ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2013

An Approach for Real Time Testing Reliability & Usability Testing Process

R.Karthikeyan[1], Dr.S.R.Suresh[2]

[1]Assistant Professor, Dept of CSE, Bharath University, Chennai, TN, India,

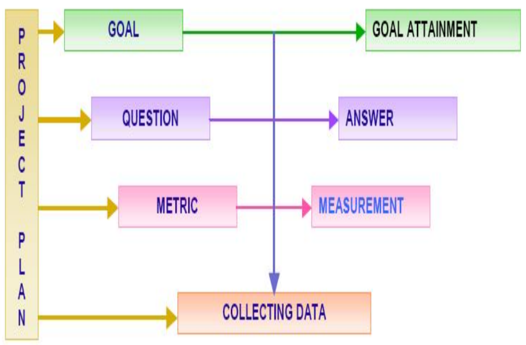
[2]Dean, Dept of MCA, Bharath University, Chennai, TN, India

ABSTRACT: The criticality of correct, complete, testable requirements is a fundamental ideology of software engineering. Also the complete quality of the system is checking the process and product metrics of the final product. Modern tools for managing requirements allow new metrics to be used in support of both of these critical processes. Using these tools, potential problems with the quality of the requirements and the test plan can be identified early in the life cycle. Some of these quality factors include: ambiguous or incomplete requirements, poorly designed requirements databases, excessive or insufficient test cases, and incomplete linkage of tests to requirements. This thesis discusses how metrics can be used to evaluate the quality of the requirements and test to avoid problems later. Test metrics are very powerful "Risk Management Tool". They help you to measure the current performance of the system. Because today's data become the tomorrow's historical data. This data can be used to improve our future work that estimates the quality levels. Keywords: Metrics, GUI, Quality, risk management.

I. EVALUATION OF TEST METRICS

The test metrics evaluation is based on the following tests is conducted in the Window/Web applications with their functional and performance using the manual and automation testing tools and find out the effective testing is Manual or Automation for various Window/Web applications. This decision is considered based on the time expending and resources consumed during testing [3]. Based on the situation of system it is classified into Manual testing, Automation testing. Each of these is further categorized based on the focus area:

- Productivity
- Quality.
- People.
- Environment.
- Tools.



The following testing techniques selected for this thesis.

- System Testing (or) Functional testing.
- Performance testing.
- GUI testing.
- Regression testing.

System testing

- Database testing.
- Security testing.
- Recovery Testing.

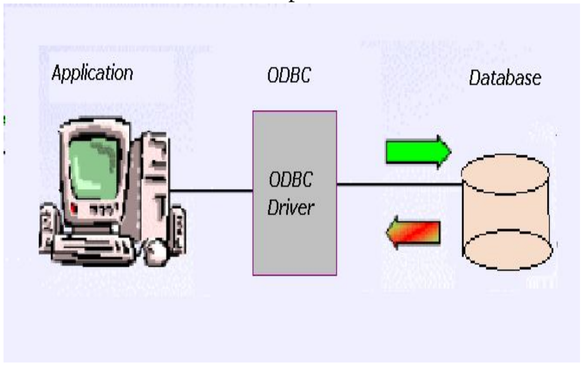
Database testing

To check the functionality of the database in Window/Web applications. Those following tests are conducted in this type of the testing.

- Scaffolding code (e.g. triggers or updateable views) which support refactoring.
- Database methods such as stored procedures, functions, and triggers
- Existence of database schema elements (tables, procedures, ...)
- View definitions.
- Referential integrity (RI) rules

Default values for a column

To test the database with above conditions we should write separate database test cases for different situation.



Classical model of Database Application

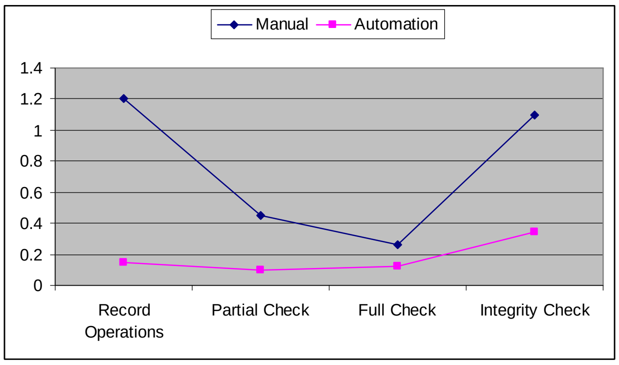
Along with the tremendous growth of the World Wide Web (WWW) over the past decade, there has been a growth and transformation in the nature of services accessible over the Web. Many new services are web sites that are driven from data stored in databases. Examples of such web applications include services that provide access to large data repositories, E-commerce applications such as online stores, and business-to-business (B2B) support products[6].

It is essential that these applications function correctly and provide suitable protection to customer data and enterprise assets. Increasingly, communicating with other components via open protocols to form large distributed systems [11]. Testing individual components comprising a web service is an important aspect of validating the service as a whole, along with testing the integration of the various components [6].

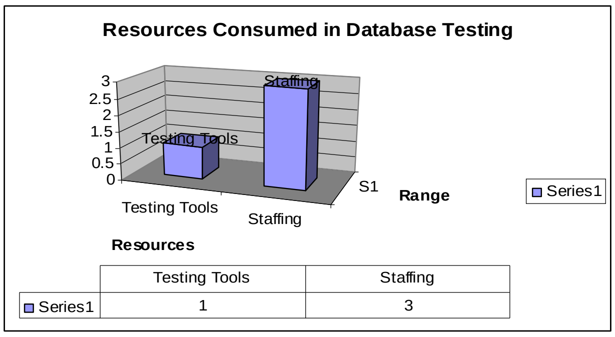
Most web database applications consist of three layers of application logic. At the base is a Database Management System (DBMS) [1] and a database. At the top is the client web browser used as an interface to the application. Between the two lies most of the application logic, usually developed with a web server-side scripting language or Java extended with library that can interface with the DBMSs [11].

Report analysis of Manual & Automation

Here the following report has been created based on the separate test execution of the same database application with different database test cases traceable to the requirement specification.



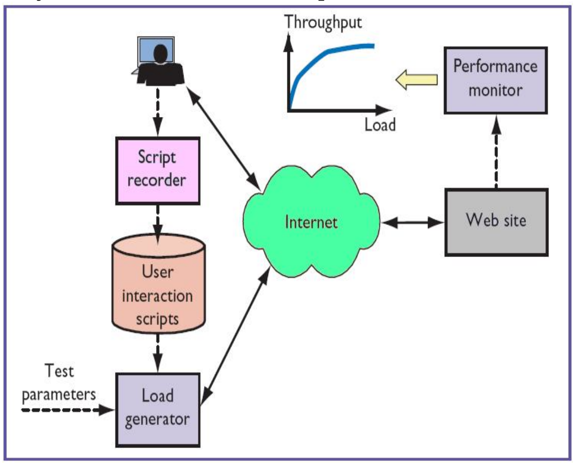
Resources consumed in database testing



	Testing Tools	Staffing
▨ Series1	1	3

Performance Testing:

Performance Testing is conducted to evaluate the compliance of a system or component with specific performance requirements. Performance testing can be undertaken to: 1) show that the system meets specified performance objectives, 2) tune the system, 3) determine the factors in hardware or software that limit the system's performance, and 4) project the system's future load- handling capacity in order to schedule its replacements"[10]



The load-testing process. The script recorder creates user interaction scripts based on actual requests.
Load testing Process:

(i) Plan to load test.
(ii) Create Test
(iii) Define scenario.
(iv) Execute Scenario.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2013

(v) Analyze the test results.

Plan to load test:

Define your performance testing requirements. For example number of concurrent users to test the web applications and required response time.

Create Test:
Based on the plan then create the test by either Manual or automation testing tools.

Define the Scenario:
Setup the load test environment and define the list of the scenarios to be tested with the web application.

Execute Scenario:
Implement the all the load test requirement as per expected in the load test scenarios.

Analyze the test results.

After executing the load test then analyze the all the tests related to the specification and prepare the test reports.

Main areas to test:

In the load testing the major areas to be tested are listed here.
(a) Actual load level to Expected load level.
(b) Response time.
(c) Increased throughput per request.

Load Testing

As Figure 1 shows, load testing lets you measure your site's performance based on actual customer behavior. When customers access your site, a script recorder uses their requests to create interaction scripts. A load generator then replays the scripts, possibly modified by test parameters, against the Web site [10].

How It Works

The load generator mimics browser behavior It continuously submits requests to the Web site, waits for a period of time after the site sends a reply to the request (the think time), and then submits a new request. The load generator can emulate thousands of concurrent users to test Web site scalability. Each emulated browser is called a virtual user, which is a key load-testing concept. A load test is valid only if virtual users' behavior has characteristics similar to those of actual users [6]. You must therefore ensure that your virtual users_ follow patterns similar to real users, _ use realistic think times, and _ react like frustrated users, abandoning a Web session if response time is excessive. Failure to mimic real user behavior can generate totally inconsistent results. Because customers who abandon a session use fewer site resources than those who complete it, for example, planning your infrastructure capacity assuming that all started sessions will be completed can lead you to overprovision the site. Also, if you fail to consider session abandonment, you cannot accurately quantify important business metrics1 such as: _ revenue throughput, [9] which measures the amount of money a Web site generates per unit time (dollars per second, for example), and potential lost revenue throughput,1 which is the amount of money in customers' shopping carts that was not converted into sales per unit time due to session abandonment. During the time a Web site is subject to the load generated by virtual users, we measure its performance and obtain metrics such as response time and throughput for each load intensity value — that is, based on the number of virtual users [9].

When to Use It

Several circumstances call for load testing. Suppose, for instance, that you anticipate a significant traffic increase to your site following a marketing campaign. In place of what is now a peak of 3,000 session [5] starts per hour, you're expecting twice that. Currently, your dial-up customers experience an average 6.5-second response time on search requests, the most critical e-business function. What will be the response time when the site's load increases to 6,000 sessions per hour? As another example, suppose that you're adding new functionality to the site or redesigning Web pages. You must know how this will affect response time before your customers find out; doing so lets you detect potential performance problems and fix them before they occur. Another good time to perform load testing is when you plan to implement IT infrastructure changes [10].

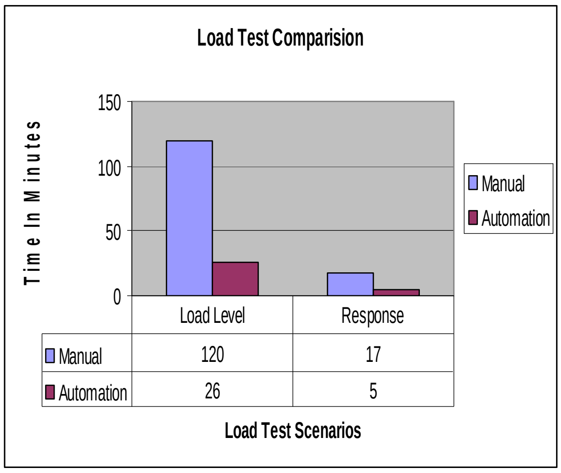
Testing Parameters and Results

There are three main parameters to vary during a load test:

(i) Workload intensity, typically measured in session starts per hour.
(ii) Workload mix, described by the scripts, which
☎)()()➂ (Customer behavior parameters, including abandonment threshold and think time.

Typical load test results include

(a) Number of completed and abandoned sessions per hour, as a function of the number of started sessions per hour [12].
(b) Revenue and potential lost revenue throughput, as a function of the number of sessions started per hour [12].
(c) Individual page download times and transaction completion times versus the number of sessions started per hour [12].



	Load Level	Response
Manual	120	17
Automation	26	5

GUI Testing:

Graphical User Interfaces are widely used and it is essential to ensure their correct operation. Manual inspections, usability evaluations and testing are employed to check a GUI for correctness. In the approach of testing a GUI, test cases are created and executed, either manually or automatically, to find errors in the GUI [7]. A common approach to test a GUI is Interaction Testing. Some ways used to carry out interaction testing are based on Finite state machines, Variable Finite State Machines, Genetic Algorithms and AI Planning [7].
A correlation between event coverage and statement coverage is attempted.

Challenges in GUI Testing:

- Traditional coverage criteria may not work well for GUI testing because what matters is not only how much of the code is tested but whether the tested code corresponds to potentially problematic user interactions [1].
- Regression testing
- GUI test case execution requires that the test oracle invocation and test execution be interleaved

A suitable representation of a GUI Graphical user interfaces have become a nearly ubiquitous means of interacting with software systems. The GUI responds to user events, such as mouse movements or menu selections, providing a front end to the underlying application code [8]. The GUI interacts with the underlying code through messages or method calls.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2013

II. TOOLS AND TECHNIQUES

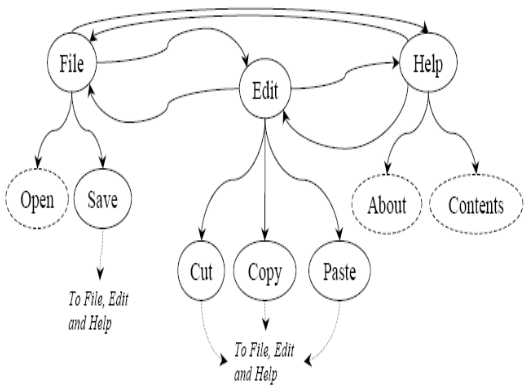
Current GUI testing techniques are incomplete, ad hoc, and largely manual. The most common tools use record-playback techniques. A test designer interacts with the GUI, generating mouse and keyboard events. The tool records the user events, captures the GUI session screens, and then stores the session—usually as a script [8].

The tester later plays back the recorded sessions to re-create the events with different inputs. This process is extremely labor intensive, often relying on the test designer's ability to generate interesting GUI interactions.

A popular alternative is to release beta copies of the software and let the users do part of the testing. For example, Microsoft tested part of its Windows 95 software by releasing almost 400,000 beta copies [8].

GUI Checklist:

Mostly we need the complete test plan to test every part of the system. Mostly GUI testing is very difficult testing process to test huge projects. A sample outline of the MS-WORD application and an "Event-Flow Graph" representation for the GUI test suite [8].



Sample MS-WORD Event Flow Graph

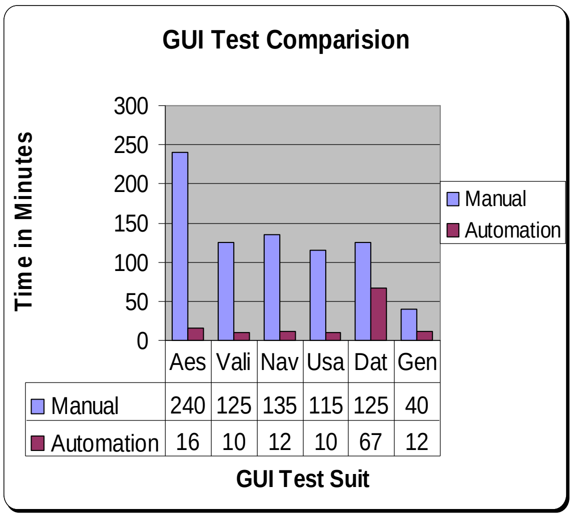
GUI in Automation

Multiple avenues of approach have been used in the attempt to conquer the test automation problem. Here some of those approaches will be presented and discussed shortly. Since maintaining GUI tests is expensive when doing regression testing, some have tried to automate the generation of tests. Other approach to the same problem is to model the test cases so that they can be automatically repaired when changes are done to the UI. Both of these solutions usually are based on the modeling of the UI as a set of events and states , which are then combined into "components" , which can be tested separately. The common feature is that these events sometimes have to be tested in sequence (like cut-paste) to be able to test all of the functionality [7]. Naturally for auto generated tests this means that the event sequences could grow to infinite length. Some limit has to be imposed and according to studies there is no point in testing very long event sequences except in special cases. What they found out is that event coverage (each event tested at least once) guarantees very high statement coverage even at sequence length of 1. Lengthening the sequence makes the statement coverage approach 100% slowly [7].

Main areas to test in GUI test coverage

In the GUI testing the following tests need to uncover the flaw in the application traceable to the specification

(a) Aesthetic conditions.
(b) Validation Conditions.
(c) Navigation Conditions.
(d) Usability conditions.
(e) Data integrity conditions.
(f) General conditions.



	Aes	Vali	Nav	Usa	Dat	Gen
▨ Manual	240	125	135	115	125	40
■ Automation	16	10	12	10	67	12

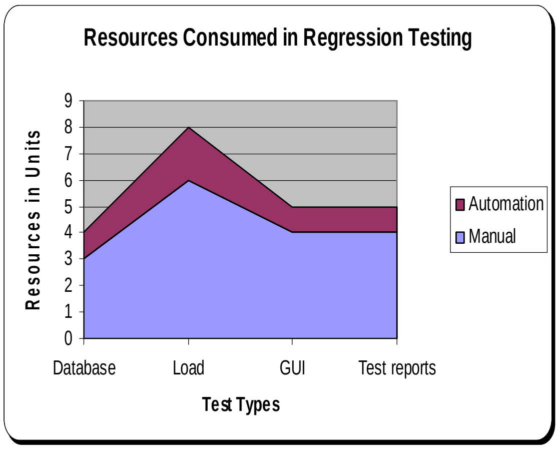
Regression Testing:

Regression testing conducted for the purpose of evaluating whether or not a change to the system (all CM items) has introduced a new failure. Regression testing is often accomplished through the construction, execution and analysis of product and system tests. More about Regression Testing Reengineering [11]. The process of examining and altering an existing system to reconstitute it in a new form. May include reverse engineering (analyzing a system and producing a representation at a higher level of abstraction, such as design from code), restructuring (transforming a system from one representation to another at the same level of abstraction), recommendation (analyzing a system and producing user and support documentation), forward engineering (using software products derived from an existing system, together with new requirements, to produce a new system), and translation (transforming source code from one language to another or from one version of a language to another) [11].

Why use regression tests?

❖ Manage risks of change:
  (a) a bug fix didn't fix the bug or
  (b) The fix (or other change) had a side effect or
  (c) Error in the build process
  (d) Faulty localization

* The essence of regression testing is repetition of tests after changes.

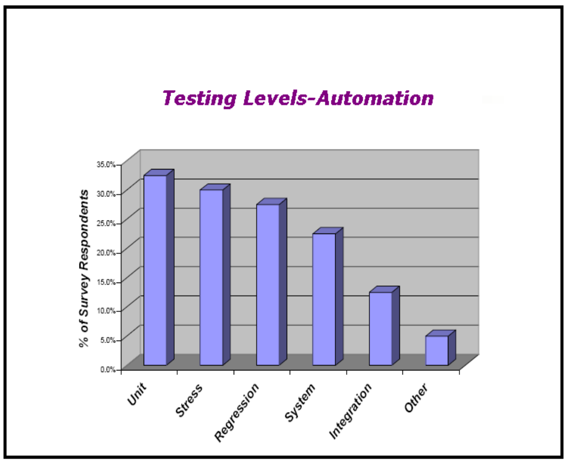


ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2013

III.COMPARATIVE ANALYSIS OF REGRESSION TESTING

This report shows that the resources consumed in the regression testing for the both of the manual and automation testing. Based on this the report the test metrics for the each test execution related to the application need to quantify the test metrics [4]. Manual testing consumes the most of the resources and timeline for testing the application as per expected in requirement specification. But manual testing is suitable and better than automation when we conduct the GUI testing, because when the application appearance changes related to the user interface design then automation need to re-record the scenario once again. But for the load testing and database testing the automation testing is suitable and we can test and take the all necessary reports related to the application in over night.



IV.CONCLUSION

Testing is one of the most important and time consuming steps in software development. Mostly based on the regression testing and the other testing reports related to the resources consumed in the testing process and the timings spent for the each test execution for the automation testing is mostly suitable for the huge projects and it is reusable, reliable and comprehensive. The manual testing is very suitable for small projects. Because the manual testing requires the heavy investment of the human resources and cost to other thinks. Mostly Automation is effective for regression testing. so inference taken from the regression testing reports for manual and automation testing.

REFERENCES

[1] G. M. Kapfhammer and M. L. Soffa. A family of test adequacy criteria for database-driven applications. In ESEC/FSE, Sept. 2002.
[2] Y. Deng, P. G. Frankl, and D. Chays. Testing database transactions with agenda. In ICSE, pages 78–87, 2001.
[3] Gregory M. Kapfhammer. Software Testing. CRC Press Computer Science Handbook. June, 2001.
[4] F. Haftmann, D. Kossmann, and A. Kreutz. Efficient regression tests for database applications. In CIDR, pages 95–106, 2000.
[5] S. Elbaum, S. Karre, and G. Rothermel. Improving web application testing with user session data. In Proceedings of the 25rd international conference on Software. IEEE Computer Society Press, 2002.
[6] Y. Wu and J. Offutt. Modeling and testing web-based applications. In CS Technical Report. George Mason University, 2001.
[7] Atif M.Memon A Comprehensive Framework for Testing Graphical User Interfaces, Ph.D Thesis, 2001, University of Pittsburgh.
[8] Atif M. Memon, Mary Lou Soffa and Martha E. Pollack Coverage Criteria for GUI Testing, 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering 2001.
[9] D.A. Menascé and V.A.F. Almeida, Capacity Planning for Web Services: Metrics, Models, and Methods, Prentice Hall,Upper Saddle River, N.J., 2002.