Win Runner- Basic Concepts

Automation Testing Tool

BFS4

HEMAMALINI M
Hemamalini.manivannan@tcs.com

Contents

Introducing WinRunner

If you have ever tested software manually, you are aware of its drawbacks.Manual testing is time-consuming and tedious, requiring a heavy investment in human resources. Worst of all, time constraints often make it impossible to manually test every feature thoroughly before the software is released. This leaves you wondering whether serious bugs have gone undetected.

Automated testing with WinRunner addresses these problems by dramatically speeding up the testing process. You can create test scripts that check all aspects of your application, and then run these tests on each new build. As WinRunner runs tests, it simulates a human user by moving the mouse cursor over the application, clicking Graphical User Interface (GUI) objects, and entering keyboard input—but WinRunner does this faster than any human user.

Features

WinRunner is:

- Functional Regression Testing Tool
- Windows Platform Dependent
- Only for Graphical User Interface (GUI) based Application
- Based on Object Oriented Technology (OOT) concept
- Only for Static content
- Record/Playback Tool

Add Ins

WinRunner includes the following Addins:

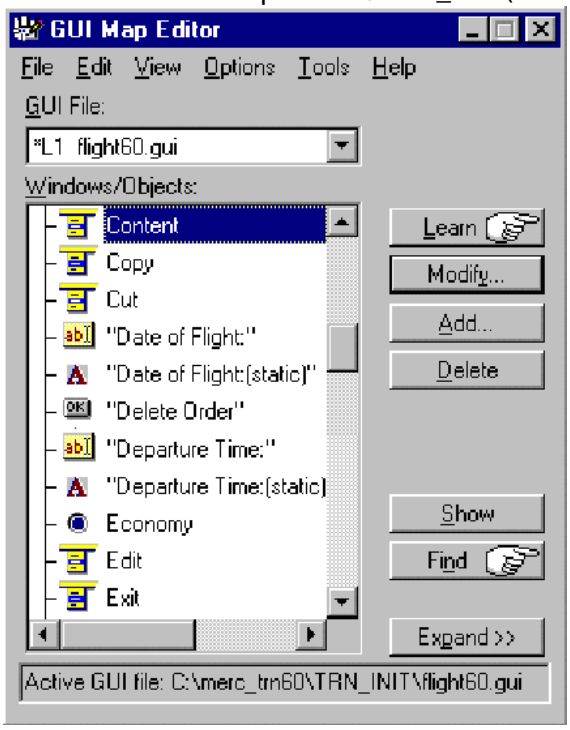
- Web Test
- Visual Basic
- ActiveX
- Power Builder

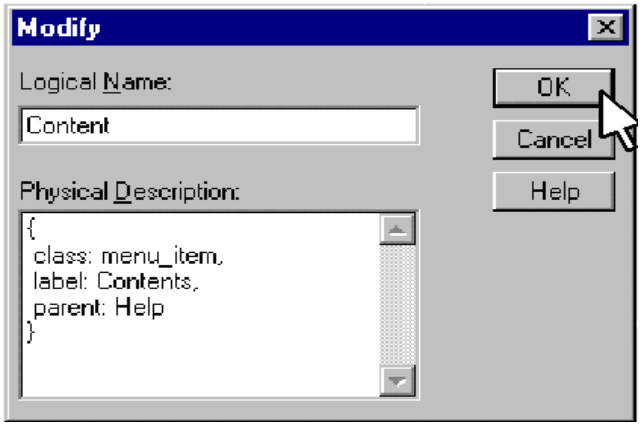
How does Win Runner identify GUI Objects

- GUI applications are made up of GUI objects such as windows, buttons, lists and menus.
- WinRunner's Rapid Test Script Wizard learns the descriptions of all GUI objects
- It saves the object description in GUI Map file.gui, which is the Heart of Win Runner
- When we run tests, WinRunner uses this file to identify and locate objects

Creating GUI Map file and Loading it

- There are three ways of creating GUI Map file
  - **Rapid Test Script Wizard-** systematically opens the windows in your application and learns a description of every GUI object.
    - Used to learn the entire application
    - User Interface Test
  - **Recording**- adds windows and objects to the GUI Map as they are encountered by the user
  - **GUI Map Editor**- used to store all the information about GUI elements present in your application. The GUI Map editor tool can be used edit the information in the map file easily.
- You can load the GUI Map file through
  - GUI Map editor / GUI_load("filename.gui")





Recording Test

- By recording, we can quickly create automated test scripts, clicking objects with mouse, entering Keyboard input
- Recording generates statements in TSL, Mercury's interactive Test Script Language, Case sensitive.

Choosing Record Mode

- Before you begin recording a test, you should select the appropriate record mode.
- There are two record modes available
    - Context Sensitive Mode- records operation you perform in terms of GUI objects

e.g. button_press("Ok")

    - Analog Mode – records exact co-ordinates traveled by mouse and keyboard inputs

e.g. Mtype("<kleft>+")

Running the Test

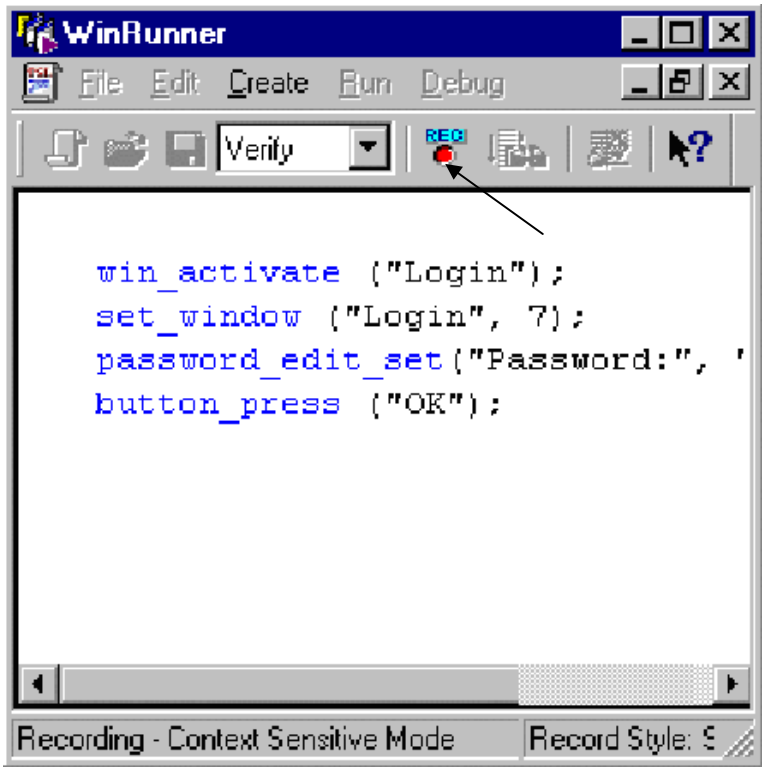
- WinRunner provides three modes for running test
    - Use **Verify mode** when running a test to check the behavior of our application and when we want to save the test result.
    - Use **Debug Mode**, when you want to check that the test script runs smoothly without errors in syntax. The debug mode will not give the test result.
    - Use **Update mode**, when you want to create new expected results for a GUI check point or bitmap check point

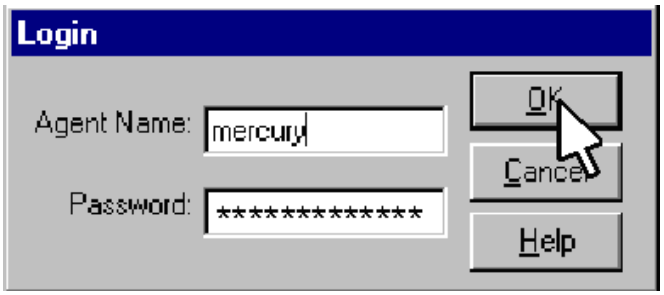
Win Runner Testing Process

- **Create GUI Map File:** By creating GUI Map file the WinRunner can identify the GUI objects in the application going to be tested.
- **Create Test Scripts**: This process involves recording, programming or both. During the process of recording tests, insert checkpoints where the response of the application needs to be tested.
- **Debug Test:** Run the tests in Debug mode to make sure whether they run smoothly.

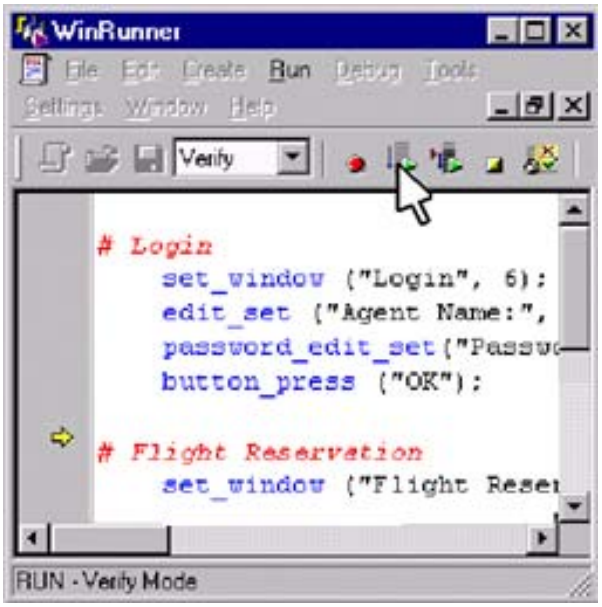
- **Run Tests:** Run tests in Verify mode to test the application.
- **View Results:** This determines the success or failure of the tests.
- **Report Defects**: If a particular test run fails due to the defect in the application being tested, defects can be directly reported through the Test Results window.

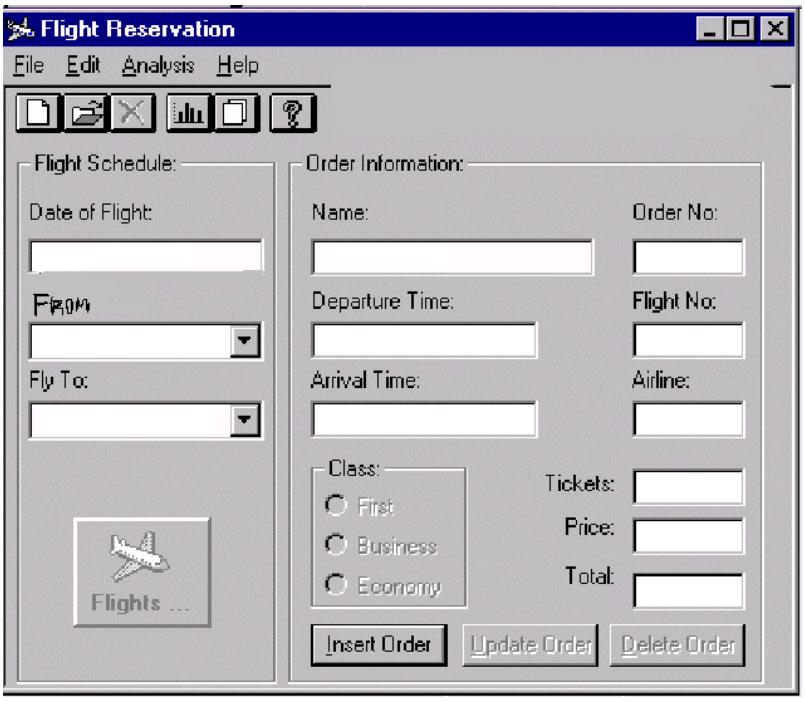
Record Test



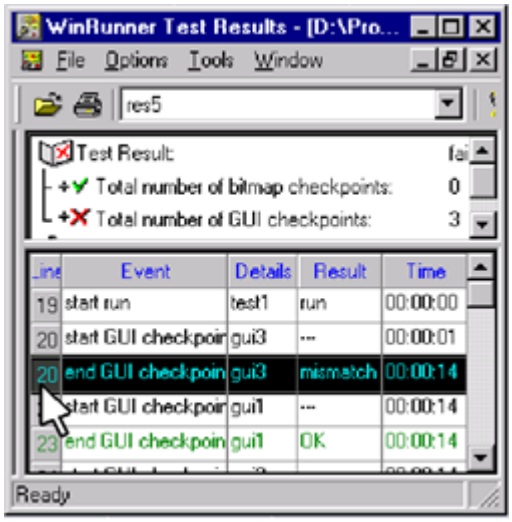


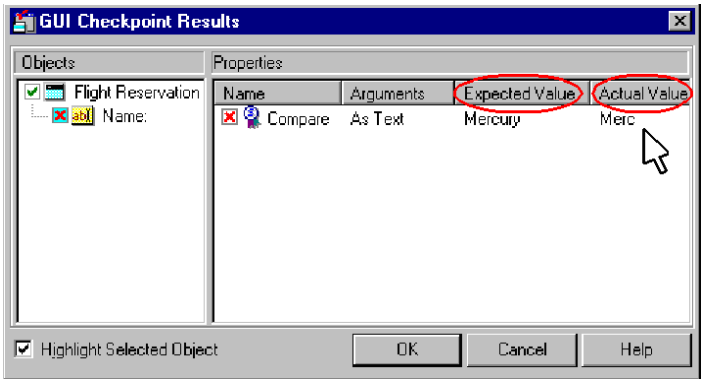
Run Test


WinRunner
File Edit Create Run Debug Tools
Settings Window Help
Verify
Login
set_window ("Login", 6);
edit_set ("Agent Name:",
password_edit_set("Passw
button_press ("OK");
Flight Reservation
set_window ("Flight Reser
RUN - Verify Mode


Flight Reservation
File Edit Analysis Help
Flight Schedule:
Date of Flight:
From
Fly To:
Flights ...
Order Information:
Name:
Order No:
Departure Time:
Flight No:
Arrival Time:
Airline:
Class:
First
Business
Economy
Tickets:
Price:
Total:
Insert Order
Update Order
Delete Order

Sample Test Result

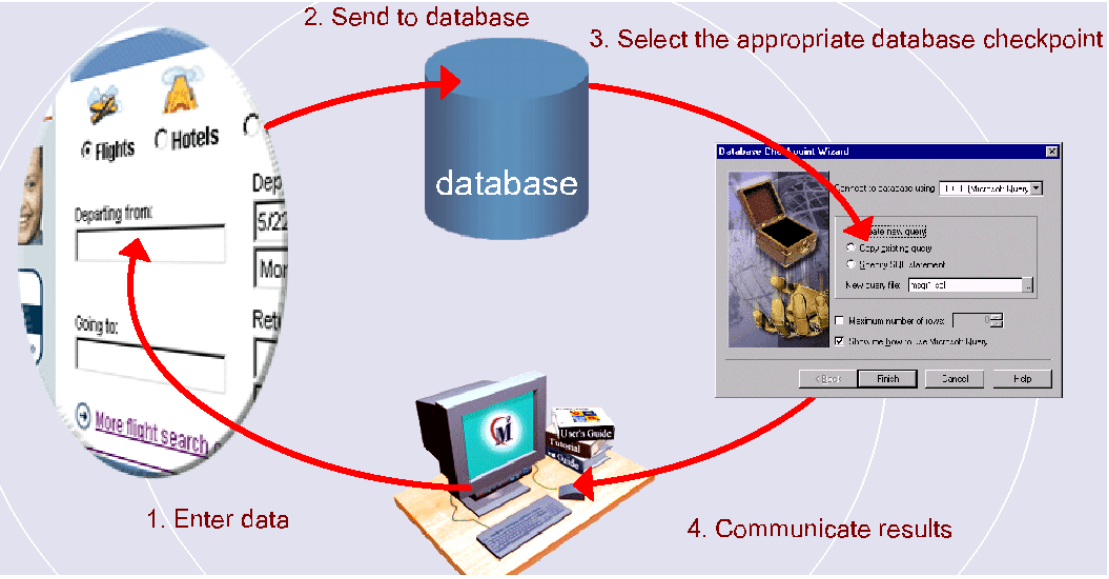




GUI Checkpoint

- Checkpoints allow you to compare the current behavior of the application being tested to its behavior in an earlier version. You can add four types of checkpoints to your test scripts
- **Object Checkpoint** verifies information about GUI objects. For example, you can check
  - Whether radio button is on or off
  - Whether a push button is enabled or disabled
- **Text Checkpoint** read text in GUI objects and in bitmaps and enables you to verify their contents. For example, you
  - Can read the text content of any button
- **Bitmap Checkpoint** takes a "snapshot" of a window or area of your application and compares this to an image captured in an earlier version.
  - Capture drawings and graphs
- **Database Checkpoint** check the contents and the number of rows and columns of a result set, which is based on a query you create on your database.
  - So you create a query and examine the result set

Database Checkpoint- Compare Expected and Actual Outcomes



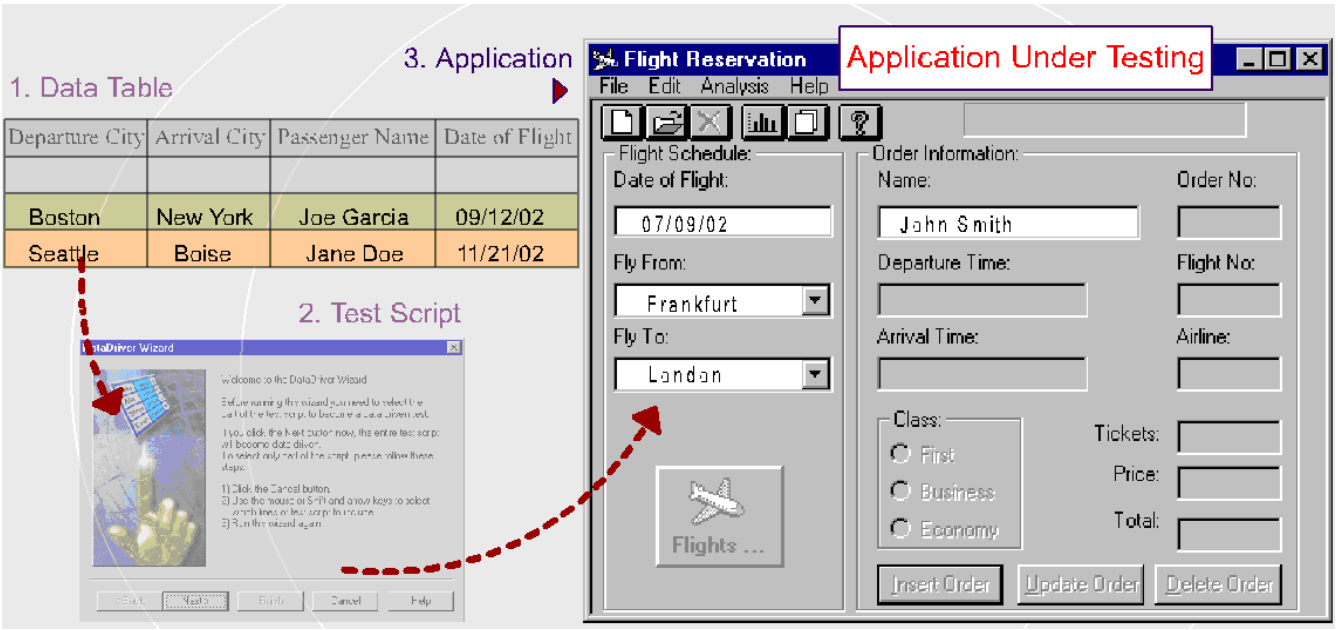
Data Driven Test

- When you want to test your application, you may want to check how it performs the same operation with multiple sets of data. So we use Data Driven Test. By replacing the fixed values in your test with values stored in a data table (an external file), you can generate multiple test scenarios using the same test.
- Two ways of Data Driven Test
  - Data driven Wizard
  - Modify Test Script manually

Data Driven Testing Process

- Creating a test
- Converting it in to a Data Driven Test
- Preparing a Data table
- Running the Test
- Analyzing the results

Application Performance with Data Driver Wizard



Synchronization

- Synchronization is used to have the uniformity between the application and test scripts. It enables you to solve anticipated timing problems between the test and your application
- For example, if you create a test that opens a database application, you can add a synchronization point that causes the test to wait until the database records are loaded on the screen.
- So you could synchronize:
    - To retrieve information from a database
    - For a window to popup
    - For a progress bar to reach 100%
    - For a status bar message to appear

Batch Test

- A Batch test is a test script that contains call statements to other tests
- It opens and executes each test and saves the test results.
- It suppresses the error message that occur while running the test script
- A test becomes a batch test when you select the run in batch mode option
    - e.g. GUI_load("a1.gui");

      call "a1"()

Dialog Boxes

- You can create dialog boxes, that popup during interactive test execution
- It will prompt the user to perform an action such as typing in text or selecting an item from the list
- Types of Dialog boxes
    - Input Dialog boxes
    - List Dialog boxes
    - Password Dialog boxes

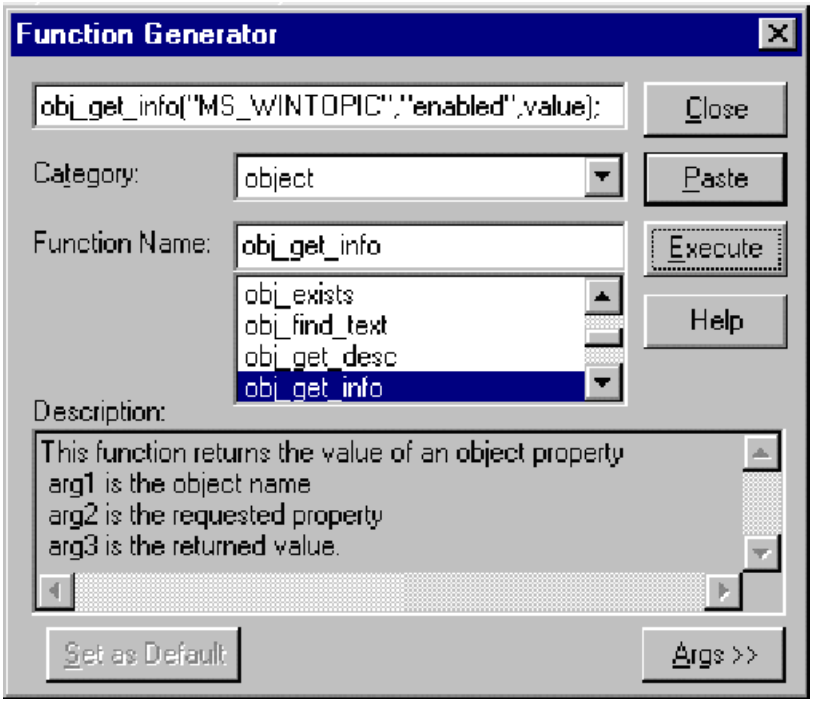
Functions

- Inbuilt functions
    - Insert Function- Object/ window

- Function Generator- a visual tool that presents a quick and error-free way to program your tests.

You can add TSL statements to your tests using the Function Generator in two ways:

- By pointing to a GUI object, or
- By choosing a function from a list.



- User defined functions are
  - Compiled modules –
    - A complied module is a script containing a library of user defined function
    - When you load a compiled modules in a script, its function are automatically compiled and remain in memory.
    - Compiled modules can improve the performance of your tests
    - Since you debug the compiled module before using them, your test will require less error checking

Regular Expressions

- Regular expression enables Win Runner to identify objects with varying names and titles.
- You can use regular expressions in TSL statements or in object descriptions in the GUI map
    - 3[0-9]
    - 3*
    - *.*

Exception Handling

- Using exception handling, you can instruct Win Runner to deduct an unexpected event when it occurs, and act to recover the test run
- Types of Exceptions
    - Pop up exception
    - TSL exception
    - Object exception

Break points

- By setting a break point you can stop a test run at a specific place in the test script
- You can set break points
    - Break at location
    - Break in function

Appendix

GUI- Graphical User Interface

TSL- Test script Language