



## A Software Testing Framework

### Introduction

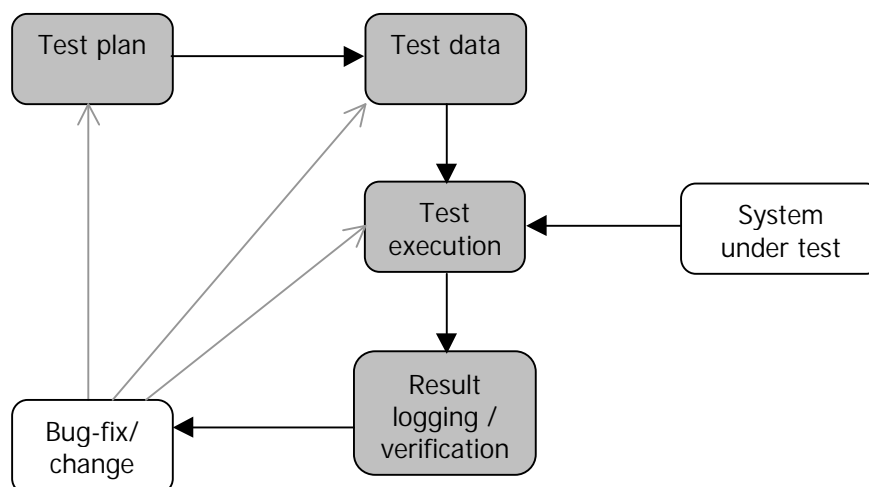
Software testing has always been a tedious and expensive affair. It takes over 50% of software development time and cost. And at the end of this expensive process, it is still difficult to say with confidence that the software is error free – leading to heartburn both for the software developer and the user.

There are four critical steps (detailed below) to any successful software testing exercise. Yantra is a suite of tools aimed at *automating* these steps of checking if the software meets its required functional specifications.

- 1. Test planning:** This step involves determining the right test cases and scenarios – the most critical step in software testing. Decisions regarding the exact test strategy, the process of testing, the desired coverage, the scenarios of testing, etc. are taken during this phase.
- 2. Test data selection:** The next step encompasses deciding the actual test input values – based on the test plan that has been prepared – to test each scenario or case. This test data should be chosen so that the desired levels of specification and program coverage are achieved. This has a direct bearing on the effectiveness of the actual testing.
- 3. Test execution:** In this step, the tests are actually executed – using the selected test data – to exercise the program on the chosen scenarios. Typically, the test execution itself may *log* the associated test results in some desired form. This is the easiest of the steps involved in testing.
- 4. Test verification:** The *results* of executing each test have to be verified in the final step of testing. This involves determining which test cases went through successfully and which failed to do so. Thus, this activity is the one that exposes the faults in the system, and is followed by defect analysis and problem fixing. The process of test verification is often very tedious and highly error-prone.

These four steps are required at all *levels* of testing – unit testing, module testing, integration testing and system testing. Of course, the level of detail involved at each step may vary depending on the level of testing.

The process of software testing is graphically depicted below. Yantra automates the steps shown by the shaded boxes.



**Software Testing using Yantra**

---

## Features

The steps in the process of testing a software system, aided by Yantra, are as follows:

- 1. Test planning:** Yantra supports formal descriptions of test scenarios and test inputs – called *test specifications*. These test specifications enable Yantra to automate the various testing activities. Moreover, if the system under test has been built from a formal *model*, such as a UML or MasterCraft model, Yantra provides tools to produce a *template* test specifications from such models. This simplifies the task of producing test specifications as they are *automatically derived* from the same specifications used for developing the code.

The test specifications support description of testing at all *levels* such as unit testing, integration testing and system testing, and of all *kinds* such as functional testing and stress testing.

- 2. Test data selection:** Given the *formal test specifications*, Yantra provides tools to generate test data that satisfy these specifications. Moreover, it also *guarantees* that the generated test data is *meaningful* (according to the logic or rules of the specific system-under-test) and that the generated test data *covers* different conditions and combinations of conditions.

Testing of Object Oriented systems has specific issues - like testing of inheritance, overloading and virtual functions, without breaking encapsulation. Yantra provides special features to address these issues.

- 3. Test execution:** Automation of this step is dependent on the particular platform in which the application has been developed. Yantra has been designed and implemented to support quick generation of customized test drivers to automate test execution for different platforms and languages. Yantra can also interface with other test execution automation tools for this phase. Currently, Yantra supports test drivers to load data into Oracle databases, to unit-test MasterCraft-based systems, and to test systems developed in C++.

- 4. Test verification:** Verification is a semi-automatic process in Yantra. Two methods of verification are currently supported:

- a) Logging test results:** The test designer can choose which data to log at what points of the testing process and Yantra generated test-drivers will log that data into a test log file. This log file can then be used for manual verification.
- b) Invoking user defined 'oracle functions':** The test designer can achieve a higher level of test verification automation by providing functions to verify the systems' consistency at various points in the testing process and invoking them appropriately from the test-specification. The success or failure as indicated by the user-provided 'oracle functions' are then logged by Yantra generated test-drivers.

Thus, with Yantra, the focus is firmly on one step: that of creating the input specifications. If this step is done well, the rest of the process can be largely automated.

## Benefits

- Guaranteed adherence to the test plan
- Improved coverage of the features tested
- Large savings in time and cost of testing
- Increased confidence that the software meets its requirements
- Improved test management, because logs and statistics of testing can be automatically generated

## Platforms Supported

- Windows
- Sun Solaris
- Yantra can also be made available on other platforms.

## For further information contact

[yantra@pune.tcs.co.in](mailto:yantra@pune.tcs.co.in) Or [lptools@pune.tcs.co.in](mailto:lptools@pune.tcs.co.in)