Journal of Theoretical and Applied Information Technology
10th March 2015. Vol.73 No.1
© 2005 - 2015 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN ANALYTICAL WAY TO IMPROVISE TEST EXECUTION AND REVIEW OF SOFTWARE METRICS FOR THE SOFTWARE QUALITY

CHANDU P.M.S.S.

Research Scholar, Department of CSE, Sathyabama University, Chennai, India.

chandupmss@gmail.com

ABSTRACT

The escalating density of today's software products, combined with ever-increasing costs of software breakdown have pushed the need for testing to new peaks. The successful execution of the control over software quality requires software metrics. Using effective software metrics we can monitor requirements, predict development resources, tracking development progress and minimize the maintenance cost. The main objective is to improvising the performance of testing with various attributes connection with metrics for gaining low cost and high quality Software.

The propose research work is to identify the possible measuring attributes of software test execution and test review processes. This work introduces a novel framework called vector space model, to recognize software metrics related to test execution and test review phases also to identify the support of such metrics for the measurable attributes. Moreover, it is important to analyze the assumptions in the calculation of the metrics. The metrics studied against each attribute needs to be assessed for their practicality in terms of project's context and benefits to the testing team.

Keywords: *Software metrics, Vector Space Model (VSM), software testing life cycle, Software complexity, Software Test Execution and Test Review.*

1. INTRODUCTION

Increase in competition and leaps in technology have forced companies to adopt innovative approaches to assess themselves with respect to processes, products and services. This assessment helps them to improve their business so that they succeed and make more profits and acquire higher percentage of market. Many organizations around the globe are developing and implementing different Standards to improve the quality needs of their Software. In order to face the innovative and rapidly changing challenges posed by software industry, the testing process should be able to find the schedules and costs that improve the profitability, efficiency and effectiveness of the business [3]. So, the main thing of effective testing effort is measuring and knowing what is done [16].Software metrics is the cornerstone in assessment and also foundation for any business improvement.

The below mentioned are the objectives of this paper:

1. Resolving the main phases in the Software Testing Life Cycle
2. Understanding the measurements role in software testing process improvement.
3. Investigation of test review and test execution process attributes
4. Analysis of resolutions supported by the metrics and when to gather them.
5. Observing the present metric support for the identified test execution and test review processes.

Several organizations are bringing to realize the important role that software metrics can play in planning and controlling software projects, as well as improving software processes, products and projects overtime. Such improvements results in increased productivity and quality, and reduced cycle time all of which made a company competitive in the software business [12].The main aim of this paper is to enquires the support of metrics offered for the test execution and test review activities in parallel validation as well as development testing model, which enables the organizations to have best knowledge in software testing process.

2. RELATED WORK

At high maturity organizations, metrics are expected to play a key role in overall process management as well as in managing the process of a project. It is the keystone in evaluation and foundation for any business improvement. It helps in an organization for acquiring the information needed and to improve the productivity, products and services, and to achieve the desired goal in the software life cycle model. The quality of software is corresponding to our expectations results in development process [6]. Software metrics are essential to maintain the high quality of project and also cost effective. It tells the progress of the project, so it helps to maintain the standards. To maintain the metrics, it's very important to have a communication between the teams to get details about project.

The scientific community has combined a large literature survey on software metrics [4]. The study of software metrics aggregation has been presented in the previous work. These aggregate functions metrics also can be used to review the software maintainability index [7]. Predicting the software metrics defects using with formulas introduced by Chidamber and Kemerer's [11] where the problems of linear regression depends on the linear character also between the dependent and independent variables. It is increasing the dependency among those variables. The number of defects is unconditional over the original value in the software metrics. One way to reduce cost through defects prediction is in using software metrics in general and based on the call graph in particular to predict and improve possible problems in the software design [1]. An intuitive expectation is given into class increasing from adverse number of defects than increasing from metrics. Software quality models are frequently used to calculate the threshold value of software quality [8].

The aggregate information ranging from smaller elements or methods to larger elements has been recognized by Squale E. [8]. Another popular approach in selecting distributions and fitting its parameters is to approximate the metric values observed [2]. Consolidating the different views leads us to categorized software testing life cycle into test planning, test design, test execution and test review phases. In previous work there are measurable attributes and related metrics for test planning and test design was discussed [5]. The study of this paper is to identify measurable attributes for software test execution and test review phases. And to investigate the identified metrics with related equations which can be used beneficial and valuable to the organization gaining low cost and high quality software. Such improvements results in increased productivity and quality, and reduced cycle time all of which make a company competitive in the software business.

3. SOFTWARE TESTING LIFE CYCLE

Consolidating the different views leads us to categorized software testing life cycle into test planning, test design, test execution and test review phases. In previous work there are measurable attributes and related metrics for test planning and test design was discussed [5]. The Software Testing Life cycle model architecture is illustrated in fig.1. With the help of this diagram any one can understand the various phases of STLC.

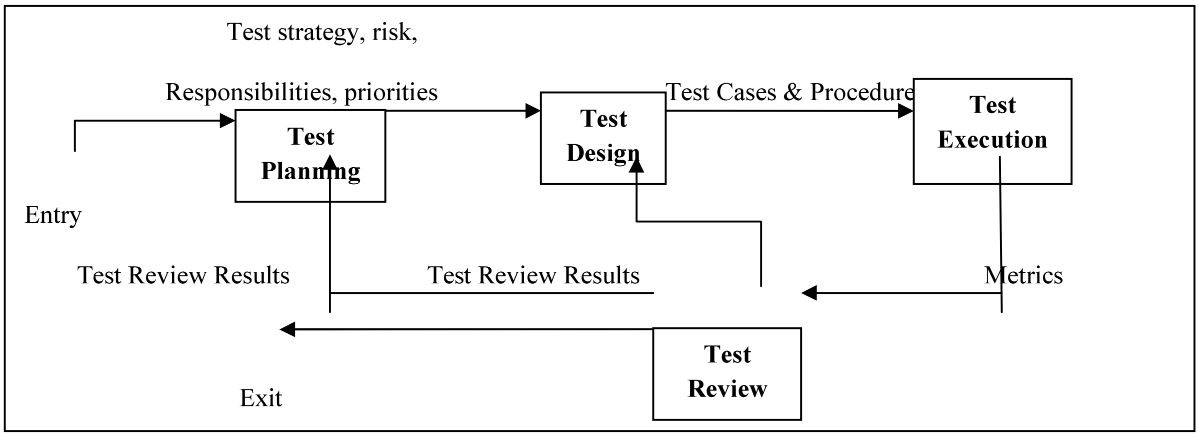

Fig. 1 Software Testing Life cycle phases [5]

Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined unambiguous rules. Software metrics can be classified into three categories.

3.1 Product metrics describes the characteristics of the product such as size, complexity, design features, performance and quality level.

3.2 Process metrics: It can be used to improve software development and maintenance i.e., the effectiveness of defect removal during development, the pattern of testing defect arrival and the response time of the fix process.

3.3 Project metrics: It is mainly to adjust the project to avoid the problems or risks and help to optimize the development plans.

4.1. Measurable Attributes For Test Execution

In this phase the testing team will handle the testing process based on the test plans organized and test cases created. Then errors will be informed back to the development team for rectification and retesting will be executed. The following are the measurable attributes which are identified in this work.

4.1.1 Progress: Parameters that help identify test progress to be matched against success criteria. Progress metrics are collected iteratively over time. Tracking the test execution progress gives early indication if the testing activity is behind schedule and to flag appropriate measures to deal with the situation.

4.1.2 Size: Size of system is usually counted as lines of code (KLOC) or function points (FP). It identifies the quantity of test cases executed, failed, passed or blocked. This is an important factor used to monitor the test execution status and represent the status of execution phase in a quantitative manner. Measures that establish the quantity of test cases executed, passed, failed are required to determine how good a test case was handled by the execution environment

4.1.3. Cost: Metrics supporting testing budget estimation are required foe test execution phase. The amount of resources and memory utilized during testing execution should be measured in order to identify cost effectiveness of tool. The resources such as operating systems, databases and programming languages used are taken into account. The measures that identify system memory utilization and total cost of various resources installation are used to represent this measurable attribute.

4.1.4 Quality: Quality is the result of high intention, sincere effort, intelligent direction and skillful execution. The effectiveness of test execution phase should be measured to identify how good the execution phase was completed. The measures that identify test cases executed per day, test procedures retested during the test execution, the number of defects accepted or valid, and rejected and test cases failed at first time execution are used to measure the quality of test execution phase.

4.2 Measurable Attributes For Test Review

The purpose of the test review process is to analyze the data collected during testing to provide feedback to the test planning, test design and test execution activities. Different assessments can be performed as part of test review which includes reliability analysis, coverage analysis and overall defect analysis [5].

4.2.1 Quality: The quality of test review phase should be measured to identify the status of test review phase. This phase identifies how good the entire software testing goals are achieved during all the three phases. The measures that measure the total lines covered by testing, the total test procedures covered, system interfaces covered, total errors discovered, capability of a code reviewer in examining the code, efficiency of the testing team in discovering the defects and the amount of system functionality successfully demonstrated.

4.2.2 Progress: The operations of test review phase should be monitored in order to identify defects and delays in the testing life cycle. The metrics that measure number of test cases executed in the test suite and the defects found per unit of time are useful in measuring the progress of the review phase.

The set of 22 measures considered in this work is listed in Table 1. These 22 measures serve as the basis for information retrieval and expert opinion.

S.NO.	Measures
1	No. of Test Cases Executed
2	No. of Test Cases Passed
3	No. of Test Cases Failed
4	No. of Test Cases Blocked
5	Defect Acceptance
6	Defect Rejection
7	First Run Failure
8	Test Execution Productivity
9	Test Case Retesting
10	Time remaining to complete the testing
11	Memory Usage of Tool
12	Test coverage
13	System Coverage Analysis
14	Test Case Execution Status
15	Error Discovery Rate
16	Inspection Rate
17	Defect Detection Rate
18	Current quality ration
19	Code Coverage
20	Test Efficiency
21	Risky Areas Identification
22	Tool Support

Table 1: Set of 22 Measures

5. PROPOSED METHODOLOGY

In the research work we are proposing a methodology called Vector Space Model. A lot of traceability recovery methods utilize the Vector Space Model as a basic algorithm. The model converts the documents into vector where each term represents dimension of vector. It classifies the metrics as per the cosine similarity between attribute and metrics documents.

This model was initially introduced by Gerard Salton [13]. In this model all related objects for a data retrieval system are indicated as vectors. The terms of vectors are words in the documents, queries. K_i (Each Term) is indicated as t-dimensional vector, where t- number of different term in the collection.

if x_r is rth term of vector c_i, then

$$Ci = (x1, x2, x3, x4\ldots x_t)$$

$$xr = 0 \Leftrightarrow r \neq i$$

$$xr = 1 \Leftrightarrow r = i$$

i.e., $c_1 = (1, 0,0,0,0,..,0)$

$$c_2 = (0, 1,0,0,0,..,0)$$

$$c_3 = (0, 0, 1, 0, 0,..,0)$$

.

.

.

$$c_t = (0,0,0,0,0,..,1)$$

The C = {c1, c2...ct} creates the canonical basis for space C^t and it is linearly unrelated with each other.

The terms are pair wise orthogonal. As a result the relevant terms are assumed to be independent. The set C of term represent the query and document vectors.

The document dj represented by the vector **dj** can be defined as:

$$dj = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

or

$$dj = a_0 + \sum_{i=1}^{t} (w_{i,j}\ c_i)$$

The vectors for query can be defined as

$$q = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$$

or

$$q = \sum_{i=1}^{t} W_{i,q} \, C_i$$

In above equations, $w_{i,q}$ and $w_{i,j}$ are the weights of ith term in query q, and document j. VSM employs TF- IDF [13] as a weighting technique since it is the most efficient information retrieval method.

The frequency of every term occurred in the document is selected for similarity calculation. Consequently the documents are linked based on the degree of similarity. The significance of a document for specific query is directly proportional to the distance among the respective vectors.

$$sim\left(d_{j,q}\right) = \frac{d_j \cdot q}{|d_j| \cdot |q|} = \frac{\sum_{i=1}^{t}(W_{i,j}, W_{i,q})}{\sqrt{\sum_{i=1}^{t}(W_{i,j}, W_{i,j})}\;\sqrt{\sum_{i=1}^{t}(W_{i,q}, W_{i,q})}}$$

More closely a document in the vector space to the query represents more degree of similarity between them. Finally the answers are returned for the set of query. Once the computation is made, it is painless to arrange a list of documents i.e. ranking and their respective degrees of relevance to the query.

In this work, metric and attribute documents are first preprocessed then given as input to the VSM. During preprocessing the terms in documents are stemmed to improve the accuracy of our classification process. VSM calculates the similarity of every metric with respect to attributes. The attribute which has the maximum cosine similarity with a chosen metric is selected and metric is classified. The metrics are compared with all the possible attributes in order to provide maximum metric support for the identified phases.

6. RESULTS & DISCUSSION

The proposed system is implemented in Java platform JDK 7 and MySQL server. The definition for metrics and attributes are stored as XML property files and they are pre-processed for VSM

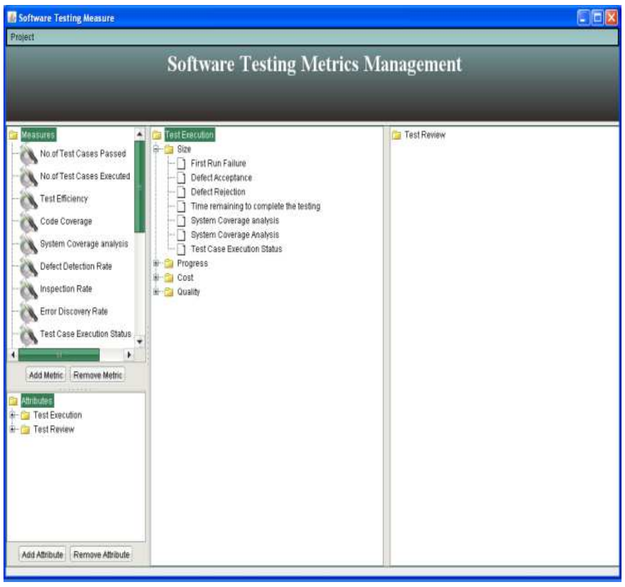

Fig. 2 Proposed Systems - VSM

Results

The Figure 2 shows the classification results of VSM achieved by the proposed system. The metrics with higher score value to the identified attribute is taken for further classification.

The Table II represents all the possible metrics involved in Test execution phase of software testing life cycle. This table displays the VSM score for each metric with respect to the identified attributes. From the listing, it's said that higher the value of similarity represents greater metric support for the corresponding attribute.

S.No	Metric	Test Execution			
		Progress	Quality	Cost	Size
1	No. of Test Cases Executed	0.931	0.75	0.132	0.124
2	No. of Test Cases Passed	0.435	0.577	0.447	0.223
3	No. of Test Cases Failed	0.435	0.471	0.447	0.288
4	No. of Test Cases Blocked	0.416	0.597	0.597	0.554
5	Defect Acceptance	0.408	0.258	0.447	0.583
6	Defect Rejection	0.408	0.258	0.447	0.583
7	First Run Failure	0.449	0.277	0.627	0.683
8	Test Case Retesting	0.234	0.892	0.544	0.231
9	Test Execution Productivity	0.955	0.088	0.111	0.124
10	Time remaining to complete the testing	0.416	0.588	0.503	0.606
11	Memory Usage of Tool	0.304	0.561	0.894	0.441
12	Tool Support	0.454	0.332	0.789	0.776

Table II Results Of VSM – Test Execution Phase

The Table III represents all the possible metrics involved in Test Review phase of software testing life cycle. When compared to test execution phase, this one has limited number of measurable attributes. The metrics are finally classified under right attributes based on their similarity scores.

S.NO	Metric	Test Review	
		Progress	Quality
1	Test coverage	0.666	0.524
2	System Coverage Analysis	0.904	0.607
3	Test Case Execution Status	0.589	0.612
4	Error Discovery Rate	0.641	0.592
5	Inspection Rate	0.614	0.339
6	Defect Detection Rate	0.449	0.071
7	Current quality ratio	0.580	0.550
8	Code Coverage	0.710	0.438
9	Test Efficiency	0.267	0.281
10	Risky Areas Identification	0.456	0.256

Table III Results Of VSM – Test Review Phase

The Figure 3 represents the graphical representation of number of metrics under attributes of test execution and test review phase. From the figure, it's clear that attribute progress achieves maximum metric support in both the execution and review phases. In order to express the success of test execution and review phases in terms of quality, size and cost additional metrics have to be identified.

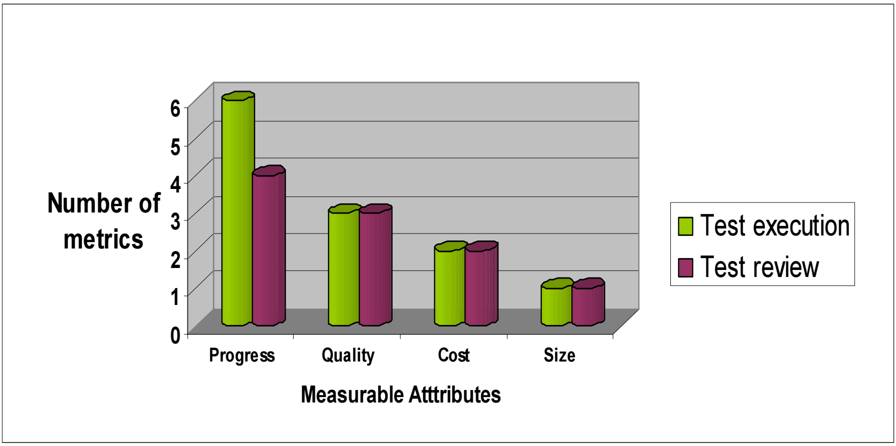

Fig.3 Metrics Classification By VSM

7. CONCLUSION

The aim of this research was to identify possible attributes in test execution and test review phases so as to provide maximum metric support for those attributes. To achieve this objective, 22 software engineering metrics and their complete description was constructed and measurable attributes were identified. Since no experimental data is available to support the problem discussed in this paper, Vector Space Model, an information retrieval technique was opted to literally classify the metrics under chosen attributes.

This proposed work will be very helpful for IT organizations in constructing the metrics plan for its software test execution and test review processes. The measurable attributes identified by this work can be an important factor to achieve successful testing process. It's observed that such an effort will direct to effective decision making about various software testing activities. The future work would be investigating how these metrics work by using related equations to enhance the confidence level of the metrics support identified by this work. The further investigation of this work is to identify appropriate metrics for test execution and test review phases to minimize time, cost and maximize the quality of the software

REFERENCES

[1] Hesham Abandah, Izzat Alsmadi2 (2013), 'Call Graph Based Metrics To Evaluate Software Design Quality', International Journal of Software Engineering and Its ApplicationsVol. 7, No. 1, January, 2013 pp.525-548.

[2] G Concas, M Marchesi (2012), ' An empirical study of software metrics for assessing the phases of an agile project', Int. Journal of Software Engineering and Knowledge Engineering,vol.22, pp.525-548.

[3]Sheikh Umar Farooq, S.M.K.Quadri,Nesar Ahmad (2011), 'Software Measurements and Metrics : Role in Effective Software Testing', Int.Journal of Engineering Science and Technology,Vol.3,No.1.

[4]Kitchenham, B.A (2010),' what's up with software metrics? – A preliminary mapping study', Journal of. Systems and Software Elsevier,, vol. 83, Issue 1, pp. 37-51.

[5]Wasif Afzal and Richard Torkar (2008), 'Incorporating Metrics in an Organizational Test Strategy', Int. Conference on Software verification and validation Workshop, pp.236-245.

[6] Paul Oman , Jack Hagemeister (2008), 'Construction and testing of polynomials predicting software maintainability', IEEE Computer Society, vol.24 , pp.251-266.

[7] Paul Oman , Jack Hagemeister (2008), 'Construction and testing of polynomials predicting software maintainability', IEEE Computer Society, vol.24 , pp.251-266.

[8]Heitlager, I., Kuipers, T., and Visser J (2007), 'A Practical Model for Measuring ... of Information and Communications Technology', IEEE Computer Society pp. 30–39.

[9] Subramanyam R and Krishnan M. S (2003), 'Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects Software', IEEE Transactions on software Engineering, vol. 29, issue: 4, pp. 297- 310.

[10]Yu P, Systa T, Muller H (2002), 'Predicting fault-proneness using oo metrics. an industrial case study. …',IEEE Computer Society ,pp.99–107.

[11]Chidamber S.R. and Kemerer F (1994), 'A Metrics Suite for Object Oriented Design', IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476- 493.

[12] Michael K.Daskalantonakia (1992), 'A Practical View Of Software Measurement and Implementation Experiences Within Motorola', IEEE Transactions on Software Engineering, Vol.18, No. 11.

[13]Salton, G. (1971). The SMART Retrieval System: Experiments in Automatic Document Processing. Englewood Cliff, NJ: Prentice Hall.

14] Myers G. J. "The Art of Software Testing". John Willey & Sons, Inc., New York, USA, 2006.

[15]S. R. Rakitin (2001), 'Software Verification and Validation for Prac- titioners and Managers . Artech House, Inc., Norwood, MA,USA, 2nd edition

[16].R. S. Pressman. *Software Engineering – A Practitioner's Approach.* McGraw Hill Education Asia, 2005

[17] http://en.wikipedia.org/wiki/Software_testing