International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 10, October - 2013

An Empirical Approach Software Testing Methods And Techniques

KiranKumar Kishor Tambalkar,[1] Srinivas Babu Kasturi[2]

Malla Reddy Institute Of Technology/Computer Science Engineering, Hyderabad India

Abstract :- This Paper looks at what Software Testing is and then briefly explains some of the more common methods and techniques used in the testing process. First we look at the traditional (procedural) environment and then the Object Oriented environment. Comparisons are made (where appropriate) between the two models and methods and techniques used. Lastly some issues with testing under both models are discussed.

KEYWORDS: Software Testing, Level of Testing, Testing Technique, Testing Process

IJERT

Introduction:-

Software has infiltrated almost all areas in the industry and has over the years become more and more wide spread as a crucial component of many systems. System failure in any industry can be very costly and in the case of critical systems (flight control, nuclear reactor monitoring, medical applications, etc.) it can mean lost human lives. These "cost" factors call for some kind of systems failure prevention. One way to ensure system's reliability is to extensively test the systems. Since software is a system component, it requires testing process also.

Software testing is a critical component of the software engineering process. It is an element of software quality assurance and can be described as a process of running a program in such a manner as to uncover any errors. This process, while seen by some as tedious, tiresome and unnecessary, plays a vital role in software development.

Testing should be based on user requirements. This is in order to uncover any defects that might cause the program or system to fail to meet the client's requirements.

Testing time and resources are limited. Avoid redundant tests. It is impossible to test everything. Exhaustive tests of all possible scenarios are impossible, simple because of the many different variables affecting the systems and the number of paths a program flow might take.

Use effective resources to test. This represents use of the most suitable tools, procedural and individuals to conduct the test. The test team should use tools that they are confident and familiar with. Testing procedures should be clearly defined. Testing personnel may be a technical group of people independent of the developers.

Testing should begin at the module. The focus of testing should be concentrated on the smallest programming units first and then expand to other parts of the system.

We look at software testing in the traditional (Procedural) sense and then describe some testing strategies and methods used in Object Oriented environment.

There are Two Testing methods **Functional Testing** and **Structural Testing**.

Functional Testing-

- Functional testing, tests are designed from functional point of view.
- It is also known as Black box testing or closed box or opaque box testing.
- This is equivalent to partitioning boundary value analysis cause-effect Graphing techniques and comparison testing.

Structural Testing-

- Structural testing, tests are designed from structural point of view.
- It is also known as white box testing or glass box or open box testing.
- This is basis path testing loop testing and control structure testing.

Definition and the goal of testing-

The general aim of testing is to affirm the quality of software systems by systematically exercising the software in carefully controlled circumstances.

Miller's description of testing views most software quality assurances activities as testing. He contends that testing should have the major intent of finding errors. A good test is one that has a high probability of finding an as yet undiscovered error, and a successful test is one that uncovers an as yet undiscovered error, This general category of software testing activities can be further divided. For purposes of this paper, testing is the dynamic analysis of a piece of software, requiring execution of the system to produce results, which are then compared to expected outputs.

IJERT

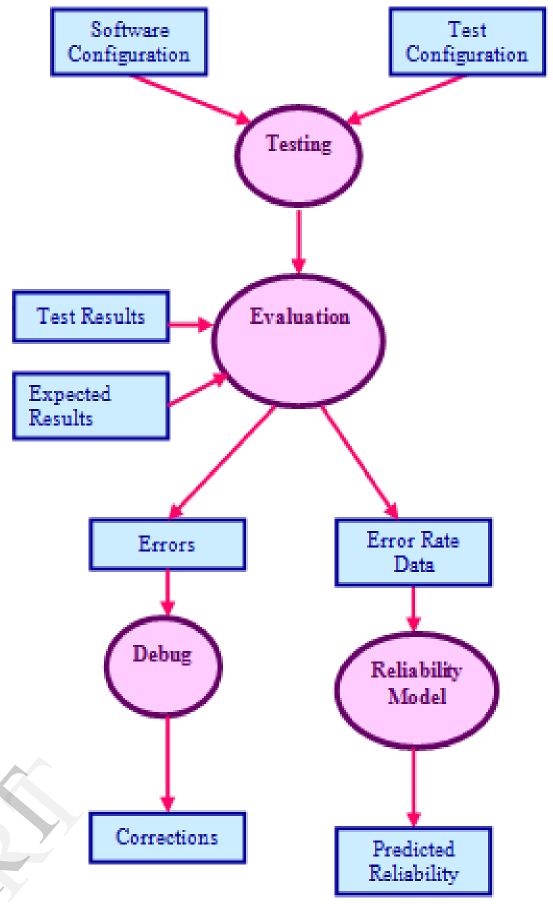


Figure 1.Test Information Flow

The Testing Spectrum-

Testing is involved in every stage of software life cycle but the testing done at each level of software development is different in nature and has different objectives.

- **Unit Testing** is done at the lowest level. it tests the basic unit of software, which is the smallest testable piece of software, and is often called "unit", "module", or "component" interchangeably.
- **Integration Testing** is performed when two or more tested units are combined into a larger structure. The test is often done on both the interfaces between the components and the larger structure being constructed, security, and maintainability, are also checked.
- **System Testing** tends to affirm the end to end quality of the entire system. system test

is often based on the functional requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability, are also checked.

- **Acceptance Testing** is done when the completed system is handled over from the developers to the customers or users. The purpose of acceptance testing is rather to give confidence that the system is working than to find errors.

Testing Methods-

Test cases are developed using various test techniques to achieve more effective testing. By this software completeness is provided and conditions of testing which get the greatest probability of finding errors are chosen. So testers do not guess which test cases to chose, and test techniques enable them to design testing conditions in a systematic way. Also, if one combines all sorts of existing test techniques , one will obtain better results rather if one uses just one test technique.

There are many ways to conduct software testing, but the most common methods. rely on following steps.

- **Test Case Design**
- **Structural Testing (White-Box Testing)**
- **Basis Path Testing**
- **Control structure Testing**
- **Functional Testing (Black Box Testing)**

Test Case Design

Test cases should be designed in such a way as to uncover quickly and easily as many errors as possible. They should "exercise" the program by using and producing inputs and outputs that are both correct and incorrect.

Variables should be tested using all possible values (for small ranges) or typical and out-of-bound values (for larger ranges). They should also be tested using valid and invalid types and conditions. Arithmetical and logical comparisons should be examined as well, again using both correct and incorrect parameters. The objective is to test all modules and then the whole system as completely as possible using a reasonable wide range of conditions.

Structural Testing

white box methods relies on intimate knowledge of the code and a procedural design to derive the test cases. It is most widely utilized in unit testing to determine all possible paths within a module, to execute all loops and to test all logical expressions.

Using white-box-testing, the software engineer can

- Guarantee that all independent paths within a module have been exercised at least once
- Examine all logical decisions on their true and false sides
- Execute all loops and test their operation at their limits
- Exercise internal data structures to assure their validity (Pressman, 1997)

IJERT

This form of testing concentrates on the procedural detail. However, there is no automated tool or testing system for this testing method. Therefore even for relatively small systems, exhaustive white-box testing is impossible because of all the possible path permutations.

Basis path Testing

Basis path testing is a white box technique. It allows the design and definition of a basis set of execution paths. The test cases created form the basis set allow the program to be executed in such a way as to examine each possible path through the program by executing each statement at least once (pressman, 1997)

To be able to determine the different program paths, the engineer needs a representation of the logical flow of control. The control structure can be illustrated by a flow graph. A flow graph can be used to represent any procedural design.

Control Structure Testing

Because basis path testing alone is insufficient, other techniques should be utilized.

Condition testing can be utilized to design test cases which examine the logical conditions in a program. It focuses on all conditions in the program and includes testing of both relational expressions and arithmetic expressions.

This can be accomplished using branch testing and domain testing methods. Branch testing executes both true and false branches of a condition. Domain testing utilizes values on the left-hand side of the relation by making them greater than, equal to and less then the right hand side value. This method test both values and the relation operators in the expression.

Data flow testing method is effective for error protection because it is based on the relationship between Loop testing method concentrates on validity of the loop structures.

Functional Testing

Black box on the other hand focuses on the overall functionality of the software. That is why it is the chosen method for designing test cases used in functional testing. This method allows the functional testing to uncover faults like incorrect or missing functions, errors in any of the interfaces, errors in data structures or databases and errors related to performance and program initialization or termination.

IJERT

Using Black-box testing, the software engineer can

- Performing the tests which exercise all functional requirements of a program
- Finding the following errors
    1. Incorrect or missing functions
    2. Interface errors
    3. Errors in data structures or external database access
    4. Performance errors
    5. Initialization and termination errors.

Research Strategies for testing techniques

We have studied the research strategies of twelve influential papers since the year 1975 and tried to find out the common form and successful examples of research settings that offer concrete guidance for future research work. The results are listed in table it's not hard to find out that most of researches on testing techniques are motivated by questioning if there is a better method of doing something. The questions is answered by inventing, implementing, combining, referring, evaluating, alternating or proposing new ways to do this task, and the result gets analyzed through analysis most of times. Combined with the test gap mentioned in last section, we contend that fundamental researches should address the challenge testing techniques are facing in the real world, generalize them and pursue practical solutions for them. Research should be carried out with industry partners on real world problems, instead of simple toy systems.

Researchers in academic community and in industry should talk often to address the need for each other.

Paper Ref.	Year	Age	Idea	Validation
GG75	1975	26	Fundamental Theorem	Analysis
Huang 75	1975	26	Edge approach Evaluation Analytic Model	Analysis
Howden 75	1976	25	Path Approach Reliability Method	Analysis
WC80	1980	21	Domain Testing strategy	Analysis
Howden 80	1980	21	Functional design abstraction method	Persuasion

RW85	1985	16	Data Flow Strategy	Analysis
ROT89	1989	12	Integrate spec. impl. testing method	Analysis
JM94	1994	7	Coverage reliability estimation method	Analysis
BBL97	1997	4	Probabilistic functional testing method	Analysis
BIMR97	1997	4	Testing based on architectural method	Persuasion
HIM00	2000	1	UML based technique	Experience
BG01	2001	0	Component based testing	Analysis

Table No. 1 Research Stratergies

Test Techniques According to the Project of the IEEE Computer Society, 2004

The IEEE Computer Society is established to promote the advancements of theory and practice in the field of software engineering.

The Society completed IEEE Standard 730 for software quality assurance (it is any systematic process of checking to see whether a product or service being developed is meeting specified requirements, see) in the year 1979. This was the first standard of this society The purpose of IEEE standard 730 was to provide uniform minimum acceptable requirements for preparation and content of software quality assurance plans. Another, new standards are meaningful not only for promotion software requirements, software design and software construction, but also for software testing, software construction, but also for software testing, software maintenance,

So for improving software testing and for decreasing risk on all fields, there is classification of test techniques according to this Society, which is listed below.

IJERT

- Based on the software engineer's intuition and experience
  - i. **Ad hoc testing-**
    Test cases are developed basing on the software engineer's skills, intuition, and experience with similar programs.
  - ii. **Exploratory testing-**
    This testing is defined like simultaneous learning, which means that test are dynamically designed, executed, and modified.
- Specification-based techniques
  1. **Equivalence Partitioning**
  2. **Boundary -value analysis**
  3. **Decision table**

  Decision tables represent logical relationships between inputs and outputs (Conditions and actions), so test cases represent every possible combination of inputs and outputs.

  4. **Finite-state machine-based**

  Test cases are developed to cover states and transitions on it.

  5. **Testing from formal specifications**

  The formal specifications (the specifications in a formal language) provide automatic derivation of functional test cases and a reference output for checking test results.

  6. **Random testing**

  Random points are picked within the input domain which must be known, so test cases are based on random.

- Code-based techniques
  1. **Control-flow based criteria**
     Determine how to test logical expressions (decision) in computer program. Decisions are considered as logical functions of elementary

logical predicates (conditions) and combinations of condition's values are used as data for testing of coverage requirement as a component part every statement in the program has been executed at least once. Control-flow criteria are considered as program-based and useful for white-box testing. For control-flow criteria, the objects of investigation have been relatively simple. Random Coverage, Decision Coverage (every decision in the program has taken all possible outcomes at least once), Condition Coverage (every condition in each decision has taken all possible outcomes at least once), Decision Condition Coverage (every decision in the program has take all possible outcomes at least once and every condition in each decision has taken all possible outcomes at least once), etc.

2. **Data Flow-based criteria**

3. **Reference models for code-based testing**

This means that the control structure of a program is graphically represented using a flow graph

- Fault-based techniques
  1. **Error guessing**

Test cases are developed by software engineers trying to find out the most frequently faults in a given program. The history of faults discovered in earlier projects and the software engineer's expertise are helpful in this situations.

A mutant is a modified version of the program under test. It is differing from the program by a syntactic change. Every test case exercises both the original and all generated mutants. Test cases are generating until enough mutants have been killed or test cases are developed to kill surviving mutants.

- Usage-based techniques
  1. **Operational profile**

From the observed test results someone can infer the future reliability of the software.

2. **Software Reliability Engineered Testing**

- Techniques based on the nature of the application

By this test we can find where the element under test does not perform as specified. Besides the goal of this technique is to select, structure and organize the tests to find the errors as early as possible.

2. **Component-based testing**

Is based on the idea of creating test cases from highly reusable test components. A test component is a reusable and context-independent test unit, providing test services through its contract based interfaces. More about this test technique on http://www.componentbasedtesting.org/site/.

3. **Web-based testing**

Is a computer based test delivered via the internet and written in the "language" of the internet HTML and possibly enhanced by scripts. The test is located as a website on the tester's server where it can be accessed by the test-taker's computer, the client. The client's browser software (e.g. Netscape navigator, MS internet Explorer**)** displays the test**,** the test taker completes it, and if so desired sends his/her answers back to the

IJERT

server, from which the tester can download and score them.

4. **GUI testing**

Is the process of testing a product that uses a graphical user interface, to ensure it meets its written specifications.

5. **Testing of concurrent programs**
6. **Protocol conformance testing**

A protocol describes the rules with which computer systems have to comply in their communication with other computer systems in distributed systems. **Protocol conformance testing** is a way to check conformance of protocol implementations with their corresponding protocols standards, and an important technology to assure successful interconnection and interoperability between different manufacturers. Protocol conformance testing is mostly based on the standard ISO 9646. Conformance testing Methodology and Framework. However this conventional method of standardization used for protocol conformance test, sometimes gives wrong test result because the test is based on static test sequences.

IJERT

7. **Testing of real-time systems**

systems require time a special attention must be given to timing during testing. components) at its normal operating frequency, speed or timing. But it is actually a conformance testing, which goal is to check whether the behavior of the system generated offline or online. In the first case, the complete test scenarios and verdict are computed a-prior and before execution. The offline test generation is often based on coverage criterion of the model, on a test purpose or a fault model. Online testing combines test generation and execution.

8. **Testing of safety-critical systems**

- Selecting and combining techniques

1. **Functional and structural**

Functional testing (also known as black-box testing), is a software testing approach in which:

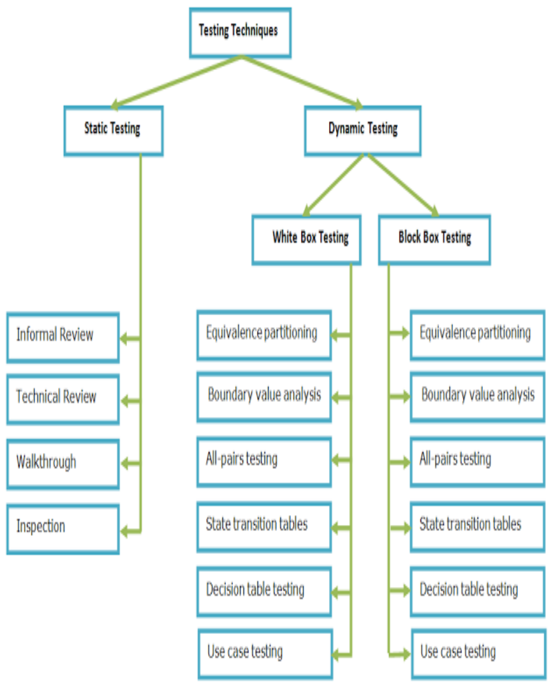
1. the tester will have a user perspective in mind,
2. not knowing and doesn't mind how the program works.
3. Input and output are the only things that matter.
4. The tester acts as if he/she is the final user of the program.

On the other hand, structural testing (also known as white-box testing), is a software testing approach in which:
1. the tester will have a developer perspective in mind,
2. knowing how the program works behind the scene,
3. such that the test will test all algorithm paths in the program.
4. Everything does matter.
5. The tester acts as a developer of the program who knows the internal structure of the program very well.

2. **Deterministic Vs. random**

Test cases can be selected in a deterministic way or randomly. drawn from some distribution of inputs, such as is in reliability testing

Fig 2. Testing Techniques



Testing Results

Testing documentation involves the documentation of artifacts which should be developed before or during the testing of Software.

Documentation for Software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing etc. This section includes the description of some commonly used documented artifacts related to Software testing such as:

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix

Test Plan

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, the limitations of the testing and the schedule of testing activities. Typically the Quality Assurance Team Lead will be responsible for writing a Test Plan.

A test plan will include the following.

- Introduction to the Test Plan document
- Assumptions when testing the application
- List of test cases included in Testing the application
- List of features to be tested
- What sort of Approach to use when testing the software
- List of Deliverables that need to be tested
- The resources allocated for testing the application
- Any Risks involved during the testing process
- A Schedule of tasks and milestones as testing is started

Test Scenario

A one line statement that tells what area in the application will be tested. Test Scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.

The term test scenario and test cases are used interchangeably however the main difference being that test scenarios has several steps however test cases have a single step. When viewed from this perspective test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test

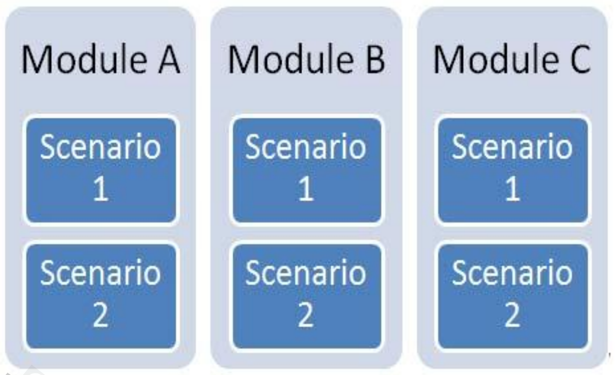


Fig 3. Testing Scenarios

IJERT

Test Case

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. The main intent of this activity is to ensure whether the Software Passes or Fails in terms of its functionality and other aspects. There are many types of test cases like: functional, negative, error, logical test cases, physical test cases, UI test cases etc.

Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing. However, following are the main components which are always available and included in every test case:

- Test case ID.
- Product Module.
- Product version.
- Revision history.
- Purpose
- Assumptions
- Pre-Conditions.
- Steps.
- Expected Outcome.

- Actual Outcome.
- Post Conditions.

Many Test cases can be derived from a single test scenario. In addition to this, some time it happened that multiple test cases are written for single Software which is collectively known as test suites.

Traceability Matrix

Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table which is used to trace the requirements during the Software development life Cycle. It can be used for forward tracing (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements). There are many user defined templates for RTM.

Each requirement in the RTM document is linked with its associated test case, so that testing can be done as per the mentioned requirements. Furthermore, Bug ID is also include and linked with its associated requirements and test case. The main goals for this matrix are:

- Make sure Software is developed as per the mentioned requirements.
- Helps in finding the root cause of any bug.
- Helps in tracing the developed documents during different phases of SDLC

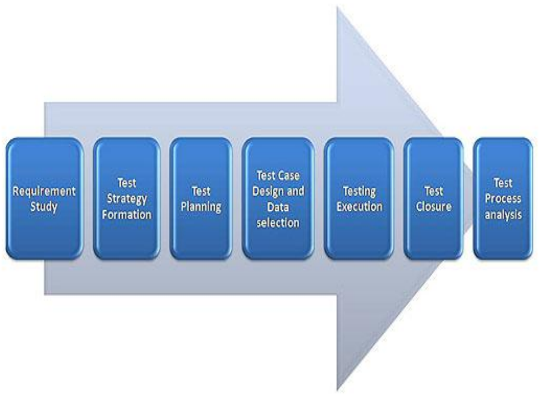


Fig 4. Testing Process

Conclusion

Testing has been widely used as a way to help engineers develop high quality systems, and the techniques for testing have evolved from an ad hoc activates means of small group of programmers to an organized discipline in software engineering.

However, the maturation of testing techniques has been fruitful. Pressure to produce higher quality software at lower cost is increasing and existing techniques used in practice are not sufficient for this purpose.

It is my intent to overstate the obvious proper testing methods must be used to conduct "true" testing. What constitutes a proper method is driven by the environment the situation and most importantly, by the objectives.

IJERT

References

[1] Suresh Chand Gupta, Prof. Ashok Kumar,” Reusable Software Component Retrieval System”, International Journal of Application or Innovation in Engineering & Management, Volume 2, Issue 1, January 2013

[2] AnupamaKaur, HimanshuMonga, MnupreetKaur,“ Performance Evaluation of Reusable Software Components”, International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 4, April 2012.

[3] www Consortium, Extensible Markup Language (XML) 1.0, February, 2004.

[4] M.E.El-Sharkawi,N.A. El-Hadi El Tazi. LNV, “Relational Database Storage Structure for XML Documents”, 3rd ACS/IEEE International Conference on Computer Systems and Applications 2005.

[5] Liu Sainan, Liu Caifeng, Guan Liming,” A Storage Method for XML document based on Relational Database”, International Symposium on Computer Science and Computational Technology, IEEE, 2008.

[6] T. Bakota, R. Ferenc, and T. Gyimothy, “ Clone smells in Software Evolution”, 23rd International Conference on Software Maintenance (ICSM 2007), pages 24-33, IEEE Computer Society, October, 2007.

[7] www Consortium, XML schema Part:0 Prime, October, 2004.

[8] Hosam F. El-Sofany, Samir A. El- Seoud, Fayed F.M.Ghaleb, Jihad M. Al Ja’am, Sameh S. Daoud, and Ahmad M, Hasnah, “ A DOM- Based Approach of Storage and Retrieval of XML Documents using Relational Databases”, International Journal of

Computing & Information Sciences Vol.5, No.2, August 2007.

[9] Irena Mlynkova, JaroslavPokorny, " From XML schema to Object Relational Database – An XML schema driven mapping", 2004.

[10] AnjuShri, Parvinder S. Sandhu, Vikas Gupta, SanyamAnand, "Prediction of Reusability of Object Oriented Software Systems using Clustering Approach" World Academy of Science, Engineering and Technology 43, 2010.

[11] Parvinder Singh Sandhu and Hardeep Singh, "Software Reusability Model for Procedure Based Domain-Specific Software Components", International Journal of Software Engineering & Knowledge Engineering (IJSEKE), Vol. 18, No. 7, 2008, pp. 1–19.

[12] T. Bakota, R. Ferenc, and T. Gyim´othy, " Clone smells in software evolution" , in Proceedings of the 23rd International Conference on Software Maintenance (ICSM 2007), pages 24–33, IEEE Computer Society, Oct. 2–5, 2007.

[13] Mylopoulos, J., Chung, L., Yu, E.: From object-oriented to goal-oriented requirements analysis, CACM **42** (1999) 31–37

[14] Parvinder Singh Sandhu and Hardeep Singh, "A Reusability Evaluation Model for OO-Based Software Components", International Journal of Computer Science, vol. 1, no. 4, 2006, pp. 259-264.

[15] Parvinder S. Sandhu, Parwinder Pal Singh,hardeep Singh, "Reusability Evaluation With Machine Learning Techniques", WSEAS TRANSACTIONS on COMPUTERS, issue 9, Volume 6, September 2007.

[16] Frakes, W.B. and Kyo Kang (2005) "Software Reuse Research: Status and Future", IEEE Trans. Software Engineering, vol. 31, issue 7, July 2005, pp. 529 - 536.

[17] James F Peters, WitoldPedrycz, " Software Engineering, An Engineering Approach", Wiley India Private Limited, 2007.

[18] Parvinder Singh Sandhu and Hardeep Singh, "Automatic Quality Appraisal of Domain-Specific Reusable Software Components", Journal of Electronics & Computer Science, vol. 8, no. 1, June 2006, pp. 1-8.

[19]. Pressman, R.S. (1997). Software Engineering, a practitioner's approach. U.S.A, McGraw Hill.

[20]. Humphrey, W.S. (1989). Managing the software process. U.S.A., Eddision-Wesley.

[21]. Marick, B. (1995a). Testing foundation, Part 1. www.stlabs.com/MARICK/1-fauld.htm

[22]. Marick, B. (1995b). Testing foundation, Part 2. www.stlabs.com/MARICK/2-sen.htm

[23]. Binder, R.V (1994). Testing objects-oriented systems.. A status report
www.rbsc.com/pages/ootstat.html.

[24]. Binder, R.V. (1995). Object-Oriented Testing: Myth and Reality.
www.rbsc.cm/pages/myths.html

[25].
http://www.his.sunderland.ac.uk/cs0mel/comm83wk5.doc, February 08.2009

[26]. IEEE, "IEEE Standard Glossary Of Software Engineering Terminology" (IEEE std 610.12-
1990) Los Alamitos, CA: IEEE Computer society press, 1990

[27]. Myers, Glenford J., IBM systems Research Institute Lecturer in computer science, Polytechnic
Institute of New York, "The Art of software Testing", Copyright 1979, by John Wiley & Sons, Inc.

[28]. Wikipedia The Free Encyclopedia
http://en.wikipedia.org/wiki/

IJERT

Guided By

Srinivasa Babu. Kasturi received B.E degree in Computer Science and Engineering from Madras University in 2003 and M.Tech degree in Computer Science and Engineering from KLCE, in 2006. Now he is a Research Scholar from reputed University in the field of Software

Engineering. Having 10 years of teaching experience. Presently working as Associate Professor in the Department of Computer Science and Engineering, MallaReddy Institute of Technology, Hyderabad. He is a Life Member of ISTE. Published four paper at National and international journal. Attended more than 6 National conferences and published articles at conference journals. Participated at various kinds of Workshops. His research interests Software Engineering and Data Mining & Ware housing.

Presented By

KiranKumar Tambalkar

Pursuing My B.tech degree from Jawaharlal Nehru Technological University Hyderabad. Got Price in National Level Presentation at Malla Reddy Institute Of Technology in 2012. Participated at various kinds of Workshops. And Participated in The National Seminar Cum Tutorials On "Advances In Image Processing & Remote Sensing at Institution of Electronics and Telecommunication Engineers (IETE) 2012. Participated in the event of Paper Presentation in (CBIT) 2011 and my favorite subjects are software testing, Web technologies, Operating system.

IJERT