Department of Computer Science

Sleuth User's Guide Version 1.2

Jeffrey J. Walls

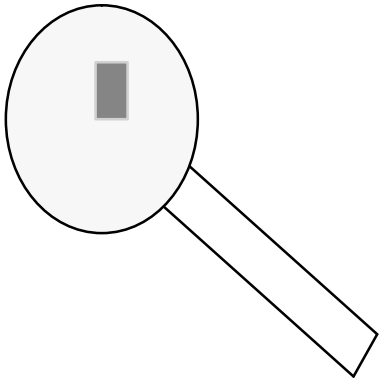
Technical Report CS-94-103

February 7, 1994

Colorado State University

Sleuth User's Guide

Version 1.2



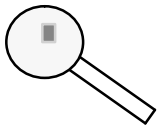
For Sleuth Version 1.2

Sleuth User's Guide

Version 1.2

Sleuth User's Guide

Version 1.2



1.0 Introduction

This document will guide you through the steps for creating and reusing test scripts with Sleuth. Sleuth was designed to meet the needs of the **Software Tester** and uses a processes similar to that of an actual engineer. Using domain information, sleuth generates tests for a Command Line Interface that are both syntactically and semantically correct.

1.1 Typographical Conventions

In this document, the following conventions will be used:

Text that the user should type will appear in **BOLDFACE**
UNIX Files will appear in the *`Italics Courier`* font
If a Motif Menu Item is to be selected by the user, s/he should use Mouse Button 1. Furthermore, a menu item will be denoted with the following notation:

Menu->MenuItem

For example,

Specification->Script

would mean for the user to select the menu item "Script" in the "Specification" Menu.

1.2 Where To Go From Here

If you are unfamiliar with the Motif Interface, go to section 1.3, *Using Motif.*
If you have never run ***Sleuth*** in your life, go directly to section 2.0, *Getting Started*.
If you need help with *Specification,* see section: 3.0
If you need help with *Configuration*, see section: 4.0
If you need help with ***Sleuth*** syntax for generating commands, see section: 5.0
If you need help with the *Syntax Diagram Editor,* see section: 6.0

1.3 Using Motif

The OSF/Motif User Interface is extremely simple to navigate. This section describes all Motif Widgets (as they are called) that are used in ***Sleuth***, and how to use them. A *widget* is basically the way Motif represents its Input and Output. For example, a Push Button is a widget that lets the user make a selection (i.e. Input). So, with that in mind, we now proceed with a description of each Motif Action and Widget and how they are used.

1.3.1 Motif Windows

Figure 1 shows what a typical Motif Window looks like. The user should become familiar with the areas indicated in the figure.

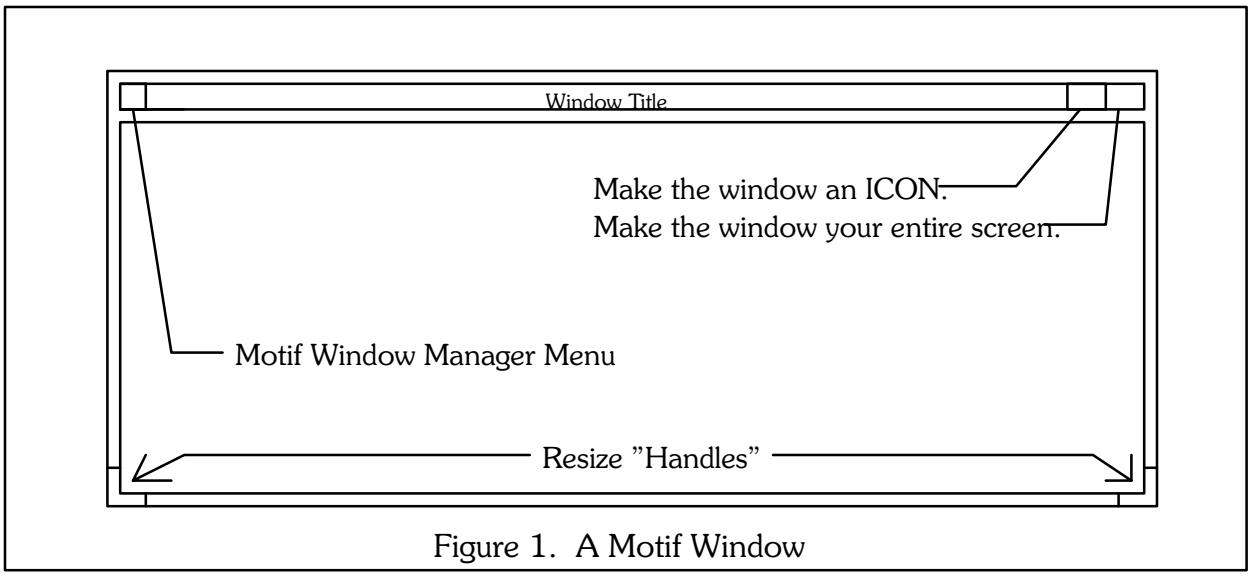


Figure 1. A Motif Window

1.3.2. Using the Mouse

During this course of this guide, we will refer to particular Mouse Buttons (usually 1, 2 or 3). Usually, the Left-most mouse button is Mouse Button 1 and the right-most button is Mouse Button 3. If you are left-handed, you may have these reversed. Whatever the case, you should determine the Mouse Button layout for your Mouse so the rest of this guide will make sense to you.

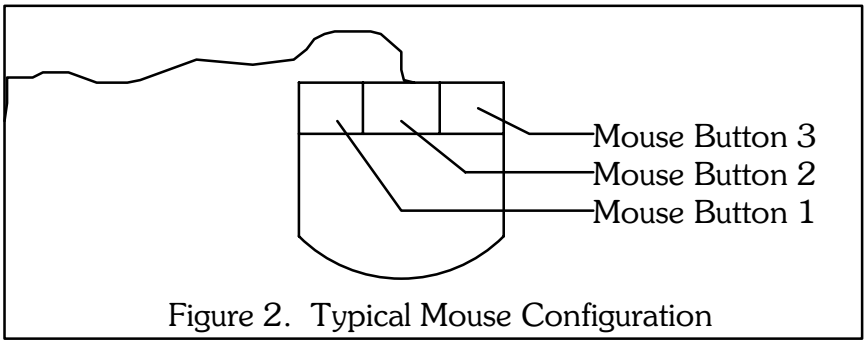


Figure 2. Typical Mouse Configuration

There are also two terms that you should be familiar with about using your mouse. The term **click** refers to pressing a Mouse Button and releasing it in one motion, over one position on the screen. The term **drag** refers to pressing a Mouse Button, moving the cursor to a new position on the screen (while the button is pressed), and releasing the Mouse Button when the new position is reached.

1.3.3 Motif Menus

In Motif, there are basically three types of menus that we will use in ***Sleuth***: *Pull-down Menu, Pop-up Menu,* and *Option Menus.* These are shown in Figures 3a-3c, respectively.

A Pull-down Menu is activated by clicking Mouse Button 1 over the text for that menu. Doing this will "pull the menu down." The actual menu items are activated by clicking Mouse

Button 1 over the text for that menu item. Figure 3a shows this action. Also notice that after

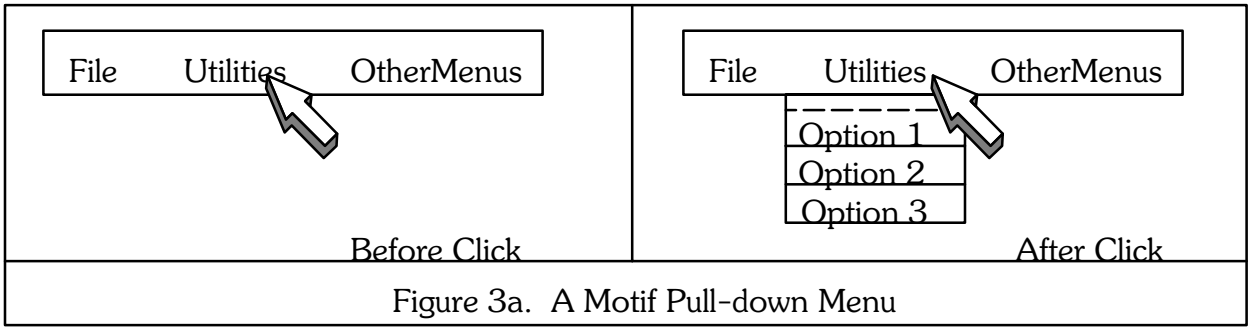


Figure 3a. A Motif Pull-down Menu

the Utility Menu has been pulled down, the first menu option is a dashed line (sometimes double-dashed line). This is called the *Tear Off* option. If this option is selected (i.e. If the user clicks on the dashed line), the Pull-down menu will become a window all to itself. This is extremely useful when menu items are repeatedly selected. It becomes a bother to Pull the menu down each time. This facility lets the user have a separate window comprised only of the Pull-down Menu's options. This is shown in Figure 3aa.

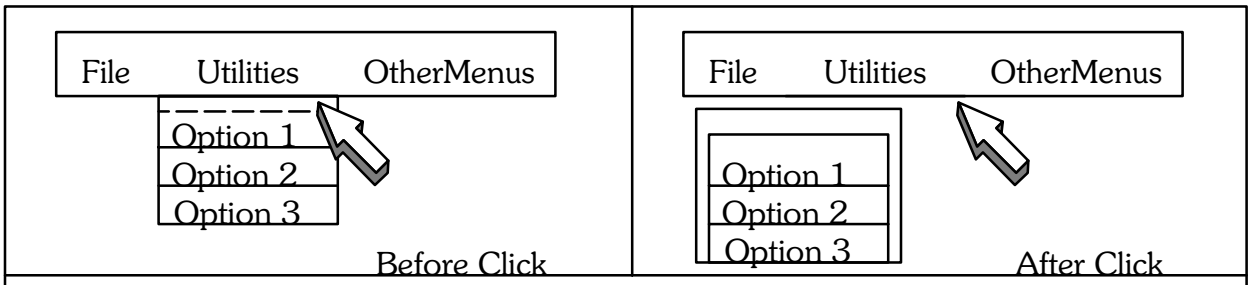


Figure 3aa. A Tear Off Menu

Another type of Motif Menu is the Pop-up menu. In ***Sleuth***, this will only occur in the Syntax Diagram Editor. It is activated by clicking Mouse Button 3 over any part of the Editor. See Figure 3b. Pop-Up Menus also exhibit the Tear-Off Model. This proves invalu-

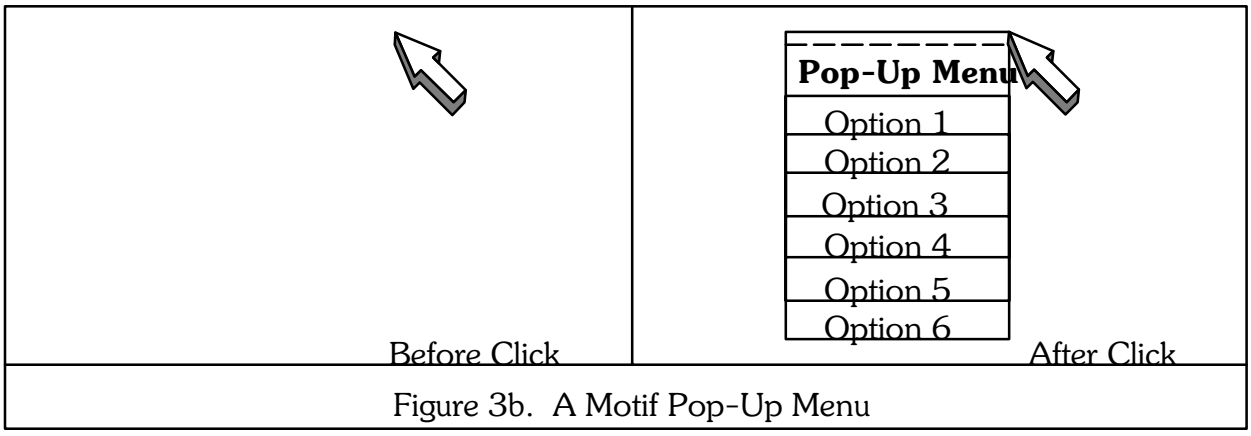


Figure 3b. A Motif Pop-Up Menu

able when using the Syntax Diagram Editor. Once the Pop-up menu is visible, either Mouse Button 1 or Mouse Button 3 may be used to actually select an item from within the menu.

The last type of Motif Menu is the Option Menu. This is utilized only in the Menu Option: Utilities->Generate Commands From Class. This menu is activated by clicking Mouse

Button 1 over either the Text of the Menu or the **dash** to the right of the Text. Once the menu is visible, the menu items will be selected by clicking Mouse Button 1 over the text of the desired option. The menu item selected will then become the text for the entire menu. This may take some getting used to, but it is by far one of the more convenient widgets in Motif. Figure 3c shows how to operate this menu.

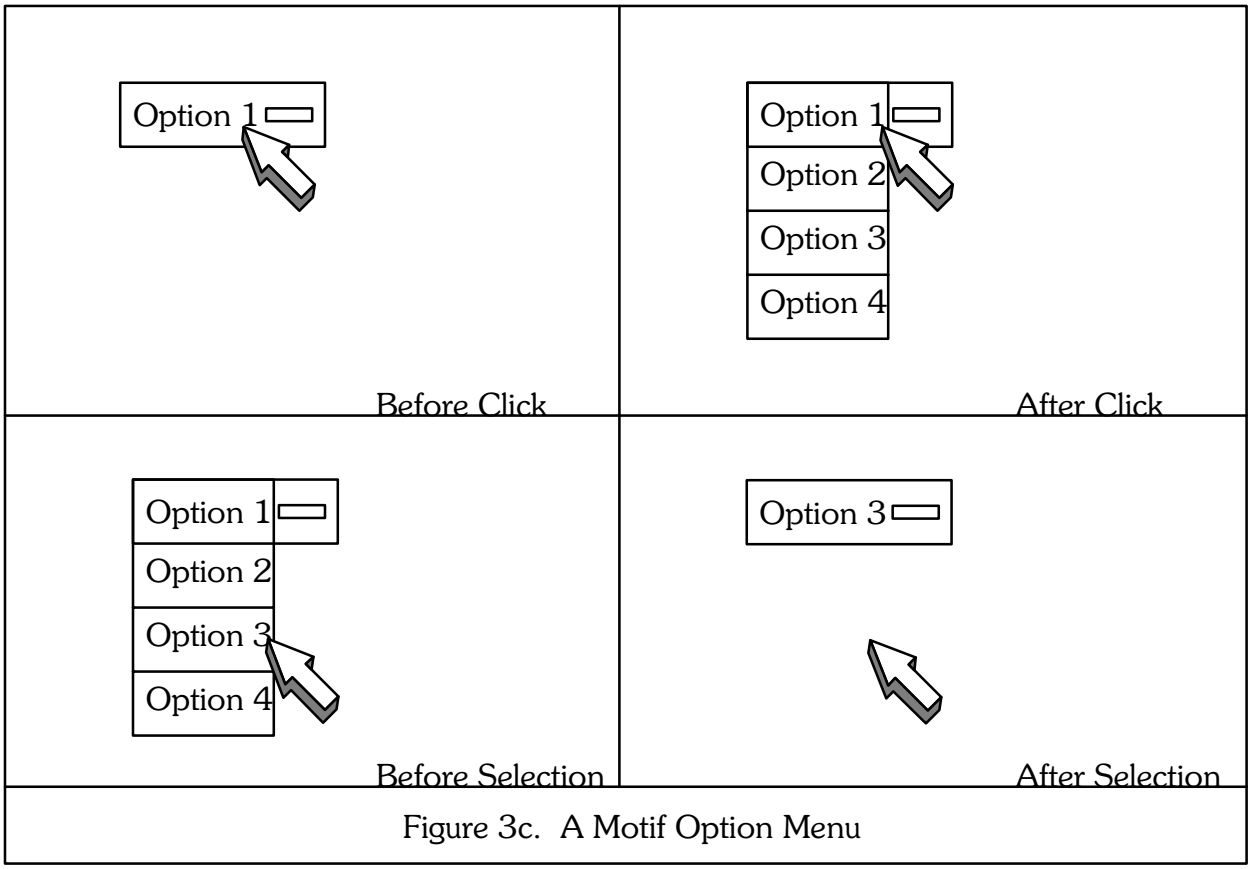


Figure 3c. A Motif Option Menu

Keep in mind that all of these menu operations could be performed by dragging the menus, and selecting the appropriate choice. Simple Button Clicking was described for simplicity.

1.3.4 Motif Buttons

Once again, in Motif, there are several types of Buttons that will be utilized throughout ***Sleuth.*** These are the: Push Button, Drawn Button, and Toggle Button.

The Push Button is analogous to the RETURN Key on your keyboard. It is a button that you can push, and when you do, it initiates some action. In Motif, you activate the Push Button by clicking directly over it with Mouse Button 1. See Figure 4a. Usually, the Text on the Push Button will indicated something about what action will be initiated. For example, the **Export** button will be used to Write (Export) a script to a file.

The Drawn Button is analogous to the UP Button for an Elevator. Besides having some Text associated with it, there is also a picture of some sort indicating some greater level of

Figure 4a. A Motif Push Button

detail than the text. These are activated using the same technique used for Push Buttons. Figure 4b shows an actual ***Sleuth*** Drawn Button.

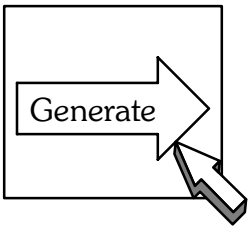


Figure 4b. A Sleuth Drawn Button

The Toggle Button is more correctly thought of as a *Switch*, such as a Light Switch. The Button (or switch) is in one of two states, On or Off. If a Toggle Button is On, we say it is *Selected*, and if a Toggle Button is Off, we say it is *Not Selected*. A Toggle Button's state is altered by clicking Mouse Button 1 over either the Text of the button or the **box** to the left of the Text. Figure 4c shows the result of selecting three of the Toggle Buttons. The current

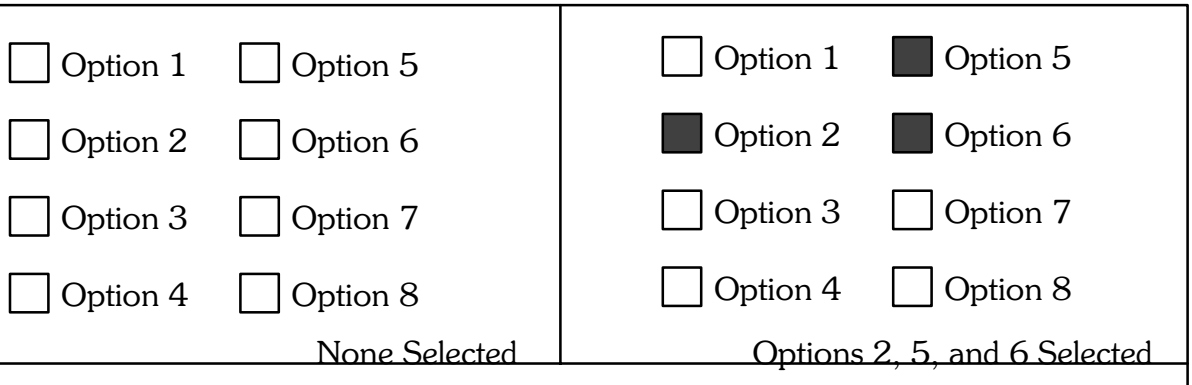


Figure 4c. Motif Toggle Buttons

Sleuth defaults show a Selected Toggle Button with a Blue Box, and Not Selected Toggle

Buttons with a Gray Box. This may be modified to the user's preference as described in section 1.4.

1.3.5 Text Areas

The text area is very simple. Basically, it is a place for displaying large quantities of text, or a place where the user can enter some data to be processed. To enter text into a Text Area, the user should click Mouse Button 1 in the text area, and then simply type the desired text.

1.3.6 Selection Boxes

In ***Sleuth*** there are two types of selection boxes the user will need to navigate: File Selection Boxes, Simple Selection Boxes. A File Selection Box lets the user select a file from the UNIX Directory Hierarchy. To select a file from this type of box, the user should double-click Mouse Button 1 over the desired FILE. If the user double-clicks over an option in the DIRECTORY list, the files under the selected directory will be displayed in the FILE list. The FILTER area shows the path to the current FILE list. Figure 5 shows the user selecting the file */users/walls/Sleuth/BugBuster*.

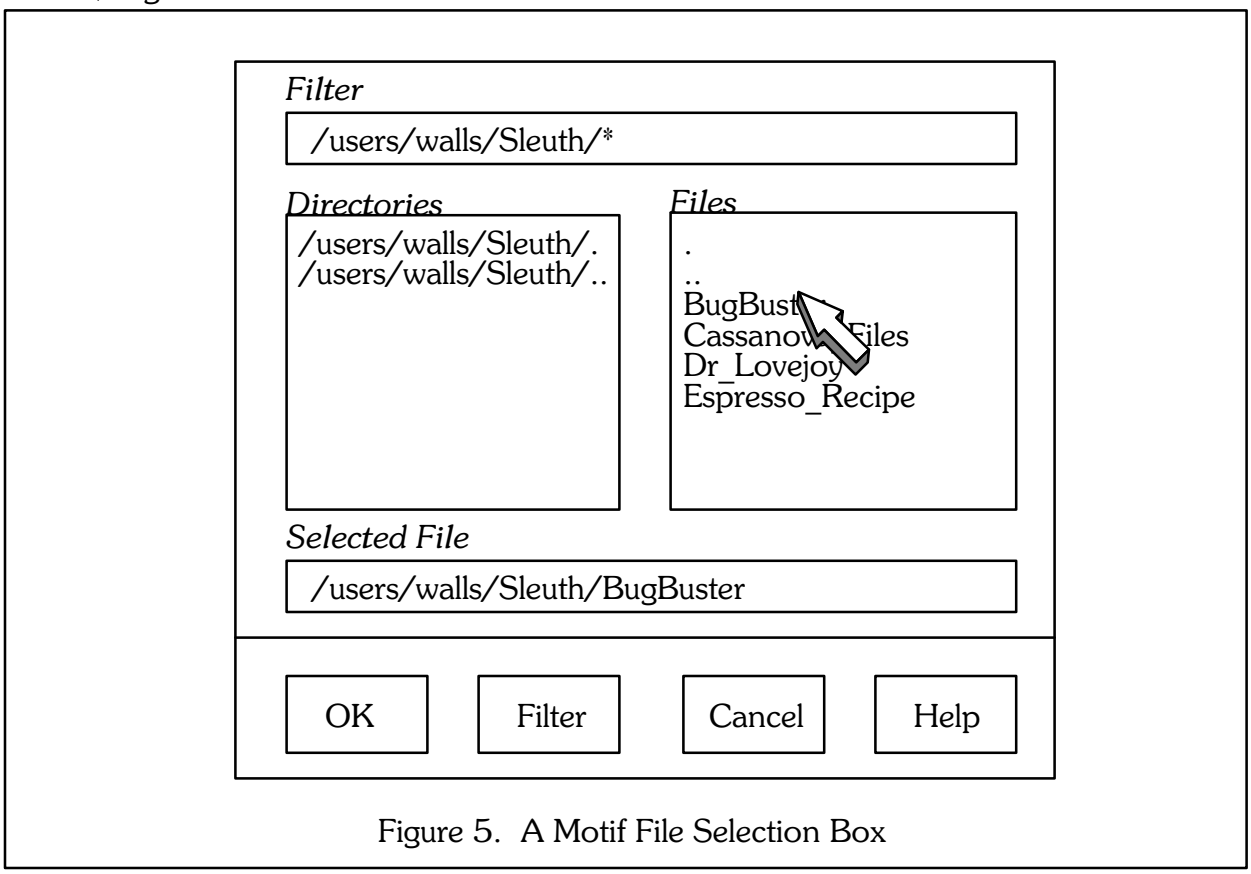


Figure 5. A Motif File Selection Box

A Simple Selection Box is similar to a File Selection Box, except that there is only one list to choose from.

1.4 Modifying Sleuth Defaults

Sleuth is now equipped with a way for the user to easily modify text, color, and geometry attributes of ***Sleuth***'s Windows. The Sleuth Defaults File is Located in the same directory as Sleuth's **bin** directory in a file called **Sleuth**. DO NOT MODIFY THIS FILE DIRECTLY! Copy this file to your home directory:

```
% cp  <path_to_Sleuth>/Sleuth   $HOME/
```

Now, you may edit the file $HOME/Sleuth and customize Sleuth to your own liking. This section will only briefly describe possible changes. For a complete discussion of resource files, consult the Motif Users Guide.

1.4.1 Modify Colors

To modify a color, edit the $HOME/Sleuth file and search for the Widget you wish to modify. The following table shows the relationship of Motif Widgets to names given in the $HOME/Sleuth file. Currently, you must modify all instances of a given widget. If the need arises, full control over all of the widget's individual attributes will be provided.

Motif Widget	Sleuth Name
Push Button	pbutton
Drawn Button	dbutton
Toggle Button	toggle
Label Widget (not described in guide)	label
Text Widget	text
Menu Options	menubutton
Selection Boxes	FS

Next, you will need to know a list of the colors available to you. There is a file containing all colors that you may use, and it is usually located in:

```
/usr/lib/X11/rgb.txt
```

If this file does not exist, you should contact your system administrator to find the exact location.

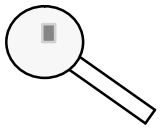
1.4.2 Modify Fonts

Currently, the user cannot modify the font for the Syntax Diagram Editor. This is because of special properties of this editor. Again, if the need arises, contact avm@cs.colostate.edu and we will provide this.

Using the same table shown in section 1.4.1, find the Motif Widget whose font you wish to change. Next, you will need to locate the file containing the fonts available to you. The usual method is to type:

```
% xlsfonts | more
```

at the command line. If this program does not exist, consult your system administrator. Notice that in the $HOME/Sleuth file, the full font name is not listed. Instead, only the unique part of the name is written with asterisks ("*") surrounding it. This is all that is required. If you type an ambiguous name (i.e. *courier*), the first instance of "courier" will be used for the font.



2.0 Getting Started

2.1 Starting Sleuth

To start ***Sleuth***, go to the *bin* directory of the Sleuth Directory Hierarchy, and at the UNIX Command Line, type: **sleuth**. This will start the ***Sleuth*** session. Alternatively, the user may put the Sleuth Bin Directory (..../Sleuth/bin) in the current user PATH (via the .cshrc for *csh* or .profile for *ksh* or *sh*) and type **sleuth** from anywhere in the system.

The ***Sleuth*** session begins with an animated Disclaimer about ***Sleuth*** and how to get further licensing. Using Mouse Button 1, click anywhere inside the Disclaimer Window to continue.

Once the Disclaimer is dismissed using the above action, the **Sleuth** window will appear. This window is distinctive by it's *Three Column Interface.* Section 6.0 will discuss the Navigation of this interface as well as detail how each phase may be used to enhance testing.

2.2 Getting Help

Sleuth provides the user with On-Line help at *every* level of the interface. The user should experiment with the different Help Options to see how this works. If all else fails, contact:

Dr. Anneliese von Mayrhauser
Colorado State University
avm@cs.colostate.edu

2.3 The Sleuth Environment

When you initiate ***Sleuth***, a special environment is created to facilitate moving around the UNIX Filesystem. The following *UNIX Environment Variables* are set by ***Sleuth*** but may be modified by the user in the file: *$HOME/.sleuthrc*.

USERSCRIPTWORK -*Where Mega Scripts will be Stored*
USERCOMMANDWORK -*Where Mega Commands will be Stored*
USERPARAMWORK -*Where Mega Parameters will be Stored*

A sample *$HOME/.sleuthrc* file for csh could look like the file shown in Figure 6.

2.4 Sleuth Terminology

To run ***Sleuth*** properly, the user should be familiar with the terminology discussed in this section. The following is a list of terms that will be used in this document followed by a brief

```
#!/bin/csh
setenv USERSCRIPTWORK $HOME/Sleuth/Mega/Script
setenv USERCOMMANDWORK $HOME/Sleuth/Mega/Command
setenv USERPARAMWORK $HOME/Sleuth/Mega/Param
```

NOTE: The above assignments assume the pre−existance of the following directory structure:
$HOME
$HOME/Sleuth
$HOME/Sleuth/Mega
$HOME/Sleuth/Mega/Script
$HOME/Sleuth/Mega/Command
$HOME/Sleuth/Mega/Param

Figure 6. Sample sleuthrc files

discussion of the term and how it may be used.

MegaScript - *A Saved Sequence Of Commands.*

MegaCommand - *A Saved Sequence Of Command Templates*.

MegaParameter - *A Saved Sequence Of Parameterized Commands.*

Sequence Of Commands - *A list comprised only of Command Names from the current domain. These will appear in the SCRIPTING WINDOW of the* ***Sleuth*** *Interface.*

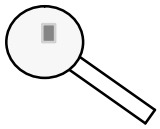
Command Templates - *An instance of a Command in a Syntax Diagram format. In other words, these commands will be random occurrences of commands with no parameter PLACEHOLDERS instead of actual values. These will appear in the COMMANDS WINDOW of the* ***Sleuth*** *Interface.*

Parameterized Commands - *The full command with actual parameter values substituted for the parameter PLACEHOLDERS.*

Scripting Class - *A Collection of Commands. In general, a Scripting Class is an easy way for the user to group commands together in various ways. For example, all commands that perform some action, could be a member of the* Action *class. For completeness, it is generally a good idea to have an* Any *class, which is the Union of all other classes. See Sections 3.2.1.1-3.2.1.3 for Creating, Modifying, and Removing Scripting Classes. See Section 4.2.1.1 for Turning Commands within a class On and Off.*

Specification - *The Specification of a domain is the Current Valid state of your domain. In other words, if your current software release is Version 1.2, the information necessary to run Version 1.2 of your software would be installed as the current Specification. The user may change this at any time but should exercise caution when doing so, because any change made here will alter other user's specifications as well.*

Configuration - *The Configuration of a domain is a user's local copy of the Specification. This copy may be modified in any way without altering any other user's copy of the specification. Configurations are set up according to USERNAME (i.e. One Configuration Per username), although in section 4.3 you will see how they may be saved and restored.*



3.0 The Specification of a Domain

3.1 Modifying the Current Specification

The Specification of your domain should be a fairly static entity. In other words, once the specification is built, it should not need to be modified on a regular basis. The Specification is the *Global Copy* of the Domain. What this means is, any modification made to the specification will be felt by all users of ***Sleuth***. Currently, this option is not protected in any way, so each user should confer with others before making changes here.

3.2 What About The Specification Can I Change?

Using Mouse Button 1, click over the word: **Specification** on the ***Sleuth*** Interface. This Pull-down menu gives the user five options:

- Script
- Command
- Parameter
- Save Specification (See section 3.3.1)
- Load Specification (See section 3.3.2)

3.2.1 Scripting Specification

To modify the specification of the Scripting Phase (generating lists of commands), the user should select: **Specification->Script**. This option is fairly slow to come up, so be patient. If you click on this multiple times, you will get multiple Specification Windows. The ***Sleuth Specification Window (Scripting)*** will let the user do the following:

- Add a Scripting Class
- Modify a Scripting Class
- Remove a Scripting Class
- Add/Modify/Remove Commands within a Scripting Class
- Modify Frequency Information related to a command within a scripting class
- Add/Modify/Remove Scripting Rules for a Command (applies to a command whether is appears in one scripting class or multiple ones)

3.2.1.1 Adding a Scripting Class

To Add a Scripting Class to the Domain Specification, select: **Tools->Add Class.** A small dialog will appear asking for the name of the class. Click in the text area of the dialog and type the name of your new class (press [RETURN] or click OK to accept).

3.2.1.2 Modifying a Scripting Class

To Modify a Scripting Class in the Domain Specification, select: **Tools->Modify Class Name**. This will cause a small *Selection Dialog* to appear on your display. Using Mouse Button 1, click ONCE on the class whose name you wish to change, then Press the OK Button. A new dialog will appear asking for the new name. Enter this the same way you did in section 3.2.1.1 to add a class. The current name for the class will appear in the text area of the dialog. To remove this, you can use one of two methods:

- Using Mouse Button 1, double–click inside the text area (this causes the text to be fully highlighted), and press the [BACKSPACE] key.
- Using Mouse Button 1, single–click to the RIGHT of the text (inside the text area) and press [BACKSPACE] until the entire text is removed.

3.2.1.3 Removing a Scripting Class

To Remove a Scripting Class from the Domain Specification, select: **Tools->Remove Class**. A *Selection Dialog* will appear with a list of all of the current scripting classes. Using Mouse Button 1, single-click on the Class you wish to remove, and press the OK Button. ***NOTE: The class must be completely empty before it may be removed. (i.e. there may not be any commands associated with this class) See section 3.2.1.4 for information regarding the removal of commands from a class.***

3.2.1.4 Scripting Values

To modify the actual command information within a class, select the following:
Mode->Script
Attributes->Specify Values
Actions->Apply Mode & Attribute

When the user becomes more familiar with the ***Sleuth*** Interface, s/he will realize that the some of the above options may not need to be selected. Until then, use the above method to modify Scripting Values. This action will cause a *Selection Dialog* to appear with a list of all the Current Scripting Classes. Either double-click on the class you want to modify, or single-click on it and press the OK Button.

This will cause a list of all the Commands associated with this Scripting Class to be displayed. There are four things the user may do with these commands:

- Add another Command to this Class
- Modify a Command's name
- Remove a Command from the Current Class
- Modify the Frequency Information for a Command in the Current Class

Adding, Modifying, and Removing Commands should be straightforward, just remember to Always Type the Command Names in ALL CAPS. If the user selects the **Modify Element**

or **Remove Element** options, nothing will appear to happen. The user should move the mouse so that the cursor appears in the "Command Area." When Modifying an Element, the cursor will change to a "+". This indicates Modify Mode. The user selects a command to modify by clicking with Mouse Button 1 DIRECTLY OVER THE NAME OF THE COMMAND. The same is true for Remove Mode, except that the cursor changes to a Skull and Crossbones.

Modifying the Frequency information is a little more interesting. Select: **Actions->Frequency Definition**. The Frequency Dialog will appear asking for a Command Name. Type the command name in ALL CAPS in the Command Text Area, and either click on the **Apply** button, or press [RETURN]. This will show the current Frequency information for this command in the "Frequency" Text area. Type in a new value for this command by removing the current value in the Frequency Text Area and the keyboard to type in the new value. Either click the **Accept** button or press [RETURN] to accept the new value. Selecting **Close** will not save the new value you typed in.

The value in the Frequency Text Area simply means that the command listed in the Command text area will be **x** *times* more likely to be selected than a command with a frequency of "1" where **x** is the new number you type in.

3.2.1.4 Scripting Rules

To modify the scripting rule information, select the following:

Mode->Script
Attributes->Specify Rules
Actions->Apply Mode & Attribute

The commands that have Scripting Rules associated with them will be displayed in the Command Area. Select: **Actions->Rule Definition** to view/add/modify/remove scripting rules.

To View a Scripting Rule, type the Command Name (that appears in the Command Area) in ALL CAPS in the "Rule Name" text area and press [RETURN] or click on **Read**. The current rule for that command will be displayed in the "Rule Value" text area. Currently, these rules take the form:

INIT_COMMAND <n/<class>> TERM_COMMAND

where:

INIT_COMMAND is the initiating command of the "bracket."
<n/<class>> is a generation of **n** commands from the <class> scripting class.
TERM_COMMAND is the terminating command of the "bracket."

A "bracketed" command simply means that, if the Command for which this rule applies was generated (see section 5.1.1), place the commands in the following order:

INIT_COMMAND

<n/<class>>
TERM_COMMAND

Furthermore, if INIT_COMMAND and TERM_COMMAND share parameters, the values for the parameters will be equivalent as well (for the Mount/Dismount example, you want to dismount the same tape you mount).

To Remove the Rule at this point, select **Remove**.
To Change the Rule at this point, make the corrections and select **Write**
To Add a rule, change the "Rule Name" and "Rule Value" and select **Write**
To End the Rule Definition session select **Close**

3.2.2 Command Specification

There are basically three operations available for Changing the Specification of Commands: Command Values, Command Rules, and Removing Commands from the Specification. There are also two ways to get at this information:

From the ***Sleuth*** Interface, select: **Specification->Command**
From the Specification Interface, select: **Mode->Command**

3.2.2.1 Command Values

Command Values are the actual Syntax Diagrams (railroad diagrams) for the commands. These may be edited using the Syntax Diagram editor (see section 6.0). To select a command for modification, select:

Mode->Command
Attributes->Specify Values
Actions->Apply Mode & Attribute

A *Selection Dialog* will appear with a list of all the commands currently in the Specification. Either double-click on the name of a command, or single-click on a command, and select **OK**.

3.2.2.2 Command Rules

Currently, the user probably does not want to mess with "Command Rules" since they are highly complex. Suffice it to say that a Command Rule has been defined for the MOVE Command and may be viewed using the same techniques discussed in section 3.2.1.4.

3.2.2.3 Removing Commands From the Specification

To Remove a Command from the Domain Specification, select: **Tools->Remove Command**. This will display a list of all the commands in the Domain Specification. Using Mouse Button 1, single-click on the command you wish to remove, and the select **OK**.

3.2.3 Parameter Specification

The Parameter mode of the Specification combines both Rules and Values in one step. This is because the Rules and Values actually appear within the same file. To access this, use one

of the following:

From the ***Sleuth*** Interface, select: **Specification->Parameter**
From the Specification Interface, select: **Mode->Parameter**

To select a parameter to modify, select;

Mode->Parameter
Attributes->Specify Rules *or* **Attributes->Specify Values**
Actions->Apply Mode & Attribute

This will display a list of the current parameters of the Domain Specification in a *Selection Box*. Using Mouse Button 1, double-click on the parameter you wish to modify. This will display the current information for the parameter.

The Parameter File contains the following information:

1. Value values for the parameter
2. Inheritance information (i.e. what values can constraint the choices for this parameter)

Figure 7 shows an example parameter file:In this figure, the following should be noted. The

```
##############################
# this file contains information relating to the FOO parameter
##############################
DEFAULT: 00 01 02 03 04 05 06 07;
SET1: 00 01 02;
SET2: 03 04 05;
SET3: 06 07;
SET4: DEFAULT - SET3;
SET5: SET2 + SET1;
CATCHALL: &DEFAULT;
%%
BAR = 001 => SET1;
BAR = 002 => SET2;
BAR = 003 => SET3;
BAR = 004 => SET4;
BAR = 005 => SET5;
%%
```

Figure 7. Sleuth Parameter File Format

"DEFAULT" value should appear in ALL PARAMETER FILES. This is the set from which values will be selected if no other set applies. The *Values* portion of the file is terminated with the string "%%" and the *Rules* section is also terminated with the string "%%". These termination string should appear whether there are inheritance rules or not. Now, the *Values* syntax of the file will be discussed:

 # <anything> denotes a COMMENT.
<SET> : val1 val2 ... valN ; denotes a SET of values. The values will pertain to the named set and must be terminated with a semicolon.
<SET1> - <SET2> denotes Set Difference
<SET1> + <SET2> denotes Set Union
<SET1> * <SET2> denotes Set Intersection
&<SET> denotes Set Copy.

Next, we will discuss the *Rules* syntax of the file:

<VAR> = <Val> => <SET> ;

has the following meaning:

When parameterizing a command template
If the variable <VAR> has been previously selected
If the VALUE of <VAR> was <Val>
Then use <SET> to choose values for the current parameter.

Using figure 7, if a command template reads:

MOVE <bar-id> TO <foo-id>

and is currently parameterized as:

MOVE 001 TO <foo-id>

the only values we can use for <foo-id> will be: 00, 01, and 02.

When you have completed the necessary changes to this file, select **Update**. If you wish to abandon the changes you've made to this point (since you initiated the Parameter Edit Dialog or since the last time you selected **Update**), select **Close**.

3.3 Saving and Restoring Specification

3.3.1 Saving Specification

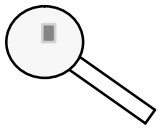
Sometimes it becomes necessary to save the Current Domain. This could be the case when:

- You are about to change the specification and want to have a Backup copy "just in case"
- The software you are testing is bumping a version (say from 1.2 to 2.0) and you wish to save the Current Domain for Regression Testing.

This is accomplished by selecting (From the ***Sleuth*** Interface): **Specification->Save Specification**. This will cause a *File Selection Dialog* to appear on the screen. In the "Selected Specification" text area, use Mouse Button 1 to click immediately AFTER the rightmost "/" and type the name you wish to save the Specification to. Now, either press [RETURN] of select **OK**.

3.3.2 Restoring a Specification

It is equally likely that you will need to re-load a Specification that you had previously saved. This is accomplished via the selection: **Specification->Load Specification**. Again, a list of the previously saved specifications will appear in a *File Selection Dialog*. Using Mouse Button 1, double-click on the Specification (in the "Files" area) that you wish to load. You will be asked if you really want to load the specification or not. Select **YES** if you do and **NO** if you do not.



4.0 The Configuration of a Domain

4.1 About the Configuration

Recall that the *Configuration* of a Domain is a user's LOCAL copy of the Domain Specification. Changes may be made freely to this without fear that it will interrupt another user's work.

4.2 What Can I do to my Configuration?

There are basically seven choices for modifying your configuration:

- Turn Commands On/Off within a Scripting Class
- Modify Frequency Information for a Command within a Scripting Class
- Turn Scripting Rules On/Off
- Modify Scripting Rules
- Modify the Syntax Diagram for a Command
- Modify Parameter Information
- Reset/Save/Load Configuration

4.2.1 Scripting Configuration

To access this information, from the ***Sleuth*** Interface, select: **Configuration->Script**. This will initiate the Configuration Module (which will take some time to appear, so be patient). Here, the user will be able to modify Scripting Values and Scripting Rules.

4.2.1.1 Scripting Values

In Domain Configuration, modifying Scripting Values simply means that the user can either: Turn Commands within a class On or Off, or modify the Frequency Information for a Command.

To turn a command within a scripting class On or Off (i.e. If the Command is OFF, it will not be generated when the user issues the Generation Command. See section 5.1.1), select:

Mode->Script
Attributes->Configure Values
Actions->Apply Mode & Attribute

This will display a list of the current Scripting Classes in your Configuration. Select the class you wish to modify in the standard way. Each command in the selected class will be displayed in the Command Area. Each command is a *Toggle Button*. When the command is selected (ON), the square to the left of the command will be dark, whereas if it is not selected (OFF), the square will be a lighter color. Clicking (with Mouse Button 1) on a Selected Command will change its state to OFF, while clicking on an Unselected Command will change its state to ON. When you are happy with the state of the commands within this class, select **Apply Settings**.

See section 3.2.1.4 for information regarding changing the Frequency Information for a command. The same information applies here,except that you will be updating your LOCAL copy, not the Global Specification.

4.2.1.2 Scripting Rules

Basically, the information in section 3.2.1.4 applies here except that you will be changing your LOCAL copy, not the Global Specification. Also, you may turn rules On and Off temporarily using the method described in section 4.2.1.1.

4.2.2 Command Configuration

See section 3.2.2 for information regarding the modification of Commands. The same information applies here except:

- You will be changing your LOCAL copy, not the Global Specification
- You cannot Remove Commands

4.2.3 Parameter Configuration

See section 3.2.3 for information regarding the modification Parameters. The same information applies here except that you will be changing your LOCAL copy, not the Global Specification.

4.3 Reset/Load/Save Configuration

4.3.1 Resetting your Configuration

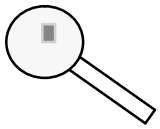
Resetting your Configuration basically means that you want to copy the information in the Domain Specification over to your local copy of the Domain (i.e. your LOCAL copy of the Domain). From the ***Sleuth*** Interface, select: **Configuration->Reset Configuration**. You will be asked to confirm this action.

4.3.2 Saving your Configuration

At various stages of your testing endeavors, you may find it necessary to save the changes you've made in your LOCAL copy. To do this, from the ***Sleuth*** Interface, select: **Configuration->Save Configuration**. See section 3.3.1 for further details.

4.3.3 Loading a new Configuration

If you find the need to load a configuration that you had previously saved, you may do this via the Load Configuration option. From the ***Sleuth*** Interface, select: **Configuration->Load Configuration**. See section 3.3.2 for further details.



5.0 The Testing Process

In the previous four sections, you learned how to navigate various interfaces to set up the Domain so that it reflects the way you want to test. We turned commands On and Off, we turned rules On and Off, we modified values for parameters, etc. Now we are ready to generate some test cases.

5.1 The Scripting Window

The Scripting Window allows for three types of actions:

- Generation of Commands
- Including of Mega Files
- Merging of Mega Files

5.1.1 Generation Of Commands

5.1.1.1 Hand-Crafted Generation

To generate a list of commands, single-click with Mouse Button 1 in the Text Area of the Scripting Window. In ***Sleuth***, the "@" sign is the Generation Symbol. The following is the syntax required for generating a list of commands:

@n/<class>

where "n" is an integer number (1..50)

<class> is the scripting class you want the commands to derive from.

Some examples are shown in Figure 8. After you have typed in the correct syntax to generate these commands, select **Expand** under the Scripting Window to actually generate the commands. This list may now be saved as a Mega Script via the **Export** button under the Scripting Window.

5.1.1.2 Using the Interface to Generate Commands

The menu item, *Utilities->Generate Commands From Class* may also be used to generate commands. This option is recommended if the user becomes confused about the command generation syntax or wishes to generate several different generation commands.

Once Utilities->Generate... has been selected, the **Command Generation** window will appear. The two areas of interest are the: *Number To Generate* and the *From Class*

```
@10/Any      - Generate up to 10 commands from the Scripting Class "Any"
@25/Mode     - Generate up to 25 commands from the Scripting Class "Mode"
@50/ALLOC  - Generate 50 ALLOC Commands
```

Note that a COMAND NAME is a special instance of a Scripting Class.
If there are Scripting Rules active, they may affect the total number of commands actually generated. Recall that there is a generation command embedded within a Scripting Rule (see Section 3.2.1.4), so the total number may be far greater than you imagined. Turning Scripting Rules off (See section 4.2.1.2) will alleviate this problem.

Figure 8. Generating Command By Hand

fields. The user may enter any number desired (less than 50 provides the best results) in the *Number To Generate* field. Once the user presses [RETURN], the *Current Command* will be updated to show what the actual command will look like once the user presses the *Generate* Button.

The *From Class* field is an Option Menu that may be used to select what class the commands should be generated from. When the correct command appears in the *Current Command* field, the *Generate* button should be pressed. When this happens, the *Current Command* field will be transferred to the Scripting Window.

5.1.2 Including Mega Files

The generic term "Mega File" is used to denote a MegaScript, MegaCommand, or MegaParameter. Regardless of the type of Mega* file you choose to include, the syntax to generate this list will be placed in the Scripting Window. Furthermore, only MegaScripts will be resolved in the Scripting Window, MegaCommands in the Command Window, and MegaParameters in the Parameter Window. This will become clear with actual use.

To include a MegaScript, select: **File->Include MegaScript**. A list of the currently saved MegaScripts will be presented in a *File Selection Dialog*. Using Mouse Button 1, double click on the MegaScript you wish to include in your test. Suppose you selected *foobar.ms*, the syntax: ***%include_foobar.ms*** will appear at the top of the Scripting Window. To actually place the contents of this file into your test, select **Resolve** under the Scripting Window.

5.1.3 Merging Mega Files

Merging files is a special case of Including Mega Files in the sense that, if you only merge a single file, the net result will be an "include" of that file into the Scripting Window. However, Merging is much more powerful than simply including a file because if multiple files are se-

lected (which is the intended usage of this option), the contents of these files will be *Interleaved* in the window in which they are Resolved. To initiate a merge of some MegaScript files, select: **File->Merge MegaScript**. Eventually, a list of all the saved MegaScripts will be displayed in a *Selection Dialog*. Using Mouse Button 1, single click on each MegaScript you wish to Merge together. Select **OK** when you are satisfied with the list (clicking on a file that is already selected will de-select it). Assume you clicked on *foo.ms* and *bar.ms*, when you select OK, the syntax: ***%merge_foo.ms_bar.ms*** will appear in the Scripting Window. To actually perform the merge, select **Resolve** underneath the Scripting Window.

5.1.4 Comments about Including and Merging

If you *include* or *merge* either MegaCommands or MegaParameters and select **Resolve** under the Scripting Window, nothing will happen. Mega* files may only be *Resolved* in the window they are associated with; therefore, you will need to generate a list of command templates, THEN select **Resolve** under the Command Window to see the results of actions applied to MegaCommands. Similar of MegaParameters.

5.2 Generating The Command Templates

At this point, you have a list of commands in the Scripting Window and you wish to generate a list of command templates associated with these commands. Simply select the **GENERATE (->)** button located between the Scripting and Commands windows.

5.3 Generating Command Names from Command Templates

It is sometimes advisable to generate the list of command names associated with the Command Templates. To do this, select the **DE-SCRIPT (<-)** button between the Scripting and Command windows.

5.4 The Commands Window

Now you should have a list of Command Templates in the Commands Window. If you have any MegaCommands that need *Including* or *Merging*, select **Resolve** under the Command Window now.

If you wish to save the current list of command templates as a MegaCommand, select **Export** under the Commands Window.

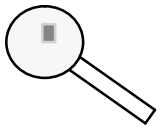
If you wish to Edit or simply View the contents of the Commands window in an "easier-to-use" context, select **View** under the Commands Window.

5.5 Generating Parameterized Commands

Once you have the list of Command Templates the way you like them, you will need to generate parameter values for the parameter PLACEHOLDERS that occur in the templates. Sim-

ply press the **GENERATE (->)** button between the Commands and Parameters windows. This operation takes some time due to the lengthy computations needed to construct values for the parameters, so please be patient. If you have an MegaParameters the need *Including* or *Merging*, select **Resolve** under the Parameter Window now.

If you wish to save the current list of parameterized commands (and you should, this is the end result of your testing), select **Export** under the Parameters Window.



6.0 The Syntax Diagram Editor

The Syntax Diagram Editor has been designed to be as user-friendly as possible. The following sections describe its usage.

6.1 Navigating the Syntax Diagram Editor

Click Mouse Button 3 anywhere inside the editor for the Pop up menu of options that you may select from.

6.2 Editor Options

6.2.1 Create New Diagram

This option will remove all information in the current diagram, thus providing a *clean slate* from which to work. This option is useful if you have just finished a diagram, saved it, and want to start another without leaving the editor.

6.2.2 Load/Save Syntax Diagram

If you are not editing the diagram you want to be editing, use the *Load Syntax Diagram* option to load in a different diagram. If you have completed editing of a diagram, choose the *Save Syntax Diagram* option to save it to disk. If you do not save your diagram, the changes you have made will not be reflected when generating command templates.

6.2.3 Add Command

A **command** is a structure of the form:

NODE----------END
NODE----------RETURN

The first command in the syntax diagram MUST end with the word **END**. If it does not, you may have problems when generating command templates. Subsequent commands must end with the word **RETURN**. You should think of the first command as the **Main Function** of a computer program. When this syntax diagram is parsed, the first command will initiate the command template generation.

Furthermore, the first word (i.e. the Name of the command) is **not** generated as part

of the command template. So if the Command "Alloc" should contain the word "Alloc," the command should be constructed as follows:

Alloc-----Alloc----------END

The subsequent commands (i.e. the ones that end in RETURN), should be thought of as **Subroutines** of a computer program. They are **called** by creating a Node with the label selected to be the name of sub-command enclosed in square brackets. The following is an example:

CMDA---------- [CMDB] ----------END
CMDB---------OPTION1---OPTION2------RETURN

When CMDB is called, it's command line will be parsed until the RETURN node is encountered. Then, the creation of the Command Template will resume following the call to CMDB.

6.2.4 Add Node

A **node** is a *Terminal* or *Non-Terminal* of your command language. Terminals of your language will be created by simply entering the Terminal-literal as the name for the node. For example, if the word **Alloc** is to be used in the command template, you should enter **Alloc** as the name.

A non-terminal of your language would be either a Parameter or a call to a Sub-Command. For sub-commands, see section 6.2.3. A Parameter is entered as follows:

[<param_name>-<param_type>]

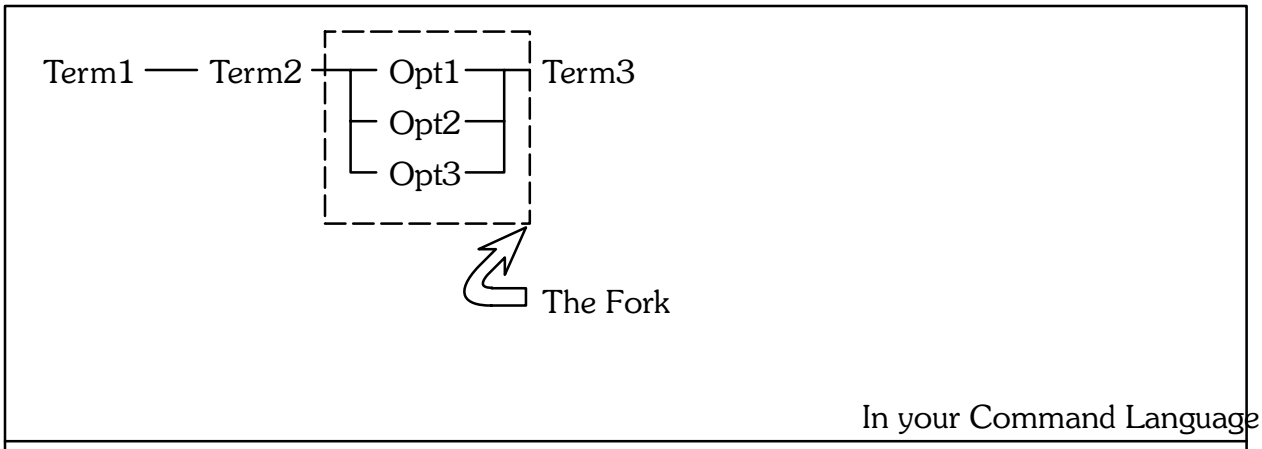
where
<param_name> is the name of the parameter
<param_type> is either: **id, ls,** or **rg**
id - Generate a single instance of the parameter
ls - Generate a comma separated list of the parameter
rg - Generate two parameters separated by a hyphen

6.2.5 Add Fork/Add Option

Fork and Options define decision structures in your command language. An Option is a special case of the Fork where the first choice is defined to be epsilon (the empty choice). You want to use a Fork structure if your language defines a **choice** of some sort. Figure 9 shows an example.

6.2.6 Modify Element

If you have incorrectly entered the name for a node, you should use this option to change it. When you select *Modify Element* from the menu, your cursor will change into the shape of a *Question Mark*. This indicates that ***Sleuth*** is asking you which Node you want to modify. Click directly on the text of the node that you want to change.



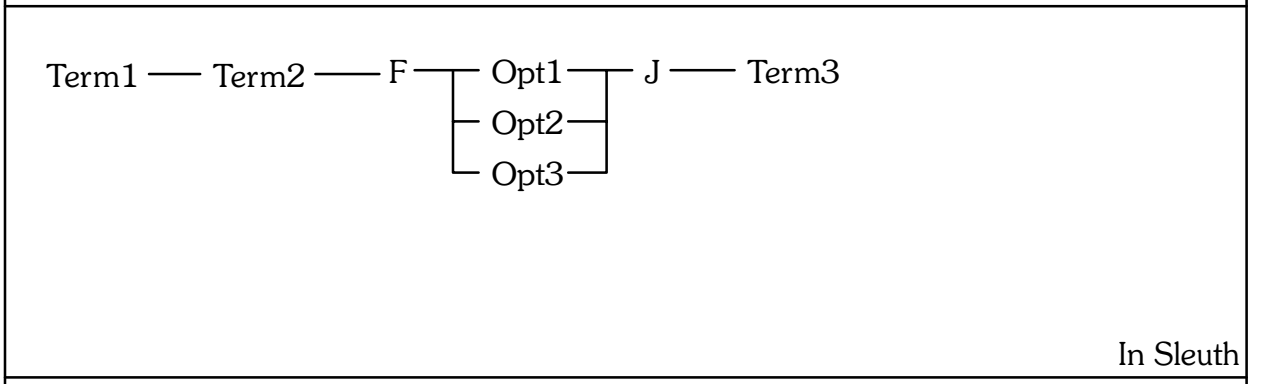


Figure 9. A Decision Structure

6.2.7 Cut Element

If you have decided that you no longer need a node in the diagram, use this option to remove it. Once you have selected *Cut Element* from the menu, the cursor will change into the *Skull And Crossbones*. This indicates that ***Sleuth*** is asking you to select the node you wish to change.

If you cut a Fork/Option node (i.e. an **F**), every node until the adjoining Join (**J**) will be removed. This is handy for removing the entire Fork/Option structure in one operation. Similarly, if you cut a Command node, every node until the adjoining End/Return will be removed. This is useful for removing an entire sub-command from the diagram.