Automating Quality Assurance Testing of C++/Motif for the TSD Using AutoTest

Version 1.1

by John J. Xenakis

May 12, 2011

Revision History

Revision	Date	Authors Initials	Description of the Change

Table of Contents

1 Executive Summary

This technology was developed for the Traffic Situation Display (TSD), but the same technology can be used with any legacy software written in C++ using Motif.

Historically, quality assurance testing for the Traffic Situation Display (TSD) has been done manually rather than using automated testing tools. Manually-executed quality assurance testing is both time-consuming and error prone. TSD testers have not been able to use advanced testing methodologies like "test driven design and development" (TDD) that would improve the accuracy and speed of quality assurance testing.

The problem is exacerbated by the fact that graphical user interface (GUI) for the TSD is designed using Motif, a GUI (Graphical User Interface) API that uses old technology. Motif is difficult to program and the resulting GUI tends to be full of errors.

The problem is further compounded by the complexity of the TSD GUI; there are thousands of combinations of user inputs, so a complete system test can require days or weeks, and minor changes to the code can result in errors that might not be discovered for a long time. Using a scripted quality assurance test tool, it would be possible to run a complete set of quality assurance tests overnight, every night, to guarantee that no new bugs have been introduced during the previous day's programming. Automating quality assurance testing will save time, be more cost-effective and will produce more accurate testing results.

This paper describes the AutoTest quality assurance automated test system that has been developed for TSD. It has the following capabilities:

- The ability to execute user interactions through scripts. User interactions include clicking menu options, clicking buttons, filling and testing values in dialog fields, filling and testing values on widgets in XrtTable cells, and other standard operations on Motif widgets in the TSD GUI.
- The ability to perform memory stress tests and other tests on the TSD,
- The ability to identify memory leaks and other system problems.
- The ability to execute quality assurance testing with a scripting language that supports variables, C++-like expression evaluation, and subroutines, with varying verbosity levels of output.

AutoTest has already been used to identify and test numerous errors in the TSD, but more work is needed for it to be used to fully test the TSD. This document describes AutoTest functionality available today.

2 Control of dialog boxes

The TSD system uses over 100 dialogs and approximately 100 system commands, usually invoked by the TSD user from menu selections. AutoTest scripts can control or invoke any of these commands and dialogs.

This section explains how dialogs are made up of widgets, and explains how scripts control dialogs. A more complete example, including a screen shot of a TSD dialog box, is in the last section of this document.

The following is a "widget tree dump" of a portion of a dialog box. This kind of dump is produced by AutoTest, and it's very useful for developing scripts and identifying programming errors.

```
0xd90ff68 XmRowColumn 'workarea' (0+16, 0+16)/0
  0xd9103a8 XmForm 'WorkAreaForm' (0+0, 0+0)/0
  | 0xd910568 XmXrtAligner 'workarea' (0+4, 0+9)/0
  |   0xd9108e8 XmLabel '30 min' (0+55, 0+22)/0
  |   0xd910a78 XmForm 'eAccScore30BotForm' (0+0, 0+0)/0
  |   | 0xd910c70 XmLabel '30%' (0+36, 0+22)/0
  |   0xd910e28 XmLabel '60 min' (0+55, 0+22)/0
  |   0xd910fe8 XmForm 'eAccScore60BotForm' (0+0, 0+0)/0
  |   | 0xd911190 XmLabel '60%' (0+36, 0+22)/0
  |   0xd911350 XmLabel '120 min' (0+64, 0+22)/0
  |   0xd9114f8 XmForm 'eAccScore120BotForm' (0+0, 0+0)/0 Unmanaged
  |     0xd9116b8 XmLabel '120%' (0+45, 0+22)/0
  0xd911870 XmRowColumn 'eAccShowForm' (16+16, 16+16)/0
    0xd911a30 XmXrtToggleButton 'Show Scoring Region' (0+204, 0+26)/0
```

It is not important to understand all the data in this dump. It is important to understand the relationship between the widgets.

Motif has two kinds of widgets: widgets that display data or allow user interaction (labels, buttons, combo boxes, etc.), and manager widgets (also called container widgets), which usually appear in the dialog as a box or region containing other widgets.

The dump shown above, has a row-column (**`XmRowColumn`**) container widget named "**`workarea`**." Workarea contains two other widgets: A Form widget named "**`WorkAreaForm`**", and a row-column widget named "**`eAccShowForm`**."

The Form widget contains only one widget, an **`XmXrtAligner`** widget with the name "**`workarea`**" (same name as the top row-column widget).

The Aligner widget contains several other widgets -- Labels, and Forms containing Labels. One of the forms is Unmanaged, meaning that it will not be displayed in the dialog box.

To control a dialog from a script, follow three steps in writing the script:

1. Use the **`TsdDlg`** command to identify the dialog box, and make it the "current" dialog box.
2. Use the **`Find`** and **`FindNext`** commands to identify the widget that you wish to manipulate, and make it the "current" widget. If you are searching with an **`XrtTable`**, use the **`XrtTab`** command to find the desired widget within the table, and make it the "current" widget.
3. Use the **`Widget`** command, with subcommands (**`pushbutton`**, **`toggle`**, **`gettext`**, **`text`**, etc.) to manipulate the current widget.

For example, the script commands for filling in some fields in the Reroute Dialog are as follows:

```
# Set the start/end times to 1230 and 2359, respectively, and
# set the reroute name to BNA-AAAA
TsdDlg RerouteDialog
Find label=.Start.Time: XmXrtDateField XmXrtDateField
Widget text 1230
Find label=.End.Time: XmXrtDateField XmXrtDateField
Widget text 2359
FindNext label=.Name: XmXrtStringField XmXrtStringField
Widget text BNA-AAAA
FindNext label=Domain: button=Shared
Widget toggle
```

In the script shown above, the first **Find** command contains three subcommands.

- The first subcommand finds a label matching ".**Start.Time**:" (the "." matches any character).
- The second finds the next widget with type **XmXrtDateField**.
- The third finds the next widget after that with type **XmXrtDateField**.

When this script is executed, this sequence skips over the date portion of the state time, and goes to the time portion. Then, "**Widget text 1230**" fills in a time value of "**1230**".

The last **FindNext** command first finds the label with name "**Domain:**", then finds the button label "**Shared**". The final command in the sequence clicks on the button.

The following is an example where the scripted "user" selects the Advisory tab, fills in the fields, and clicks the "**Send**" button:

```
TsdDlg RerouteDialog
Find tabmanager
Widget changepage 5   # move to Advisory page
Find label=CONSTRAINED.AREA XmXrtStringField
Widget text CONSTRAINEDAREA
# Click both of the Auto Fill buttons.
FindNext button=Auto.Fill
Widget pushbutton
FindNext button=Auto.Fill
Widget pushbutton
# Set the combo box at the bottom to "Self"
FindNext label=Message.Type: combo
Widget text Self
# Push the Send button
Find XmPushButtonGadget      # Send...
Widget pushbutton
```

3 Widget Manipulation Commands

The AutoTest widget manipulation commands are listed below.

3.1 TsdDlg dialog-spec [sub-command]

This command sets the "current" dialog to the one specified. There are two options:

- The dialog-spec is the name of a standard TSD dialog, such as RerouteDialog, ReroutePlaybookDialog, or WeatherSelectDialog.
- The dialog-spec is in the format “title=string”, where “string” is a pattern representing the title at the top of an arbitrary dialog. AutoTest finds the dialog by searching through all XLib windows for one with the specified title. For this option, only the “cancel” and “dump” sub-commands are available.

The optional sub-command is one of the following:

- **`post`** - Post the dialog.
- **`ok`** - Send "ok" to dialog. This will work even if the OK button is not visible.
- **`cancel`** - Send "cancel" to dialog.
- **`dcancel`** - Send "cancel" to dialog through base DialogBox class.
- **`dump`** - Provide a "widget tree dump" of the dialog box.

3.2 Find [sub-command ...] or FindNext [sub-command ...]

These two commands search for the desired widget within the dialog box widget tree, and set it as the "current" widget. They search in the order previously described (prefix tree walk order). The difference between the two commands is that Find starts at the top of the dialog box widget tree, while FindNext continues a search from a previous Find or FindNext.

There can be a list of sub-commands, each of which advances to a new widget. Each additional sub-command continues a search for the widget found by the preceding sub-command.

There are three one-word sub-commands:

- **`ok`** -- the target widget is the dialog OK button
- **`cancel`** -- the target widget is the dialog CANCEL button
- **`help`** -- the target widget is the dialog HELP button

Otherwise, the sub-command has the following format:

```
[mod:][class][=label]
```

where

- **`mod`** - not used here; used only with the XrtTab command (see below)

- **`class`** - widget type or class. It can be the exact name of a Motif widget class (**`XmForm`**, **`XmMessageBox`**, **`XmCascadeButtonGadget`**, etc.), or it can be one of the following superclasses:
  - button -- matches any type of button
  - combo -- matches any type of combo box
  - label -- matches any type of label
  - form, frame, messagebox, outliner, rowcolumn, scrollbar, tabmanager, text, textfield, xrttable -- matches any type of widget of that class
- label -- text associated with the widget. A "." matches any character. Warning: In the case of a button, the text is the text associated with the button, as provided when the button was first created. However, if the button text has changed since creation, then the original text does not change.

3.3 XrtTab [sub-command ...]

When the "current" widget is an **`XrtTable`**, then the **`XrtTab`** command is used to find a desired widget within the table, or to manipulate the table in other ways. The sub-commands have the following format:

- column=N -- make N the "current" column
- row=N -- make N the "current" column
- column:label=TEXT -- find the column with label specified by TEXT, and make it the "current" column
- row:label=TEXT -- find the row with label specified by TEXT, and make it the "current" row
- widget[:class] -- select the widget in the "current" row and "current" column, and make it the "current" widget. If a class is specified, then verify that the widget is of this class.
- visiblerow -- scroll the table, if necessary, to make the "current" row visible
- selectrow -- select all the cells in the "current" row

3.4 Widget sub-command [argument]

This command executes the sub-command on the "current" widget. The sub-commands are:

- **`pushbutton`** - push the button
- **`toggle`** -- toggle the toggle button
- **`text`** -- set the text in the widget to the specified argument
- **`gettext`** -- get the text from the widget, and assign it to the variable specified by the argument
- **`select`** -- select the item with the label specified by the argument from the widget, which must be an XrtOutliner widget.
- **`changepage`** -- change the current tab into the tab # specified by the argument; the "current" widget must be an XrtTabManager widget.
- **`monitor`** -- use XtAddActionHook to monitor events from the current widget. This can produce a great deal of output, as all events (KeyPress/Release, MotionNotify, ButtonPress/Release, Enter/LeaveNotify, etc.) will be captured, and a line of information will be printed for each one.

- **`monitor*`** -- monitor events from ALL widgets in the dialog box, for intensive debugging.
- **`unmonitor`** -- turn monitoring off.

4 Variables and Expressions

The assignment statement is as follows:

```
Let variable = expression
```

The statement declares a variable and assigns a value to it. If the **`Let`** statement appears in a subroutine, the declaration is local to that subroutine. There is dynamic scoping of variables, meaning that the variable can be accessed from any called subroutine.

The expression interpreter supports the following operators:

- Arithmetic operators: + - * /
- Logical operators: && || NOT
- Numeric comparison operators: == != < <= > >=
- String operator (match string to pattern): MATCH
- Parentheses: ( )

Operator precedence is the same as in C++.

Note that string concatenation is not needed, because it can be accomplished by variable substitution.

If a script command contains a string like $(variable), the value of the variable is substituted.

5 Flow of control statements

The following statements define flow of control within a script

- **`Label name`**

  This statement defines a label which can be targeted by a Goto.

- **`Goto name`**

  Goto the statement with the specified name. If there is no label with that name in the current subroutine, but there is in the stack of calling subroutines, then a goto is executed to that label, forcing a return from all intermediate subroutines.

- **`Sub name [parms...]`**

  Defines the beginning of a subroutine, with the specified name, and an optional list of parameter names. Each parameter name becomes a variable declaration within the newly defined subroutine.

- **`Endsub`**

  Defines the end of a subroutine.

- **`Call name [args...]`**

  Calls the subroutine with the specified name. Each of the argument values is assigned to the corresponding parameter in the Sub statement.

6 "IF" command modifier clauses

Any AutoTest command can be modified by adding a modifier clause to the end of the statement. The format is as follows:

```
command If expr
```

The command is executed only if the expression is true.

7 Memory stressing and verbosity options

The command **`Set option on|off [option on|off]`**...sets one or more AutoTest options on or off.

The following are the options:

- **`stressmemory`** -- If on, then AutoTest will perform a memory stress test after every command is executed. The stress test alternates, allocating memory after one command, and then freeing the memory after the next command. When allocating memory, 1,000 memory blocks of random sizes from 20 to 200 are allocated, and each one is initialized to a string. When freeing memory, the 1,000 memory blocks are freed after checking that the string hasn't been clobbered. This option substantially slows down testing.
- **`stressstl`** -- If on, then AutoTest performs an STL memory stress test after every command is executed. It's similar to 'stressmemory', but the allocated and freed objects are TSD objects that heavily use STL -- strings, vectors, maps, etc.
- **`verbose`** -- This is the general "verbose" option, that causes diagnostic information to be printed after a variety of operations. The more specific verbose options are listed in the following items.
- **`verbosecommands`** -- Print each command as it's being executed.
- **`verbosememory`** -- For each command, print a summary of the total virtual memory and total hardware memory allocated by TSD. This is useful for find memory leaks.
- **`verbosefind`** -- Traces the search for a widget via the subcommands of the Find and FindNext commands. This is very useful for debugging scripts that contain these commands.
- **`verbosexrt`** -- Traces the operation of the XrtTab command. This is very useful for debugging scripts that manipulate XrtTables.
- **`verbosevble`** -- Traces the search for variables.
- **`verboseexpr`** -- Traces the evaluation of expressions.

8 Miscellaneous commands

- **`Quit`** - -Ends testing.
- **`Crash`** - Crashes TSD.
- **`Sleep #periods`** - Testing is suspended for the specified number of 2-second periods.
- **`Echo args`** - Prints the arguments. This is useful for verifying that argument substitutions are doing the right thing.
- **`Dump`** - Dumps TSD's main window.

9 Example of Dialog Box

The following is a TSD dialog box, along with the "widget tree dump" produced by AutoTest:

```
********** Heading: TsdDlg dump of dialog RerouteSendDialog **********

0xcd47278 XmMessageBox 'Send Reroute' (0+379, 0+355)/0
  0xcbcdee06 XmLabelGadget '[noname]' (0+4, 0+4)/0
  0xb065bc88 XmLabelGadget '[noname]' (1+377, 1+4)/0
  0xbf271988 XmSeparatorGadget '[noname]' (0+379, 306+2)/0
  0x8e63c582 XmPushButtonGadget '[noname]' (1+82, 308+46)/0
  0x8efddb80 XmPushButtonGadget '[noname]' (296+82, 308+46)/0
  0x8f604480 XmPushButtonGadget '[noname]' (0+60, 0+42)/0 Unmanaged
  0x8e55d88 XmRowColumn 'manager' (1+377, 5+301)/0
    0x8f900b8 XmFrame 'frame' (3+371, 3+295)/0
    | 0x90f23a0 XmForm 'form_work_area' (2+367, 2+291)/0
    |   0xbce93e0 XmXrtGLabel '' (0+10, 0+10)/0
    |   0x8efdf38 XmXrtGLabel 'Send Reroutes to TSDs' (40+223, 10+32)/0
    |   0x8f03a48 XmFrame 'frame' (0+346, 40+96)/0
    |   | 0x8f5fcd8 XmXrtGLabel 'TSD Show Options:' (12+168, 0+32)/0
    |   | 0xb215d98 XmForm 'form_work_area' (2+342, 32+62)/0
    |   |   0x90cf118 XmXrtGLabel 'ShowRouteIcon' (0+84, 0+32)/0
    |   |   0xb041d18 XmXrtGLabel 'ShowLabelIcon' (104+79, 0+32)/0
    |   |   0x8dee638 XmToggleButton 'Navaids' (0+92, 32+30)/0
    |   |   0x8e69bc8 XmToggleButton 'Fixes' (100+72, 32+30)/0
    |   |   0x90da960 XmXrtGLabel 'Color:' (251+61, 30+32)/0
    |   |   0xd0ca338 XmDrawnButton '      ' (312+30, 32+30)/0
    |   0x8e8fcd8 XmForm 'advisoryForm' (0+367, 156+135)/0
    |     0xbc870b8 XmToggleButton 'sendAdvisory' (0+145, 0+30)/0 Insensitive
    |     0x90d6c00 XmToggleButton 'Attach Flight List' (0+165, 35+30)/0 Insensitive
    |     0x90ff088 XmPushButton 'Print Flight List Now' (195+172, 30+30)/0 Insensitive
    |     0x90fa8c8 XmToggleButton 'Print Flight List After Sending' (0+265, 70+30)/0 Insensitive
    |     0x902c238 XmToggleButton 'Include Protected Segments' (0+255, 105+30)/0
    |     0x8fa4718 XmXrtGLabel 'IncludeProtectedSegmentsIcon' (0+261, 0+32)/0 Unmanaged
    0x902c6e8 XmFrame 'frame' (0+391, 0+162)/0 Unmanaged
      0x8e86c08 XmXrtGLabel 'Cancellation Messages:' (12+200, 0+32)/0
      0xbcee2d8 XmForm 'form_work_area' (2+387, 32+129)/0
        0x8e17768 XmLabel 'Some flights might be dropped from the reroute.' (0+387, 0+22)/0
        0x8ea2f98 XmLabel 'Send cancellation messages for:' (0+261, 38+22)/0
        0x8fa1778 XmRowColumn 'radiobox' (60+309, 60+69)/0
          0x90d7a98 XmToggleButton 'All Dropped Flights' (3+303, 3+30)/0
          0x8e602d8 XmToggleButton 'Only Dropped Not-Airborne Flights' (3+303, 36+30)/0

-----------------
********** 20110425182247
```