2	3	3	1	3	3
4	3	3	0	3	3	1	3	3	1	2	3	1
5	3	3	3	3	3	3	3	3	2	3	3	3
6	3	3	2	2	2	1				2	2	1
7	2	2	2	2	2	2	2	2	2	2	3	2
8	3	2	1	3	3	1	2	3	1	2	3	2
9	3	2	1	3	3	2	3	3	3	2	3	1
10	3	3	1	3	3	2	2	2	2	2	3	3
11	2	3	0	3	3	3	3	3	3	3	3	3
12	3	3	2	3	3	2	2	2	3	3	3	3
13	3	3		3	3		2	2		3	3	
14	2	3	1	3	2	1	1	2	2	3	3	2
Avg.	2.64	2.79	1.54	2.86	2.79	1.85	2.38	2.62	2.17	2.43	2.93	2.15
STD	0.63	0.43	1.05	0.36	0.43	0.80	0.65	0.51	0.83	0.65	0.27	0.90
95% C.I.	2.97-1.96	3.01-2.56	2.09-0.99	3.05-2.67	3.01-2.56	2.27-1.43	2.73-2.04	2.88-2.35	2.60-1.73	2.77-2.09	3.07-2.79	2.62-1.68



Table C-1. Continued

Evaluator	Glossary			History List			Bookmarks			Annotation		
	Train.	Ref.	D.S.	Train.	Ref.	D.S.	Train.	Ref.	D.S.	Train.	Ref.	D.S.
1	3	3	3	3	3	3	3	3	3	3	3	3
2	3	3	1	3	3	1	1	3	1	1	3	1
3	2	1	1	0	1	1	3	1	1	3	2	2
4	3	3	1	2	1	1	2	3	1	3	3	3
5	2	3	2	2	2	2	2	3	2	1	3	3
6	3	3	1	1	1	1	2	2	1	2	2	1
7	2	2	2	1	1	1	2	2	2	1	3	2
8	3	2	2	0	1	0	2	2	0	2	2	1
9	3	3	2	1	1	3	1	3	3	2	2	2
10	3	3	1	3	2	1	3	3	1			
11	3	1	0	3	3	3	3	3	3	2	2	1
12	3	3	2	2	2	2	3	3	2	3	3	3
13	3	2		3	2		2	2		3	2	
14	3	3	2	2	2	0	2	2	1	2	3	1
Avg.	2.79	2.50	1.54	1.86	1.79	1.46	2.21	2.50	1.62	2.15	2.54	1.92
STD	0.43	0.76	0.78	1.10	0.80	1.05	0.70	0.65	0.96	0.80	0.52	0.90
95% C.I.	3.01-2.56	2.90-2.10	1.95-1.13	2.43-1.28	2.21-1.37	2.01-0.91	2.58-1.85	2.84-2.16	2.12-1.11	2.57-1.73	2.81-2.27	2.39-1.45

Table C-1. Continued

Evaluator	Multimedia			Equation Solvers			External Program Links			HyperCalc		
	Train.	Ref.	D.S.	Train.	Ref.	D.S.	Train.	Ref.	D.S.	Train.	Ref.	D.S.
1	3	3	3	3	3	3	3	3	3	3	3	3
2	1	3	1	3	1	3	2	2	3	1	1	3
3	3	3	1	3	3	3	3	1	1	2	1	1
4	3	2	2	3	3	3	2	3	3	3	3	3
5	3	3	1	3	3	3	2	3	1	2	2	2
6	3	3	1	3	2	1	2	2	2	2	2	2
7	3	2	1	3	2	3	3	1	3	2	2	3
8	2	2	1	0	2	3	1	3	3	2	2	3
9	2	2	1	2	1	3	2	2	3	3	3	2
10	2	1	0	1	2	3	2	2	3	2	2	2
11	3	1	3	3	3	3	3	3	3	3	3	2
12	2	1	0	3	3	3	2	3	3	3	3	3
13	3	3		2	3		3	2		2	2	
14	3	1	0	1	1	3	0	1	3	1	2	3
Avg.	2.57	2.14	1.15	2.36	2.29	2.85	2.14	2.21	2.62	2.21	2.21	2.46
STD	0.65	0.86	0.99	1.01	0.83	0.55	0.86	0.80	0.77	0.70	0.70	0.66
95% C.I.	2.91-2.23	2.60-1.69	1.67-0.64	2.89-1.83	2.72-1.85	3.14-2.56	2.60-1.69	2.630-1.79	3.02-2.21	2.58-1.85	2.58-1.85	2.81-2.12

Table C-1. Continued

Evaluator	Decision Support System		
	Train.	Ref.	D.S.
1	3	3	3
2	1	1	3
3	2	2	3
4	3	3	3
5	2	2	3
6	3	3	3
7	2	2	3
8	1	2	3
9	2	2	3
10	2	1	3
11	3	3	3
12	2	3	3
13	3	2	
14	1	2	3
Avg.	2.14	2.21	3.00
STD	0.77	0.70	0.00
95% C.I.	2.55-1.74	2.58-1.85	3.00-3.00