An Improved Approach of Software Testing Tool

Mahan Sunhare

*Corresponding author:
Mahan Sunhare
Dept of IT
MIT Ujjain, MP India

Abstract
Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs. These operations are performing to provide a Quality software product. In this project we are going to perform the automated testing and provide the new way of testing by introducing the test cases using visual parameter passing.

Keywords: testing; automated testing; parameter.

Introduction
Software testing can be stated as the process of validating and verifying that a software program/application/product: [1] meets the business and technical requirements that guided its design and development; [2] works as expected; and [3] Can be implemented with the same characteristics.
Software Testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Software is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails.
Software testing is more than just error detection. Testing software is operating the software under controlled conditions, to [1] verify that it behaves "as specified"; [2] to detect errors, and [3] to validate that what has been specified is what the user actually wanted.
Verification is the checking or testing of items, including software, for conformance and consistency by evaluating the results against pre-specified requirements. [Verification: Are we building the system right?]
Error Detection: Testing should intentionally attempt to make things wrong to determine if things happen when they shouldn’t or things don’t happen when they should.
Validation looks at the system correctness – i.e. is the process of checking that what has been specified is what the user actually wanted. [Validation: Are we building the right system?]
In other words, validation checks to see if we are building what the customer wants/needs, and verification checks to see if we are building that system correctly. Both verification and validation are necessary, but different components of any testing activity.
The definition of testing is that testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.
Software testing is both a discipline and a process. Though software testing is part of the software development process, it should not be considered part of software development. It is a separate discipline from software development. Software development is the process of coding functionality to meet defined end-user needs. Software testing is an iterative process of both validating functionality, and, even more important, attempting to break the software. The iterative process of software testing consists of:
Designing tests
Executing tests
Identifying problems
Getting problems fixed
The objective of software testing is to find problems and fix them to improve quality. Software testing typically represents 40% of a software development budget.

Types of Software Testing
Testing can be performed on Automated and Manual basis using:
[1] Unit Testing: This initial stage in testing normally carried out by the developer who wrote the code and sometimes by a peer using the white box testing technique. [2] Integration Testing: This stage is carried out in two modes as a complete package or as an increment to the earlier package. Most of the time black box testing technique is used. However, sometimes a combination of Black and White box testing is also used in this stage. [3] System Testing: In this stage the software is tested from all possible dimensions for all intended purposes and platforms. In this stage Black box testing technique is normally used. [4] User Acceptance Testing: This testing stage carried out in order to get customer
sign-off of finished product. A ‘pass’ in this stage also ensures that the customer has accepted the software and is ready for their use.

[5] White box testing: White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these. [6] Black box testing: Black box testing treats the software as a “black box”—without any knowledge of internal implementation. [7] Grey box testing: Grey box testing (American spelling: gray box testing) involves having knowledge of internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. [8] Regression Testing: If a piece of Software is modified for any reason testing needs to be done to ensure that it works as specified and that it has not negatively impacted any functionality that it offered previously. This is known as Regression Testing. [9] Load testing: Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the systems response time degrades or fails. [10] Stress testing: System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, and continuous input to system or database load. [11] Performance testing: Term often used interchangeably with ‘stress’ and ‘load’ testing. To check whether system meets performance requirements. Used different performance and load tools to do this. [12] Usability testing: User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing. [13] Install/uninstall testing: Tested for full, partial, or upgrade install/uninstall processes on different operating systems under different hardware, software environment. [14] Recovery testing: Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems. [15] Security testing: Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

Test automation may be able to reduce or eliminate the cost of actual testing. A computer can follow a rote sequence of steps more quickly than a person, and it can run the tests overnight to present the results in the morning. However, the labor that is saved in actual testing must be spent instead authoring the test program. Depending on the type of application to be tested, and the automation tools that are chosen, this may require more labor than a manual approach. In addition, some testing tools present a very large amount of data, potentially creating a time consuming task of interpreting the results. From a cost-benefit perspective, test automation becomes more cost effective when the same tests can be reused many times over, such as for regression testing and test-driven development, and when the results can be interpreted quickly. If future reuse of the test software is unlikely, then a manual approach is preferred.

Things such as device drivers and software libraries must be tested using test programs. In addition, testing of large numbers of users (performance testing and load testing) is typically simulated in software rather than performed in practice. Conversely, graphical user interfaces whose layout changes frequently are very difficult to test automatically. There are test frameworks that can be used for regression testing of user interfaces. They rely on recording of sequences of keystrokes and mouse gestures, then playing them back and observing that the user interface responds in the same way every time. Unfortunately, these recordings may not work properly when a button is moved or relabeled in a subsequent release. An automatic regression test may also be fooled if the program output varies significantly (e.g. the display includes the current system time). In cases such as these, manual testing may be more effective.

Improved Testing Tool

Testing is the primary tool for software quality assurance. It embodies not just the act of running a test, but designing tests, predicting test outcomes, establishing standards for tests, and corrective procedures for discovered errors. Developers are under great pressure to deliver more complex software on increasingly aggressive schedules and with limited resources. Testers are expected to verify the quality of such software in less time and with even fewer resources. In such an environment, dynamic testing tools are a must to improve the software testing process.

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit that originated with NUnit. JUnit is linked as a JAR at compile-time; the framework resides under packages JUnit framework for JUnit 3.8 and earlier and under org.junit for JUnit 4 and later.

Find Bugs is a static analysis tool that examines your class or JAR files looking for potential problems by matching your bytecodes against a list of bug patterns. With static analysis tools, you can analyze software without actually running the program. Instead the form or structures of the class files are analyzed to determine the program’s intent, often using the Visitor pattern.

JUnit consumes much time and due to command line interface it is more hectic work. Over this writing test cases is also very time consuming. More over it is valid for only the java programs, which are supported only for desktop applications. It is fails when we are write simple JSP programs.

Design of Experiments & Logistic Regression

Design of experiments (DOE) or experimental design is the design of any information-gathering exercises where variation is present, whether under the full control of the experimenter or not. However, in statistics, these terms are usually used for controlled experiments.

Experimental design (commonly referred to as DOE) is a useful complement to multivariate data analysis because it generates “structured” data tables, i.e. data tables that contain an important
amount of structured variation. This underlying structure will then be used as a basis for multivariate modeling, which will guarantee stable and robust models.

More generally, careful sample selection increases the chances of extracting useful information from the data. When one has the possibility to actively perturb the system (experiment with the variables), these chances become even greater. The critical part is to decide which variables to change, the intervals for this variation, and the pattern of the experimental points.

Logistic regression (sometimes called the logistic model or logit model) is used for prediction of the probability of occurrence of an event by fitting data to a logit function logistic curve. It is a generalized linear model used for binomial regression. Like many forms of regression analysis, it makes use of several predictor variables that may be either numerical or categorical.

An explanation of logistic regression begins with an explanation of the logistic function:

\[ f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}} \]

A graph of the function is shown in figure 1. The input is \( z \) and the output is \( f(z) \). The logistic function is useful because it can take as an input any value from negative infinity to positive infinity, whereas the output is confined to values between 0 and 1. The variable \( z \) represents the exposure to some set of independent variables, while \( f(z) \) represents the probability of a particular outcome, given that set of explanatory variables. The variable \( z \) is a measure of the total contribution of all the independent variables used in the model and is known as the logit.

The variable \( z \) is usually defined as

\[ z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k, \]

Where \( \beta_0 \) is called the "intercept" and \( \beta_1, \beta_2, \beta_3 \), and so on, are called the "regression coefficients" of \( x_1, x_2, \) and \( x_3 \) respectively. The intercept is the value of \( z \) when the value of all independent variables is zero (e.g. the value of \( z \) in someone with no risk factors). Each of the regression coefficients describes the size of the contribution of that risk factor. A positive regression coefficient means that the explanatory variable increases the probability of the outcome, while a negative regression coefficient means that the variable decreases the probability of that outcome; a large regression coefficient means that the risk factor strongly influences the probability of that outcome, while a near-zero regression coefficient means that that risk factor has little influence on the probability of that outcome.

Logistic regression is a useful way of describing the relationship between one or more independent variables (e.g., age, sex, etc.) and a binary response variable, expressed as a probability, that has only two values, such as having cancer ("has cancer" or "doesn't have cancer").

**Conclusion**

We described a new tool for generating and predicting software failures. When this tool is used with early releases of software, the predictive models proved useful in predicting software failures and in forecasting software readiness. We feel the tool’s underlined approach (DOE and logistic Regression) is applicable to most applications especially if the tool is used to test early software releases. However, to increase our confidence in the results, we will conduct and examine several case studies with different types of applications. Finally we can say both tools are important on their places and their performances are better in their own. But due to some lack of time it is important to design such a software testing tool, which is capable to test the simple JSP applications too.

**References**


