USER'S GUIDE

T-SCOPE

Test Data Observation and Analysis System

Ver 3.1

SOFTWARE RESEARCH, INC.

Table of Contents

List of Figures

Preface

This chapter explains the basics of *T-SCOPE*, including its role in testing and how it fits in with its companion STW / Coverage tools

1 The QA Problem

It is a sad fact of the software engineering world that, without coverage analysis tools, only around 50 percent (on average) of the source is actually tested before release. With little more than half of the logic actually covered, many bugs go unnoticed until after release.Worse, the actual percentage of logic covered is unknown to SQA management, making any informed management decisions impossible.

Questions such as when to stop testing, or how much more testing is required are not answered on the basis of data but on ad hoc comments and sketchy impressions. Software developers are forced to gamble with the quality of the released software and make plans based on inadequate data.

A related problem is that test case development is done in an inefficient manner, that is many test cases are redundant. Cases testing the same logic clutter test suites and take the place of other cases which would test previously unexplored logic. Often testers are unsure of the direction to take and can waste SQA time devising the wrong tests.

2 The Solution

The primary purpose of testing is to ensure the reliability of a software program before it is released to the end user. To ensure a reliable and solid software product, the software should be thoroughly tested with a variety of input to provide statistically verifiable means of demonstrating the product's reliability. In other words, a suite of test cases should cover, in some way, all the possible situations in which the program will be used.

Although a worthy goal, imagining every possible use, as well as developing test data and running them, is extremely complicated and time-consuming. A more realistic goal is to test every part of the program.

According to industrial studies, achieving this goal yields significant improvement in overall software quality. Coverage analysis improves the quality of your software beyond conventional levels.

3 SR's Solution

Software Research, Inc. offers a solution: the *STW/Coverage* tool suite.STW / Coverage ensures tests are more diverse than those which are chosen by reference to functional specification alone, or are chosen based on a programmer's intuition. *STW/Coverage* ensures tests are as complete as possible, by measuring against a range of high quality test metrics:

- *C1*, or branch / segment coverage, measures module testing at the unit or module testing level; it accesses the completeness of individual modules or small groups of module testing.
- *S1*, or call-pair coverage, measures all the interface of a complex system to be exercised.
- *Ct*, or equivalence class coverage, measures the number of times each path or path class in a module is exercised.

With the three test metrics, STW / Coverage ensures tests are as complete as possible. STW / Coverage includes the following products:

- *TCAT* does coverage at the logical branch (or segment) level and the call-graph level. It employs the *C1* metric. You can choose to test a single module, multiple modules or the entire program using the C1 metric.
- *S-TCAT* does coverage at the call-pair level. It employs the *S1* metric. After individual modules have been tested, you can test all the interfaces of the system using the S1 metric.
- *TCAT-PATH* does coverage at the logical path level. It employs the Ct measure. It can easily be programmed to include or to exclude the program's modules from analysis. This allows you to emphasize certain critical modules. Once these are identified, *TCAT-PATH* allows you to extract and display the logical conditions that will cause that particular path to be exercised. Based on these conditions, you can design new test suites to exercise the path.
- *T-SCOPE* provides dynamic visualization of test attainment during unit testing and system integration. It is a companion tool for *TCAT* and *S-TCAT*. While these tools report the status of modules after-the-fact, T-SCOPE visually demonstrates such things as segments and call-pairs hit or not hit while it is happening.

T-SCOPE is the focus of this manual. For complete information on use of the other STW / Coverage products, please consult the proper user guides.

4 Format of Manual

The manual is divided into five chapters:

- T-SCOPE OVERVIEW discusses what *T-SCOPE* is and how it is used.
- QUICK START demonstrates running a basic *T-SCOPE* session.
- UNDERSTANDING THE INTERFACE defines *T-SCOPE*'s GUI features.
- GUI OPERATION explains how to use the X Window System graphical user interface menu.
- CUSTOMIZING T-SCOPE describes the Xdefaults file, where you can change *T-SCOPE* GUI defaults.

CHAPTER 1

Quick Start

This chapter is a tutorial and shows step-by-step how to run a basic *T-SCOPE* test session, from initial setup to viewing coverage reports.

1.1 Recommendations

It is recommended that you complete the instructions in this chapter before continuing to other sections. This chapter gives you a feel for how the system is organized and permits you to perform coverage analysis testing.

1.1.1 STEP 1: Instrumenting Your Source Code

T-SCOPE, as mentioned, is a companion tool for *TCAT* and *S-TCAT*. *T-SCOPE* allows you to visualize coverage as a program is executed. In order for it to work, you must have already have used *TCAT* or *S-TCAT* to instrument the source program, and compile the instrumented program to create an object file.

T-SCOPE takes over from there. It links the object file to its supplied runtime module to create an executable. As the program executes, *T-SCOPE* dynamically updates its charts to show you the exercised program parts of a program and coverage percentages.

For the first part of this tutorial, you will need to use *TCAT* to create an object code file.

1. Using the TCAT manual as reference, go to the *SR/demos/coverage/tcatC.demo* directory. A program named example.c should be listed there. This is our target program.
2. Instrument example.c to place special markers at each logical branch and then compile the instrumented version.

3. If you followed (2), you should have created the following files:

An object file named example.i.o. This will file will be linked with T-SCOPE supplied runtime object module to create an executable.

Three directed-graph files: main.dig, proc_input.dig and chk_char.dig. When used with *T-SCOPE*, these displays will dynamically display branches as they are exercised during execution.

4. Move these files over to your working *SR/demos/coverage/tscope.demo* directory. STEPs 3 through 8 of this tutorial show you how to display *C1* coverage.

In STEPs 9 through 11, you will be trying to display *S1* coverage. To create an object file:

1. Using the *S-TCAT* manual as reference, go to the *SR/demos/coverage/stcatC.demo* directory. A program named *example.c* should be listed there. This is our target program.
2. Instrument *example.c* to place special markers at each logical branch and then compile the instrumented version.
3. If you followed (2), you should have created the following files:
    - An object file named *example.i.o*. This file will be linked with *T-SCOPE* supplied runtime object module to create an executable.
    - A call-graph file named *example.i.P* which displays the caller-callee relationship of the *example.c* program.
4. Do not move these files over to the *SR/demos/coverage/tscope.demo* directory until STEP 9.

1.1.2 STEP 2: Starting Up T-SCOPE

Before you begin, make sure you are in the X Window System running a window manager (e.g. **mwm**, **olwm**, etc.)

You should start with the screen organized in a particular way, as shown in Figure 6.

Initialize an xterm-type window by using the mouse to click on **New Windows** or issuing the command

```
xterm &
```

from an existing window. The xterm window will serve as the *T-SCOPE* invocation window.

Move the window to the upper left of the screen. Go to the *SR/demos/coverage/tscope.demo* directory. This directory is supplied with the product, and it consists of an example C program named *example.c*.

This application allows you to select from several types of foods. By selecting various foods, you are actually exercising various logical branches (or segments) of the example program. The goal is to achieve the highest amount of *C1* (logical branch) coverage possible for this program through your input. The more selections you make, the higher the coverage.

When initiating this quick start session, your display should look like this:

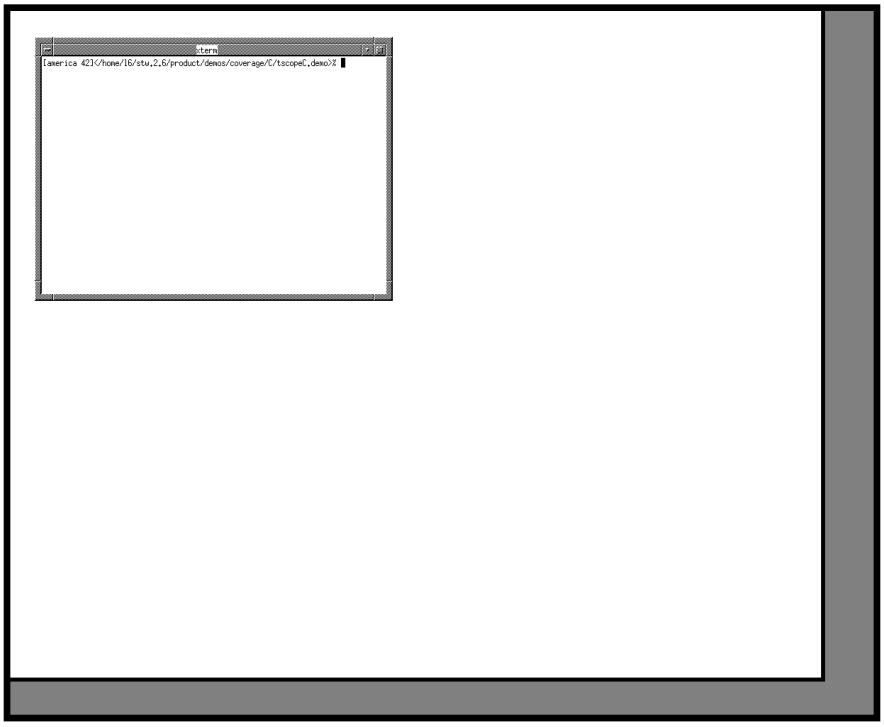

FIGURE 1 Setting Up the Display (Initial Condition)

1.1.3 STEP 3: Creating an Executable

When you used *TCAT*, you should have compiled the example.c program to create an object file named example.i.o. In this step you are going to link it with *T-SCOPE*'s supplied runtime object module *tsruntime.o*.

1. In the working *T-SCOPE* directory type:

```
cc -o a.out example.i.o tsruntime.o
```

2. This should create an executable named *a.out*.

When creating an executable, your display should look like this:

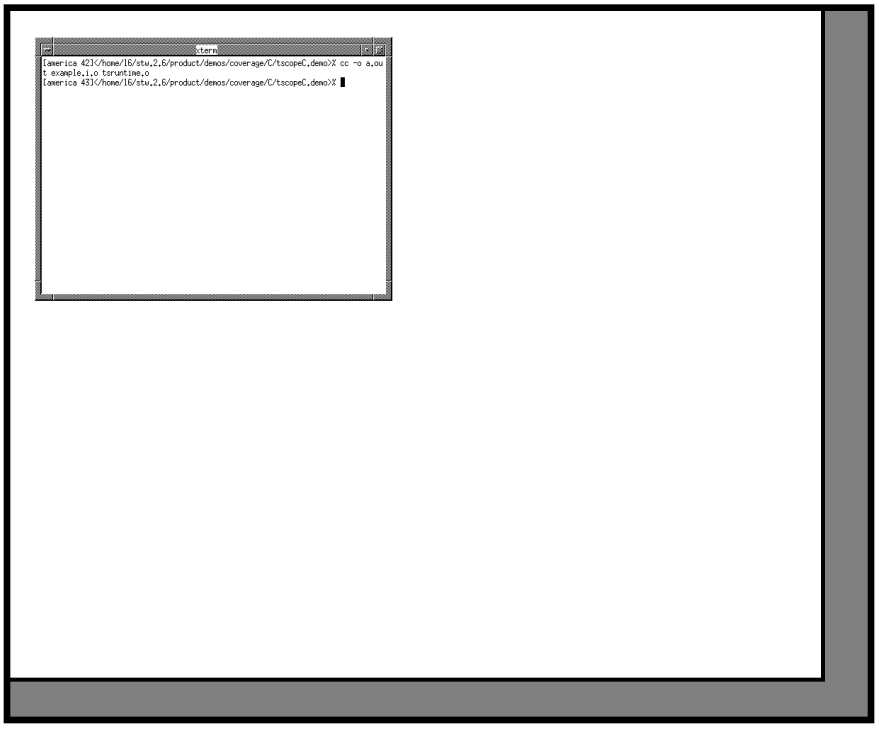

FIGURE 2 Creating an Executable

1.1.4 STEP 4: Invoking T-SCOPE

Now, invoke *T-SCOPE*.

1. Position the mouse pointer, so that it is located in the invocation window.
2. Activate it by clicking the mouse pointer on it. This window becomes the main control window. During your session, all status messages and warnings are displayed in this window.
3. To invoke *T-SCOPE*, type:

   ```
   Xtscope
   ```

4. When you type in this command, the **Main** window pops up.
5. Move the **Main** window to the upper right of the screen. You can move a window by clicking on its title bar and dragging it.
6. If you want to start over, you can terminate *T-SCOPE* by clicking on the **Exit** button.

When invoking *T-SCOPE*, your display should look like this:

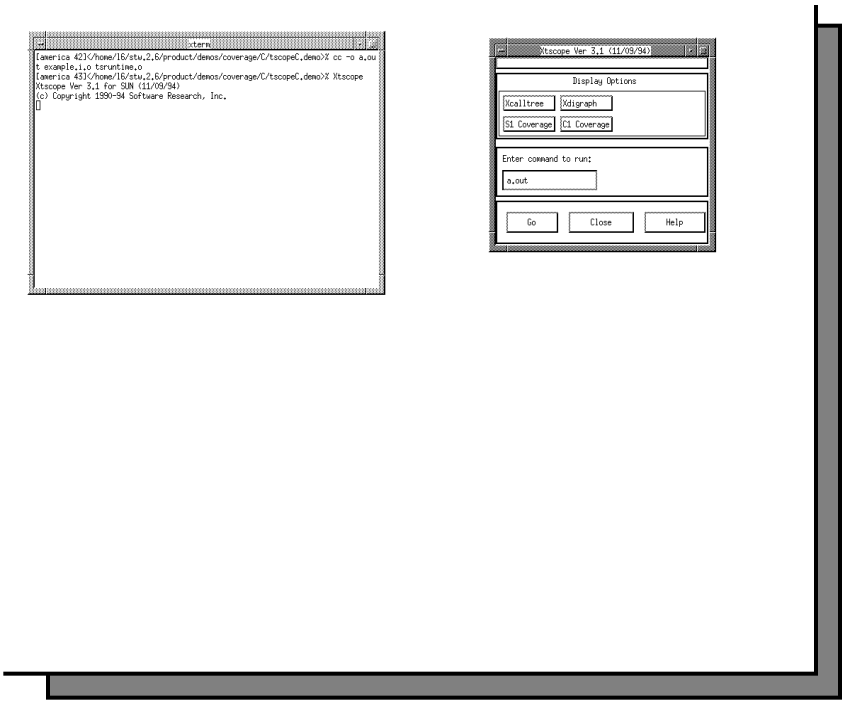

FIGURE 3 Invoking T-SCOPE

1.1.5 STEP 5: Selecting Directed Graphs

Once an executable is created, you can select a module's directed-graph and coverage chart. These displays will allow you to view coverage dynamically.

In this step, you are going to select the directed graph displays for each of example.c's modules: *main.dig*, *proc_input.dig*, and *chk_char.dig*:

1. Click on the **Xdigraph** button.
2. A **Digraph Selection Box** box pops up. The three directed graphs should be listed in the **Files** list box.
3. First, select *chk_char.dig*'s directed-graph display by:
   - Double clicking on *chk_char.dig* in the **Files** list box.
   - Or, highlighting *chk_char.dig* in the **Files** list box and selecting **OK**.
   - Or, typing *chk_char.dig* in the **Selection** entry box and selecting **OK**.
4. *chk_char*'s directed-graph pops up.
5. Select the directed-graph displays for *main.dig* and *proc_input.dig* just as you did for (3).
6. Arrange the three directed-graphs so that the invocation and **Main** windows are not covered. When you execute the *example.c* program, these windows must be clear.

When selecting directed-graphs, your display should look like the one below:

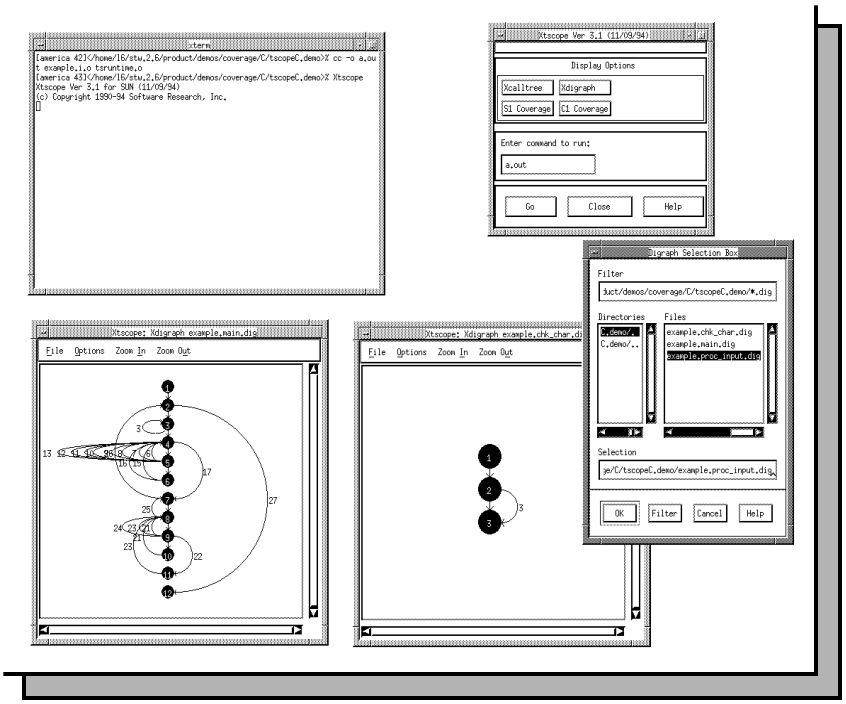

FIGURE 4 Selecting Directed Graphs

1.1.6 STEP 6: Selecting Coverage Charts

In this step, you are going to select the **C1 Coverage** charts for each of the program's modules. During a program's execution, these strip charts show the actual percentage of coverage obtained.

1. Click on the **C1 Coverage** button.
2. A **C1 Module Box** box pops up. example's three modules should be listed in the **Modules** list box.
3. First, select main by highlighting it.
4. Click on the **OK** button.
5. The ***C1*** value chart for *main* pops up.
6. Follow the steps in 1-5 for *proc_input* and *chk_char.*
7. Arrange the three value charts so that the invocations window and the **Main** windows are not covered up.

When selecting *C1* coverage charts, your display should look like the one below:

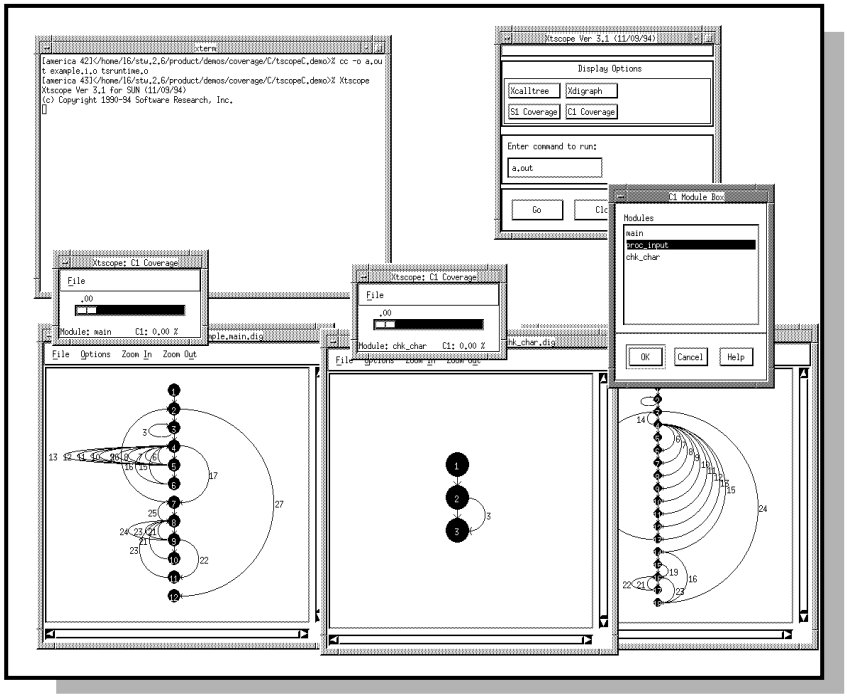

FIGURE 5 Selecting C1 Coverage Charts

1.1.7 STEP 7: Running the Application

At this point you should have an executable named *a.out* and all the displays available for *example.c* should be displayed on your screen. In this step you are going to run that executable. By running the program, you will be exercising the program and watching the directed-graphs and *C1* coverage charts dynamically update.

This application is designed to ask you which type of food in the San Francisco, CA area you would like to eat. By selecting particular types of food, you are actually exercising creating test cases to exercise the program's logical branches. The more combinations you select, the more branches you will exercise.

To run the application:

1. Make sure the **Enter command to run** specification region says **a.out** for the name of your executable.
2. Click on the **Go** button in the **Main** window.
3. The **Main** window's options will gray out and the program will start up in invocation window.
4. It prompts you:

   ```
   What type of food would you like?
   ```

5. In order to get the most coverage from this test case run, type in

   ```
   1 2 3 4 5 6 7 8
   ```

   for the eight types of foods listed and press **Enter**.
6. Eight restaurants that reflect the eight types of food you selected will be displayed, the directed-graphs dynamically display the exercised logical branches, and the value charts update the percentage of coverage.

   In the directed-graph, please take note of the following:

   - Thick lines signify logical branches that have been exercised.
   - Thin lines signify logical branches that have not been hit. On color monitors these lines are represented by the color yellow.

   For color monitors only:

   - The color pink represents the most recently executed logical branch.
   - The color yellow represents a logical branch that has been hit less than five times.
   - The color green represents a logical branch that has been hit between five and 15 times.

- The color red represents a logical branch that has been hit more than fifteen times.

The default colors and lower and upper thresholds amounts can be set for each directed-graph in the **Digraph Options** window. Please refer to Section 3.2.4.2 for further information.

7. After the restaurants are displayed, the program prompts you:

```
Do you want to run it again?
```

During an ordinary testing situation, you would normally run the application a couple of times, selecting various combinations of food types. For now, however, just type in **n** for no. You'll soon have plenty of opportunities to run several test cases.

8. The final coverage percentages for each should be:

- *chk_char* = 66.66%
- *main* = 55.56%
- *proc_input* = 62.50%

An ideal testing situation warrants around 85 percent or higher*C1* coverage. In the case of *example.c* you could rerun the program using various test case selections.

When running the application, your display should look like this:

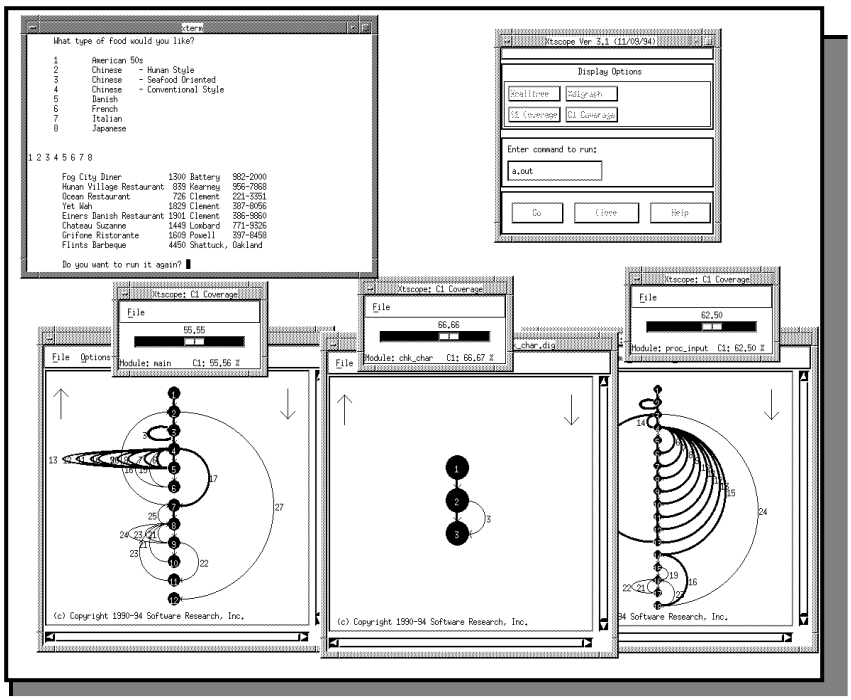

FIGURE 6 Running the Application

1.1.8 STEP 8: Cleanup

To complete the session for *C1* coverage:

1. Close the directed-graphs and **C1 Coverage** charts by clicking on the **File** menu and selecting **Exit**.
2. Close the **Main** window by clicking on the **Exit** button.

When completing the *C1* session, your display should look like this:

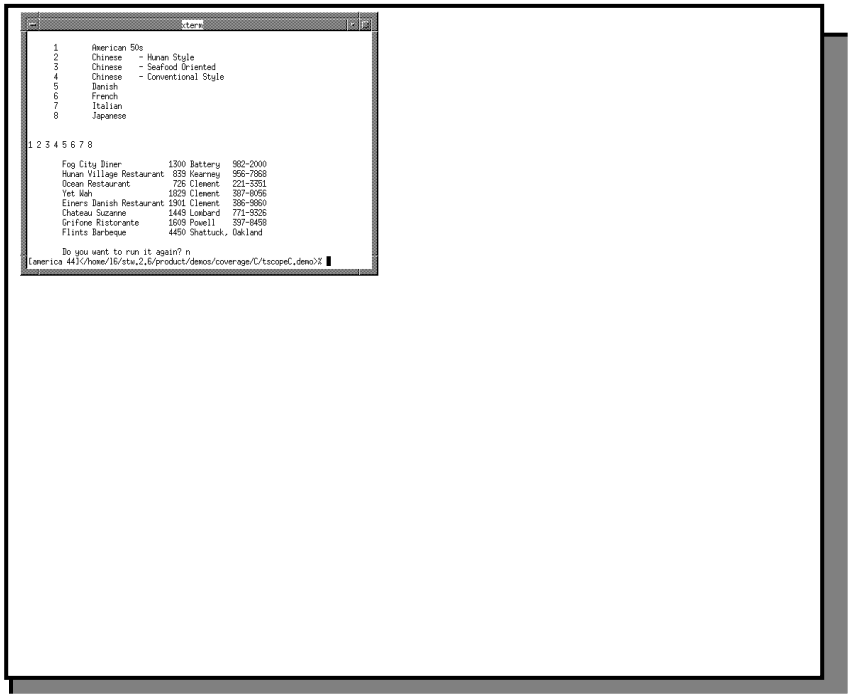

FIGURE 7 Completing a C1 Coverage Session

1.1.9 STEP 9: Setting Up for S1 Coverage

In this step you are going to set *T-SCOPE* up to work for *S1* (call-pair) coverage.

1. In STEP 1 of this tutorial you should have instrumented for call-pair markers and then compiled that instrumented program to create two files named *example.i.o* and *example.i.P*. Copy these files to your current working *T-SCOPE* directory.
2. Create an executable named *a.out* by typing:
   ```
   cc -o a.out example.i.o tsruntime.o
   ```
3. Invoke T-SCOPE by typing:
   ```
   Xtscope
   ```
   and then move the window to the upper right hand corner of the screen.
4. Click on the **Xcallgraph** button.
5. A **Call Graph Selection Box** pops up.
6. Select the *example.i.P* call-graph.
7. The *example.i.P*'s call-graph pops up.
8. Click on the *S1* Coverage button.
9. A **S1 Selection Box** pops up.
10. Select the *example.i.P* file.
11. The **S1 Coverage** chart pops up. During program execution this chart will display the percentage of coverage achieved for each executed test case.
12. Arrange the call tree and the value chart so that the invocation and the **Main** windows are not covered.

When preparing for dynamic *S1* coverage, your displays should look like this:

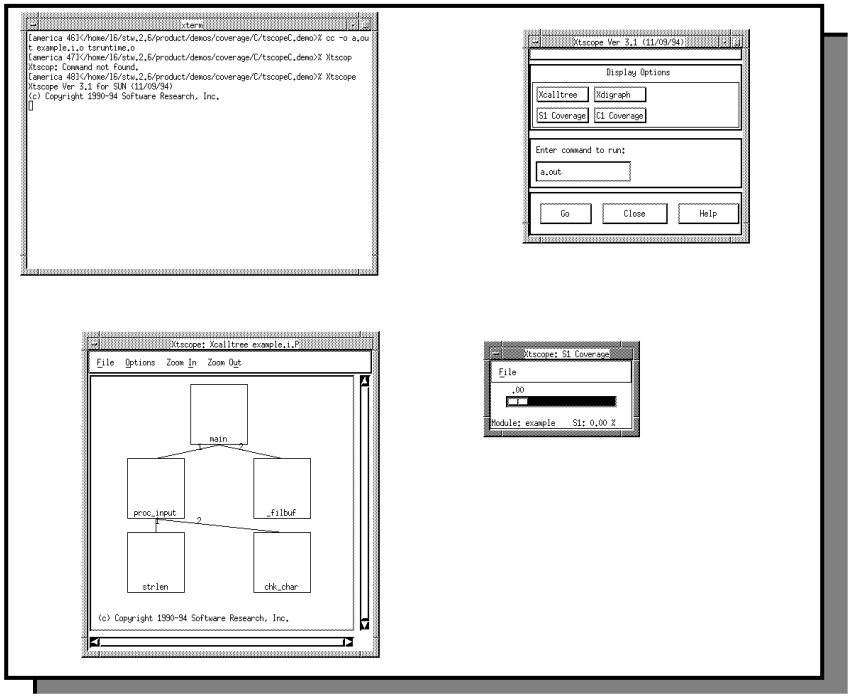

FIGURE 8 Preparing for S1 Coverage

1.1.10 STEP 10: Running the Application

To run the application:

1. Make sure the **Enter** command to run specification region says **a.out** for the name of your executable.
2. Click on **Go** button in the **Main** window.
3. Run the program just as you did in STEP 7, making sure to select all eight types of food listed.
4. As the call-graph updates its exercised call-pairs, please take note of the following:
   - Thick lines signify call-pairs that have been exercised.
   - Thin lines signify call-pairs that have not been hit. On color monitors these lines are represented by yellow.

   For color monitors only:
   - The color pink represents the most recently executed call-pair.
   - The color yellow represents a call-pair that has been hit leashing five times.
   - The color green represents a call-pair that has been hit between five and 15 times.
   - The color red represents a call-pair that has been hit more than fifteen times.

   The default colors and lower and upper thresholds amounts can be set for a call-graph with the **Call Graph Options** window. Please refer to Section 3.2.1.2 for further information.
5. When the program prompts you if would like to run the application again, type in **n**.
6. The *S1* coverage value should be 28.57 percent. For a program to be adequately exercised, *S1* coverage should be 90 percent or higher. *S1*'s goal is try to exercise all of the interfaces of a program, which means strategically planning effective test cases beforehand.

After running the application, your display should look like this:

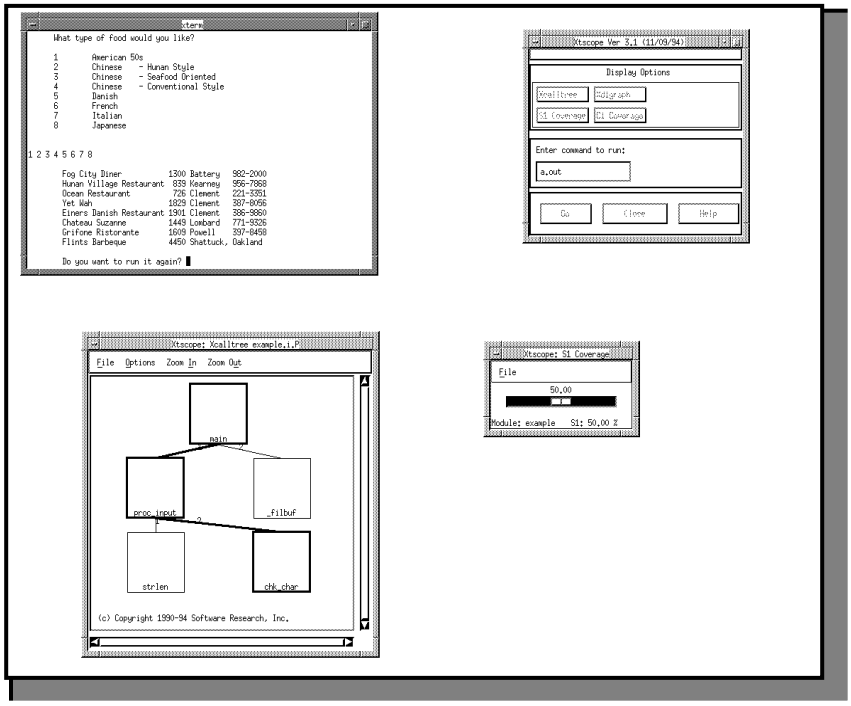

FIGURE 9 Running the Application

1.1.11 STEP 10: Cleanup

To complete the session for *C1* coverage:

1. Close the call-graph and *S1* Coverage chart by clicking on the **File** menu and selecting **Exit**.
2. Close the **Main** window by clicking on the **Exit** button.

When completing the *S1* session, your display should look like this:

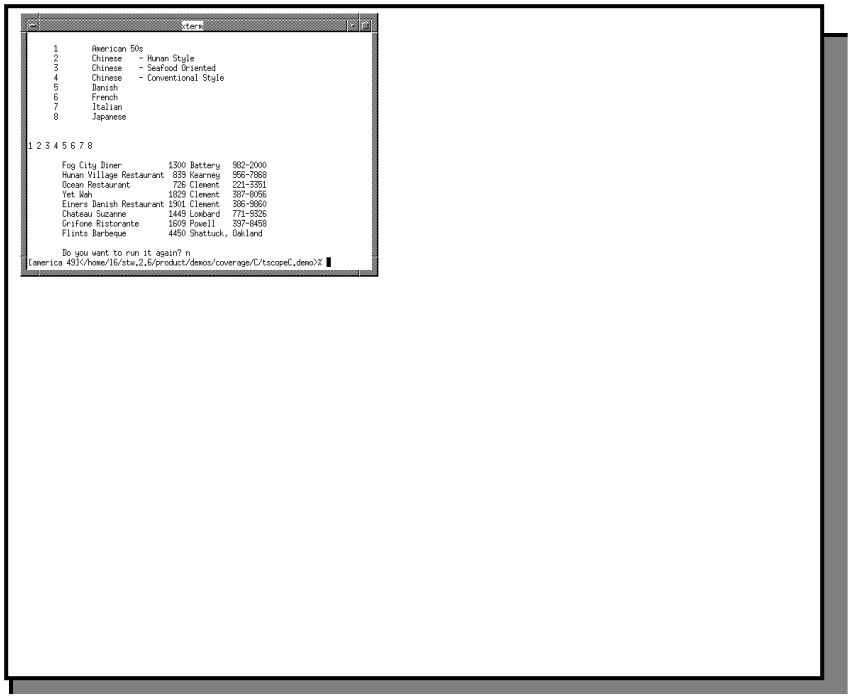

FIGURE 10 Completing a S1 Coverage Session

1.1.12 Summary

If you successfully completed the preceding 11 steps, you've seen and practiced the basic skills you need to use *T-SCOPE* productively.

In this chapter you should have learned how to:

- Link the created *TCAT* or *S-TCAT* runtime object file with *T-SCOPE*'s supplied runtime module.
- Invoke *T-SCOPE*.
- Select various displays.
- Run an application and watch the displays dynamically update.

For best learning, you may want to:

- Repeat STEPS 1 - 11 without the manual and experiment by running the application several times and looking at the amount of coverage your test input receives.
- Repeat STEPS 1 - 11 with a small application of your own.

CHAPTER 2

Understanding the Interface

This chapter covers the basic X Window System graphical user interface operations of T-SCOPE. It demonstrates using *T-SCOPE* from the OSF / Motif X Window System.

2.1 Basic OSF/Motif User Interface

This section demonstrates using the file selection dialog boxes, help menus, message dialog boxes, option menus, and pull-down menus.

If you are familiar with the OSF / Motif GUI style, you can go on to Section 3.2.

2.1.1 File Selection Windows

T-SCOPE's file selection windows allow you to select a directed-graph display or a call-graph display.

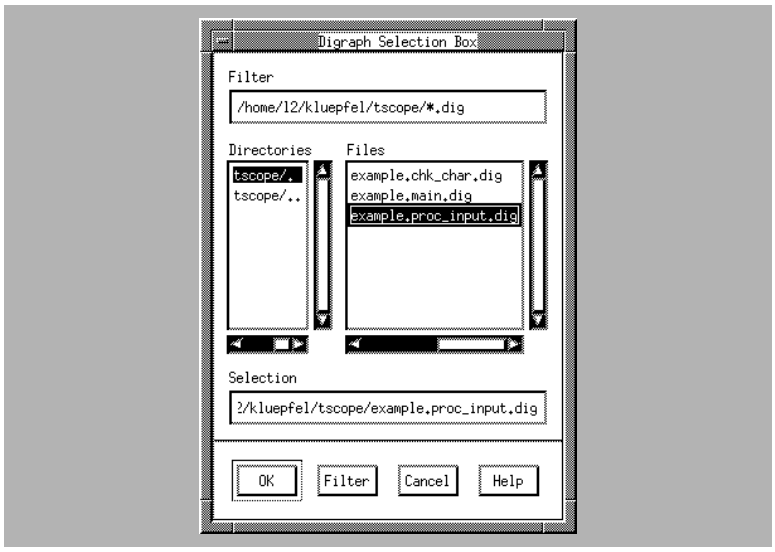

FIGURE 11 File Selection Window

Filter entry box	Specifies a directory mask. When you click the Filter-push button,the directory mask is used to filter files or directories thatmatch this mask (or pattern).
Directories list box	Lists directories in path defined in the **Filter** entry box. Use it to locate the desired directory.
Files list box	Lists files in the path defined in the **Filter** entry box.
scroll bars	Move up / down and side / side in the **Directories** and **Files** list boxes. You use them to search for the appropriate directory or file.
Selection entry box	Selects and enters a file name.

Use the three push buttons at the bottom of the dialog box to issue commands:

OK	Specifies a directory mask. Accepts the file in the **Selection** entry box as the new file or the file to be opened and then exits the dialog box.
Filter	Applies the pattern you specified in the **Filter** entry box. It lists the directories and files that match that pattern.
Cancel	Cancels any selections made and then exits the dialog box. No file is selected as a result.

To use a file selection dialog box:

1. You can restrict the file selection operation to a named region(directory path) by typing in a directory path name in the **Filter** entry box or by clicking on a path name in the **Directories** entry box. Then click on the **Filter** push button.
2. To select a keysave file name, do one of these three things:
    - Double click on the file in the **Files** entry box.
    - Highlight the file in the **Files** entry box or type in the file name in the **Selection** entry box and click on **OK**.
    - Highlight or type in the file name and press the **Enter** key.

2.1.2 Help Windows

T-SCOPE provides on-line help for each of its windows.This on-line help brings up the text corresponding to where you invoke the help. In other words, if you invoke it at the Main window, the Help window displays information pertinent to Main window. Here's how to use the help.

1. Click on the **Help** button.
2. The **Help** window pops up with text corresponding to the point at which it was invoked.
3. You can use the scroll bars to move up / down and side / side. If you don't see what you need, you can search for specific text. To do this:

   Click on the **Action** menu and select the **Search** option. A dialog box (shown below) pops up.

   - Type in the pattern you want to search for and then click on OK or press the **Enter** key.
   - If the pattern is found, then the window automatically scrolls to the location of the specified pattern.
   - If you select another **Help** button from another window while the current one is displayed, the Help window scrolls to the content of the new window.
   - To close the window, click on **Action** and select **Exit**.

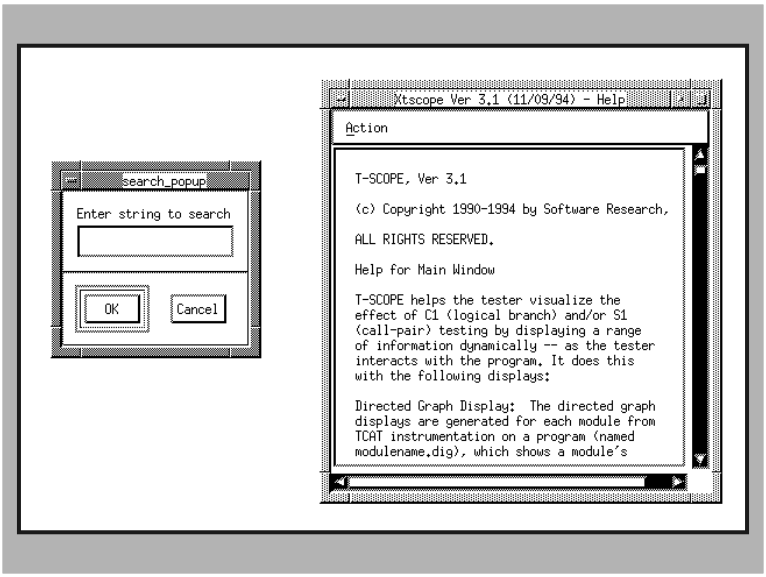

FIGURE 12 Search Pop-up/Help Window

2.1.3 Message Boxes

Pop-up message dialog boxes have three purposes:

- They display warnings and error information.
- They ask you to verify that you want to perform a task.
- They ask you to enter a command.

To remove a message box after you have read it or to tell *T-SCOPE* to go ahead with a command, click on the **OK** button. If you want to cancel a command, click on the **Cancel** button.

FIGURE 13 Message Box

2.2 Main Window Features

All the functionality necessary to operate *T-SCOPE* is accessible from the **Main** window.

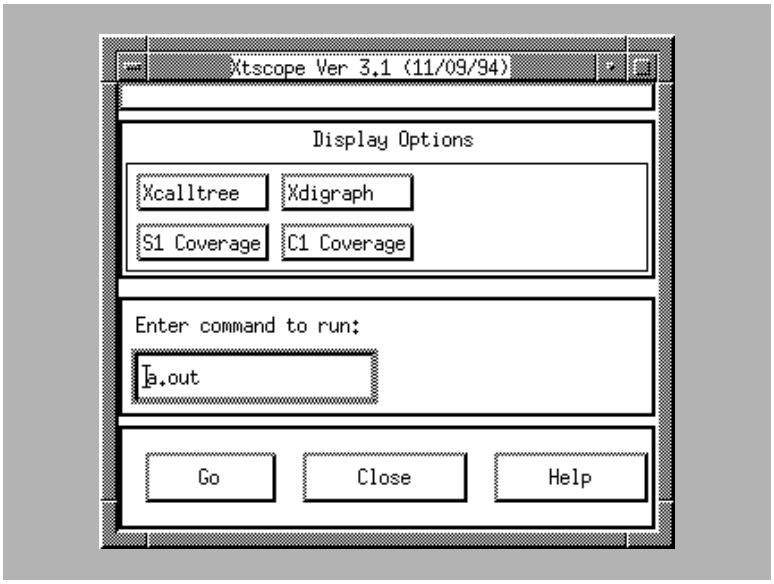

FIGURE 14 Main Window

It includes the following features:

- **Xcallgraph** button: Selects a program's call tree display.
- **S1 Coverage** button: Selects a program's S1 value chart.
- **Xdigraph** button: Selects a program's directed-graph displays.
- **C1 Coverage** button: Selects a program's C1 value chart.
- **Enter command to run:** specification region: Specifies the name of a program's executable.
- **Go** button: Executes the named executable in the **Enter command to run:** specification region.
- **Exit** button: Closes the Main window.
- **Help** button: Provides on-line help for the Main window.

2.2.1 Xcallgraph Button

The callgraph button brings up a Call Graph Selection Box from which you select a program's call-graph file, `filename.i.P`, generated from S-TCAT.

Shown below is a call-graph from our supplied example program named example.i.P. It displays example's functions and its call-pair connections during a test case's execution.

The features of the call-graph window are described next.

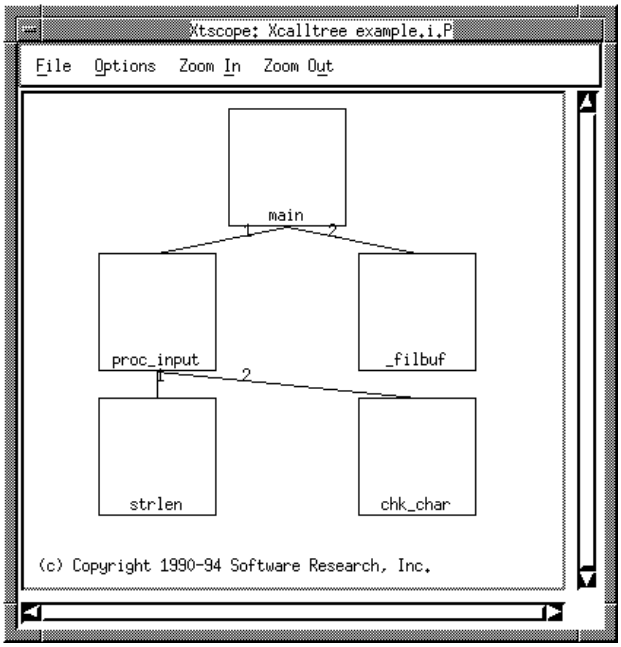

FIGURE 15 Call-Graph Display

2.2.1.1 File Menu

Exit: Closes the call-graph.

2.2.1.2 Options Button

The **Options** button invokes the **Call Graph Options** window, which allows you to adjust the geometry of a call-graph.

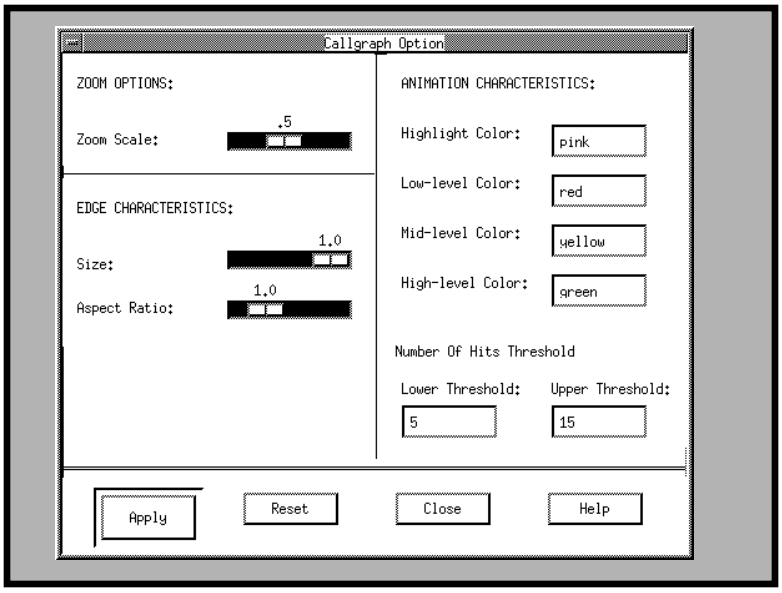

FIGURE 16 Options Window

At the bottom of the window, there are four buttons:

Apply This button applies information you change from the **Call Graph Options** window to the call-graph.

Reset This button sets the **Call Graph Options** window to the default settings.

Close This button exits the **Call Graph Options** window.

Help Displays on-line help for the **Call Graph Options** window.

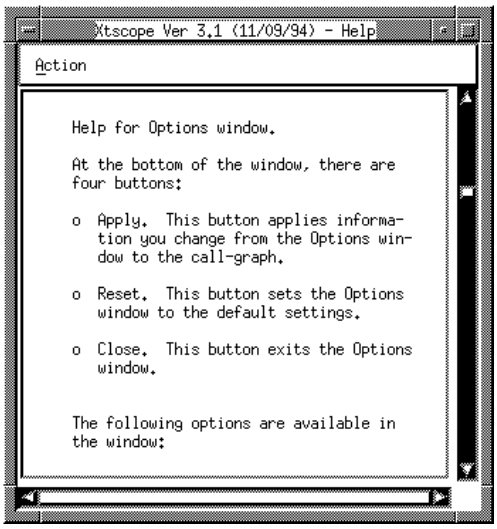

FIGURE 17 Help Window

The following options are available from this window:

- **ZOOM OPTIONS** change the percentage a call-graph zooms in or zooms out.
  - **Zoom Scale** corresponds to the magnitude a call-graph's **Zoom In** and **Zoom Out** buttons redraw the call-graph. The **Zoom Scale** default is set at .5 which magnifies the call-graph by 50 percent if the **Zoom In** button is used or reduce it by 50 percent if the **Zoom Out** button is used.

    Moving the slide ruler to the left decreases the zoom percentage; moving it to the right increases the zoom percentage.
- **EDGE CHARACTERISTICS** allow you to set the following options which apply specifically to the functions' appearance in the call-graph:
  - Size determines the size of the functions. The default is set to 1.0 which represents the real size of the functions.

    Moving the slide ruler to the left decreases the size of the-functions.
- **Aspect Ratio** changes the length to width ratio of a function. The default is set to 1.0. This default translates to a 1 to 1 ratio between the length and width.

Moving the slide ruler to the right decreases the height of thefunctions. Moving it to the left of 1.0 creates a 1 to <1 width to length ratio. Moving it to the right of 1.0 creates a 1 to >1 width to length ratio.

- **ANIMATION CHARACTERISTICS** options allow you to change the colors which reflect the dynamic coverage of a program.*T-SCOPE* dynamically shows the exercised call-pairs and their functions through colors.
  - **Highlight Color** reflects the color of the call-pairs and their functions that are the most recently executed in an executed test case. Default = pink.
  - **Low-level Color** reflects the color of the functions before test case execution and the color of the call-pairs and their functions during test case execution when call-pairs are exercised equal to or less than the number of times specified in the lower threshold. Default = yellow.
  - **Mid-level Color** reflects the color of the call-pairs and theirfunctions during program execution when call-pairs are exercised between the lower and upper thresholds. Default = green.
  - **High-level Color** reflects the color of the call-pairs and theirfunctions during program execution when call-pairs are exercisedequal to or more than the number of times specified in the upper threshold. Default = red.
  - **Lower Threshold** specifies the lower coverage threshold number.The default is 5. When a call-pair is hit five times or less,it is colored yellow or the color specified for the **Low-level Color**.
  - **Upper Threshold** specifies the upper coverage threshold number.The default is 15. When a call-pair is hit 15 times or more,it is colored red or the color specified for the **High-level Color**.

2.2.1.3 Zoom In Button

The **Zoom In** button reduces the display to the magnitude specified in the **Call Graph Options** window's **Zoom Scale** option.

Below is an example of call-graph zoomed in three times.

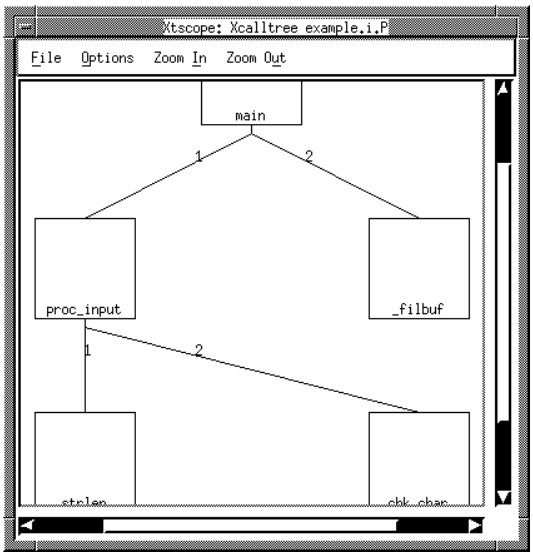

FIGURE 18 "Zoomed-In" Display

2.2.1.4 Zoom Out Button

The **Zoom Out** button undoes the last **Zoom In** applied.

NOTE: You can not **Zoom Out** or minimize the initial call-graph display.

2.2.2 S1 Coverage Button

S1 Coverage button brings up a **S1 Selection Box** where you can select a program's call-graph file generated from *S-TCAT*. When you select the call-graph file, a chart like the one below pops up.

After each test case for a program is executed, it updates to reflect the percentage of coverage achieved. It consists of a **File** menu that allows you to close the window.

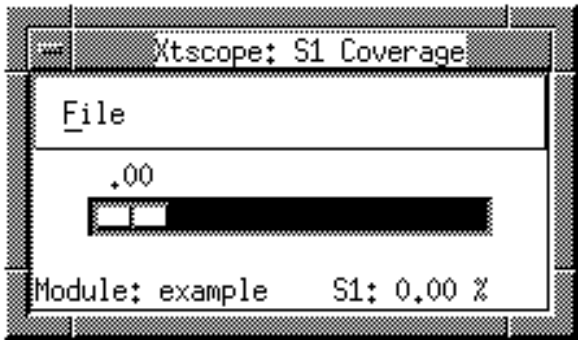

FIGURE 19 S1 Coverage Chart

2.2.3 Xdigraph Button

The **Xdigraph** button brings up a **Digraph Selection Box** from which you select a particular module's directed-graph that you want to see dynamically show exercised logical branches during test case execution. Directed-graph files are listed as *modulename.dig*.

Shown below is one of the directed-graphs from the supplied example program named main.dig during a test case's execution.

The **Xdigraph** window is described next.

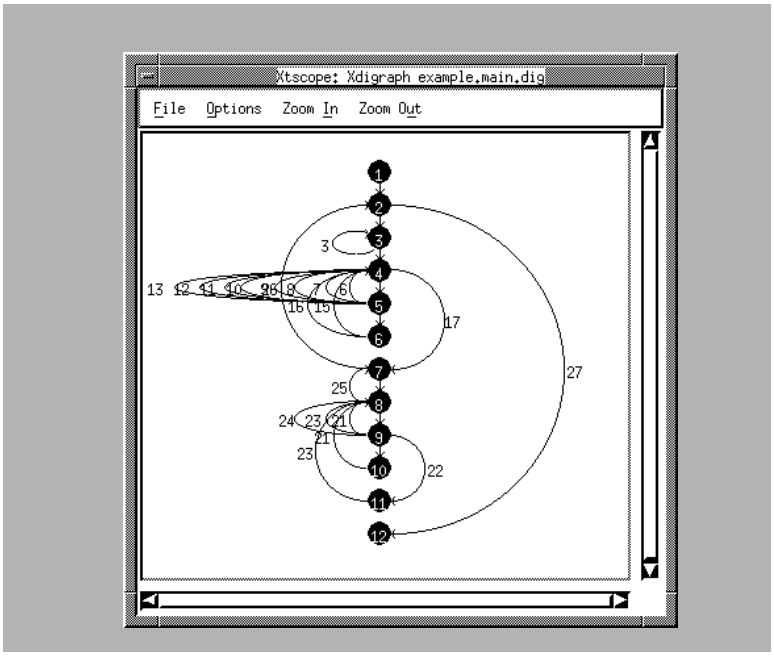

FIGURE 20 Directed Graph Display

2.2.3.1 File Menu

Exit: Closes the directed-graph.

2.2.3.2 Options Button

The **Options** button invokes the **Digraph Options** window, which allows you to adjust the geometry of a directed-graph's nodes and edges.

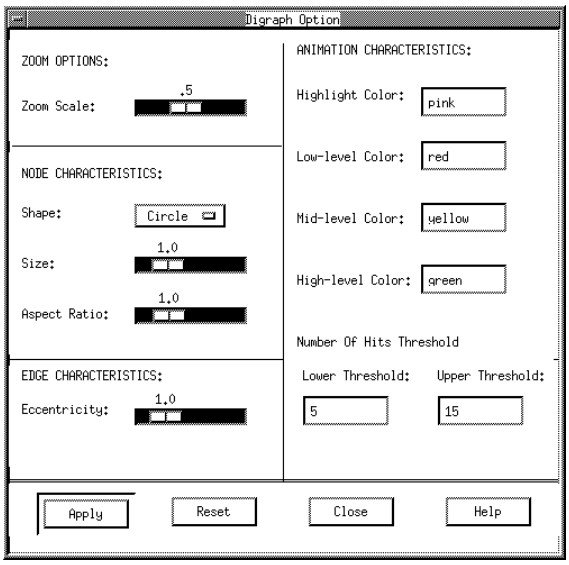

FIGURE 21 Options Window

At the bottom of the window, there are four buttons:

Apply	This button applies information you change from the **Digraph Options** window to the directed-graph.
Reset	This button sets the **Digraph Options** window to the default settings.
Close	This button exits the **Digraph Options** window.
Help	Displays on-line help for the **Digraph Options** window

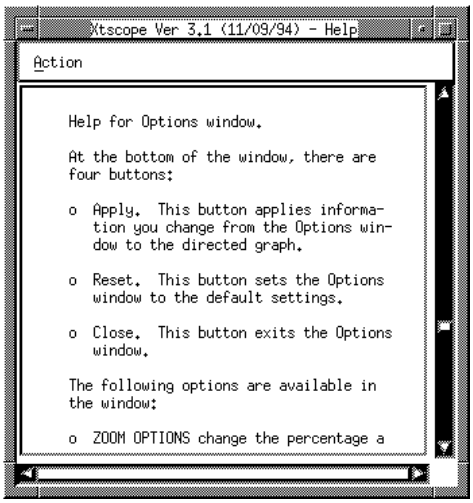

FIGURE 22 Help Window

The following options are available from this window:

- **ZOOM OPTIONS** change the percentage a directed-graph can zoom in or zoom out.
  - **Zoom Scale** corresponds to the magnitude a call-graph's **Zoom In** and **Zoom Out** buttons
  - redraw the directed-graph. The **Zoom Scale** default is set at .5 which magnifies the call-graph by 50 percent if the Z**oom In** button is used or reduce it by 50 percent if the **Zoom Out** button is used. To change the default:

    Moving the slide ruler to the left decreases the zoom percentage; moving it to the right increases the zoom percentage.
- **NODE CHARACTERISTICS** allow you to set the following options that apply specifically to the directed graph's nodes, or decision points.
  - **Shape** determines the shape of the node. Node shapes are defaulted to circles. Other available shapes include boxes ovals, or outlined circles. To change the default chape, click on the option menu and drag the mouse to the desired option.
  - **Size** determines the size of nodes. The default is set to 1.0.

Moving the slide ruler to the left decreases the size of thenodes; moving it to the right increases the size.

- A**spect Ratio** changes the length to width ratio of a node. The default is set to 1.0. This default translates to a 1 to 1 ratio between the length and width. A ratio different from 1 to 1 does not work for circles--only for oval or box-shaped nodes.

  Moving the slide ruler to the right decreases the height of the functions. Moving it to the left of 1.0 creates a 1 to <1 width to length ratio. Moving it to the right of 1.0 creates a 1 to >1 width to length ratio.

- **EDGE CHARACTERISTICS** allow you to change the edge shape, or logical branch shape.

  Eccentricity determines ellipse eccentricity of edges, chosen from a base value of 1.0 (the default) with a slide ruler.Moving the slide ruler to the left of 1.0 decreases the eccentricity; moving it to the right increase eccentricity.

- ANIMATION CHARACTERISTICS allow you to change the colors which reflect the dynamic coverage of a program. *T-SCOPE* dynamically shows the exercised edges.
  - **Highlight Color** reflects the color of the edge that is the most recently executed from an executed test case. Default = pink.
  - **Low-level Color** reflects the color of the edgesbefore test case execution and the color of the edges during test case execution when they are exercised less than the number of times specified in the lower threshold. Default = yellow.
  - **Mid-level Color** reflects the color of the edges during program execution when they are exercised between the lower and upper thresholds. Default = green.
  - **High-level Color** reflects the color of the edges during program execution when they are exercised equal to or more than the number of times specified in the upper threshold. Default = red.
  - **Lower Threshold** specifies the lower coverage threshold number.The default is 5. When a call-pair is hit five times or less,it is colored yellow or the color specified for the **Low-level Color**.
  - **Upper Threshold** specifies the upper coverage threshold number.The default is 15. When a call-pair is hit 15 times or more,it is colored red or the color specified for the **High-level Color**.

2.2.3.3 Zoom In Button"

The **Zoom In** button reduces the display to the magnitude specified in the **Digraph Options** window's **Zoom Scale** option.

Below is an example of directed-graph zoomed in three times.

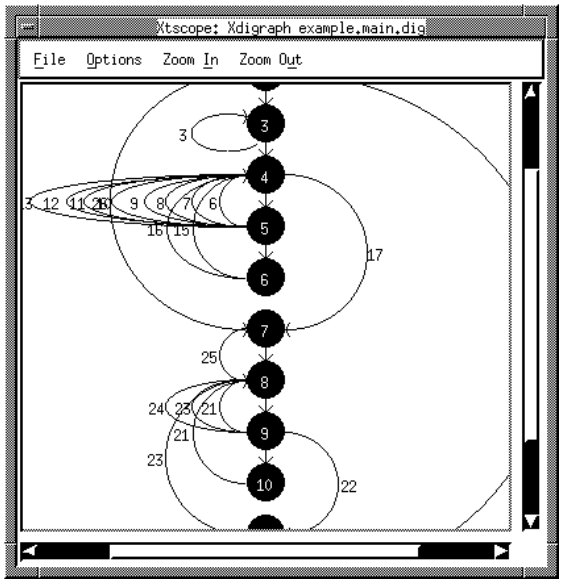

FIGURE 23 Zoomed In Display

2.2.4 Zoom Out Button

The **Zoom Out** button undoes the last **Zoom In** applied.

NOTE: You cannot **Zoom Out** or minimize the initial directed raph display.

CHAPTER 3

GUI Operation

This chapter covers the basic X Window system graphical user interface (GUI) usage of *T-SCOPE*.

3.1 Instrumenting Your Source Code

In order to dynamically view *C1* (logical branch) or *S1* (call-pair) coverage, you must use *TCAT* (for *C1* coverage) to instrument for logical branches or *S-TCAT* (for *S1* coverage) to instrument for call-pairs. Instrumentations modifies a program so that special markers are positioned at every logical branch or call-pair in each program module. Later, during program execution, these markers allow *T-SCOPE* to dynamically display when a branch or call-pair is exercised.

In order for *T-SCOPE* to understand the meaning of the markers, you must also compile the instrumented version of the program to create and object file which can then be linked with *T-SCOPE*'s supplied runtime object module. This runtime object module interprets the object file's instructions and creates an executable.

To instrument a program and compile it, follow the instructions in the user manuals for *TCAT* or *S-TCAT*. This creates the following files:

- An object file named example.i.o. This file will be linked with the *T-SCOPE* supplied runtime object module to create an executable.
- If you used *TCAT*: Directed-graph files for each program module (*modulename.dig*). Directed-graphs, as you may remember,display the control flow of a module.
- The directed-graphs displays are used with *T-SCOPE* to dynamically display branches as they are exercised during execution.
- If you used *S-TCAT*: A call-graph file for the instrumented program (*filename.i.P*). Call-graphs display the caller-callee function relationship of a program.

  Move these files over to your working *T-SCOPE* directory.

3.2 Creating an Executable

Now, to link the object file (*basename.i.o*) with *T-SCOPE* supplied runtime object module tsruntime.o to create an executable, execute the command:

`cc -o` *applicationname basename.i.o* **`tsruntime.o`**

- **`cc -o`** is the standard command to compile.
- *applicationname* is the name of executable you are creating.
- *basename.i.o* is the name of instrumented program object file.
- **`tsruntime.o`** is *T-SCOPE*'s supplied runtime objectmodule.

3.3 Invoking T-SCOPE

Once an executable is created, all that is left is to select the types of displays you would like *T-SCOPE* to dynamically update before executing the program. First, invoke *T-SCOPE*:

```
Xtscope
```

The **Main** window pops up.

If you used *TCAT* to create an executable, please go to Sections 4.4 and 4.5; if you used *S-TCAT,* please go to Sections 4.6 and 4.7.

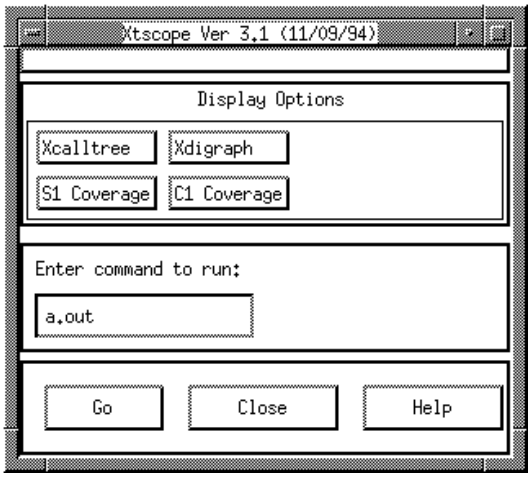

FIGURE 24 Invoking T-SCOPE"

3.4 Selecting Directed-Graph Displays

These instructions apply only if you created an executable with *TCAT*'s object file (*filename.i.o*). If you used S-TCAT, please refer to Section 4.6.

Once an executable is created, you can select a module's directed-graph and coverage chart. These displays will allow you to dynamically view *C1* coverage.

To select a directed-graph display:

1. Click on the **Xdigraph** button.
2. The **Digraph Selection Box** box pops up. All of your application's modules should have a corresponding directed-graph that was created during instrumentation listed in the **Files** list box.
3. Select a module's directed-graph display by:
   - Clicking on the directed-graph name in the **Files** list box.
   - Or, highlighting the directed-graph in the **Files** list box and then selecting **OK**.
   - Or, typing the directed-graph name in the **Selection** entry box and then selecting **OK**.

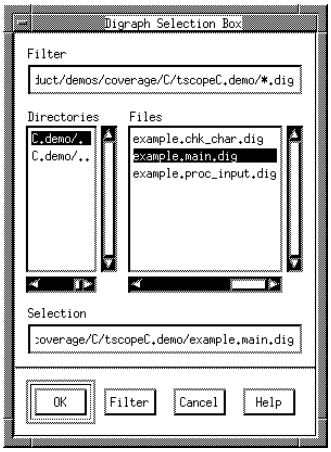

FIGURE 25 Selecting a Directed Graph Display

4. The directed-graph pops up. During test case execution, exercised logical branches will be represented by thick lines; unexercised logical branches will be represented by thin lines.

5. Select as many of you application's directed-graphs as you want as long as the **Main** window's **Go** button and the invocation window remain clear.These windows are needed to run your application.

   Because mid-size and large applications will have many directed-graphs, it is recommended that you display only those graphs that are essential. If you know, for instance, from using *TCAT* that your test cases exercised all of the logical branches for a particular module, than don't display that module's directed graph. The purpose of *T-SCOPE* is to not only determine which modules were not exercised, but also to determine how your test cases can be better improved to exercise all logical branches in a module.

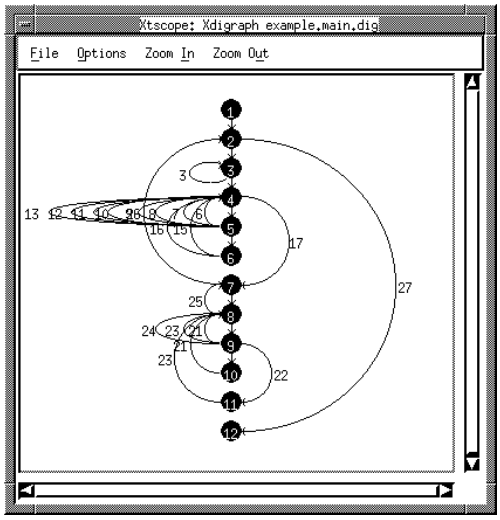

FIGURE 26 Directed Graph Display

3.4.1 Adjusting a Directed-Graph's Geometry

When you have a module's directed-graph displayed, you may want to change the threshold numbers and their colors, node or branch characteristics. You can do this by changing the defaults in the **Digraph Options** window. Simply click on the directed-graph's **Options** button and the window pops up. Please refer to Section 3.2.4.2 for further instructions.

If you want to make permanent changes to all of the directed-graph displays, you can edit the *SR* file. Please see Chapter 5 for further information.

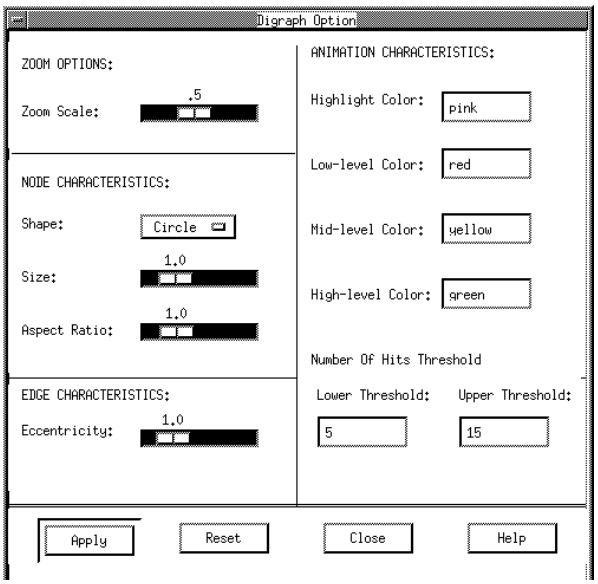

FIGURE 27 Using the Digraph Options Window

3.5 Selecting C1 Coverage Charts

Besides selecting a module's directed-graph, you can also select a module's *C1* Coverage chart once an executable is created. *C1* Coverage charts update the percentage of coverage achieved after each test case is executed.

To select a chart for a program's module(s):

1. Click on the *C1* Coverage button.
2. The **C1 Module Box** box pops up. All of your application's modules are listed in the **Modules** list box.
3. Select a module by highlighting it and then select the **OK** button.

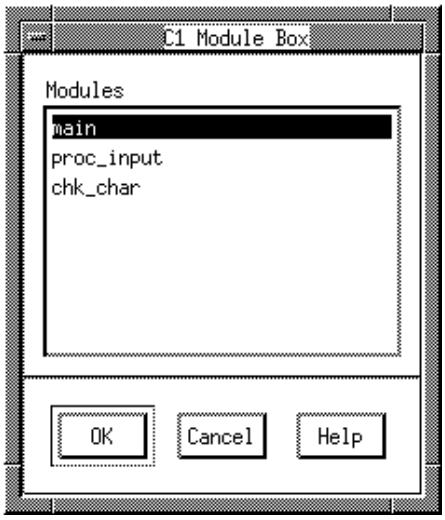

FIGURE 28 Selecting a C1 Coverage Display

4. The *C1* Coverage value chart for the module you select pops up.
5. Select as many charts as you want as long as the **Main** window's **Go** button and the invocation remain window clear.

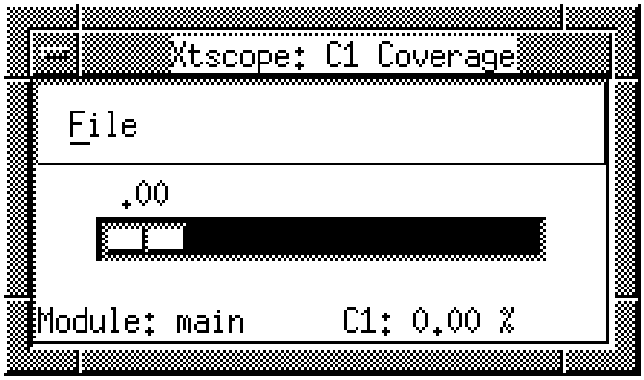

FIGURE 29 C1 Coverage Chart

3.6 Selecting Call-Graph Displays

These instructions apply only if you created an executable with *S-TCAT*'s object file (*filename.i.o*). If you used *TCAT*, please refer to Section 4.4.

This section explains how to select an applications call-graph file (f*ilename.i.P*). As you may recall from Section 4.1, this file is created from instrumentation. It represents a program's flow, or its caller-callee function relationship. During program execution, this display dynamically shows you the exercised the call-pair, allowing you to easily find the unhit call-pairs.

To select a call-graph display for *S1*:

1. Click on the **Xcallgraph** button.
2. The **Callgraph Selection Box** box pops up. Your application's call-graph file should be listed in the **Files** list box.
3. Select a module's call-graph display by:
    - Clicking on the call-graph name in the **Files** list box.
    - Or, highlighting the call-graph in the**Files** list box and then selecting **OK**.
    - Or, typing the call-graph name in the **Selection** entry box and then selecting **OK**.

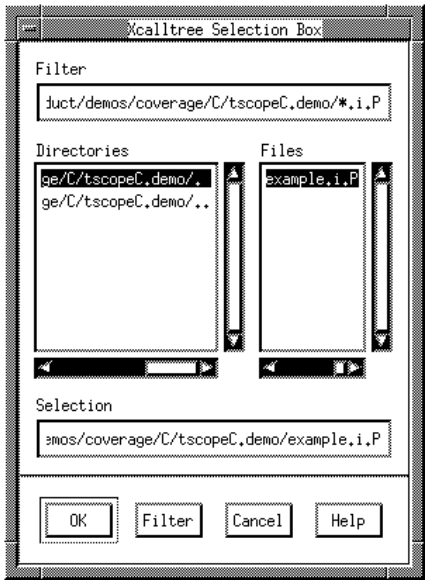

FIGURE 30 Selecting a Call-Graph Display

4. The call-graph pops up. During test case execution, executed call-pairs will be represented by thick lines.

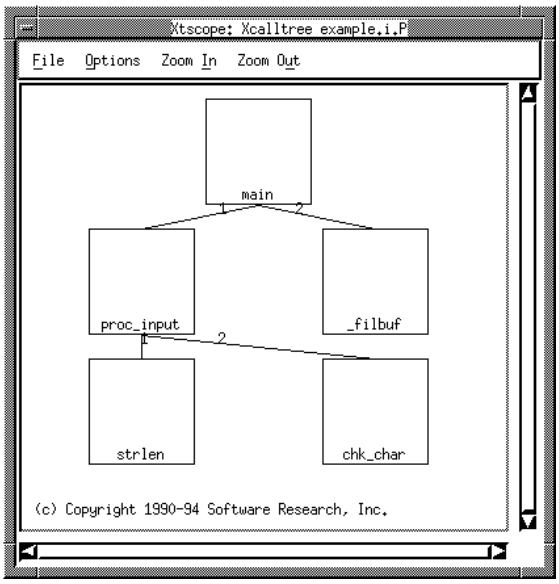

FIGURE 31 Call-Graph Display

3.6.1 Adjusting a Call-Graph's Geometry

When you have a program's call-graph displayed, you may want to change the threshold numbers and their colors, and function characteristics. You can do this by changing the defaults in the **Call Graph Options** window. Simply click on the call-graph's **Options** button and the window below pops up. Please refer to Section 3.2.1.2 for further instructions.

If you want to make permanent changes to all of the call-graph displays, you can edit the *SR* file. Please see Chapter 5 for further information.

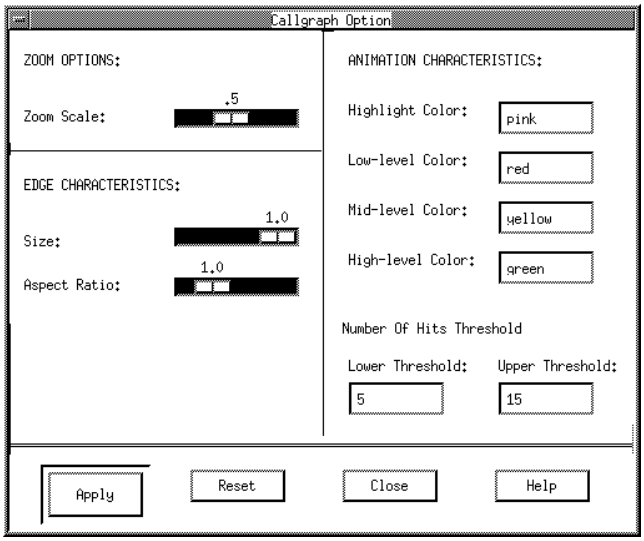

FIGURE 32 Using the Call-Graph Options Window

3.7 Selecting S1 Coverage Charts

Besides selecting a module's directed-graph, you can also select a module's **S1 Coverage** chart once an executable is created.The **S1 Coverage** chart updates the percentage ofcoverage achieved for the program after each test case is executed.

1. Click on the **S1 Coverage** button.
2. The **S1 Selection Box** box pops up. Your application'scall-graph file (filename.i.P) should be listed in the Files list box.
3. Select the file by:
   - Clicking on the call-graph name in the **Files** list box.
   - Or, highlighting the call-graph in the**Files** list box and then selecting **OK**.

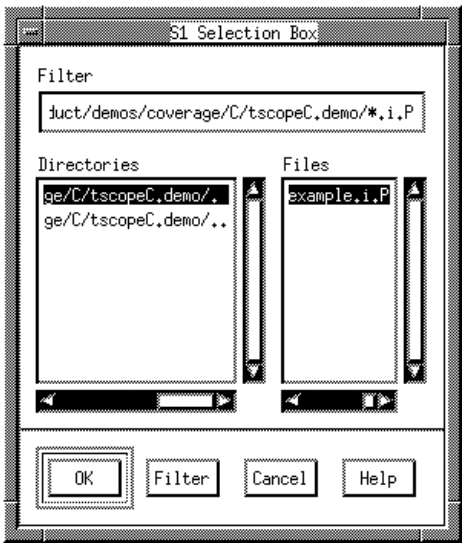

FIGURE 33 Selecting a S1 Coverage Chart

4. The S1 Coverage chart pops up. During your application's execution, this chart will dynamically update the percentage of call-pair coverage achieved after each test case is executed.

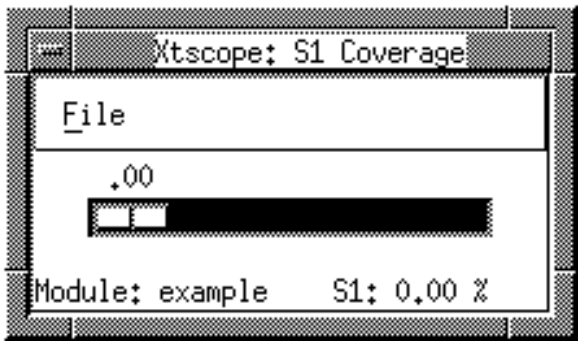

FIGURE 34 S1 Coverage Chart

3.8 Running Your Application

Once you have your application's directed-graph or call-graph and coverage charts displayed, you can run your application:

1. When you created an executable in Section 4.2 you should have created an executable named applicationname. Put *applicationname* in the **Main** window's **Enter** command to run specification region.
2. Click on the **Main** window's **Go** button.
3. Run your application just as you would normally.

   Unlike a regular run of your application, however, the instrumented version of your application displays hit logical branches or call-pairs as thick lines.

   For color monitors:

   - The color pink represents the most recently executed logical branch or call-pair.
   - The color yellow represents a logical branch or call-pair that has been hit less than five times or the number of times specified for the lower threshold.
   - The color green represents a logical branch or call-pair that has been hit between five and 15 times or the number of times specified for the mid-level threshold.

- The color red represents a logical branch or call-pair that has been hit more than 15 times or the number of times specified for the upper threshold.

These threshold colors and numbers can be set in the **Digraph Options** window or the **Call Graph Options** window.

CHAPTER 4

Customizing T-Scope

This chapter explains where the setup information is stored and gives instructions on changing it.

4.1 Location of Setup files

You can customize *T-SCOPE* by changing the X Window System resources or setup files. This chapter explains where the setup information is stored and gives instructions on changing it.Resource files setup files are text files. You can edit them with any standard UNIX text editor. Most of the graphical user interface defaults areset in the *SR* file supplied with the product. It needs to be put in the */ usr/lib/X11/app-defaults* directory. If you install *T-SCOPE* using the supplied installation script, the contents of the *SR*file are automatically copied or concatenated to the *SR* file in that directory.

Following is a list of the common GUI defaults. You can change the set defaults by manually changing the *SR* file to avoid resetting GUI parameters every time.

```
tscope*font: 6x13

tscope*hilightColor.value:                  pink
tscope*lolevelColor.value:                  yellow
tscope*normlevelColor.value:                green
tscope*hilevelColor.value:                  red

tscope*cgMinThreshold.value:                5
tscope*cgMaxThreshold.value:                15

tscope*dgMinThreshold.value:                5
tscope*dgMaxThreshold.value:                15

tscope*commandText.value:                   a.out

!
! options
!
tscope*zoomScale.value:                     5
tscope*nodeSize.value:                      10
tscope*nodeAspectRatio.value:               10
tscope*edgeEccentricity.value:              10
.fi
.in
```

Index of Terms

T

X