A Logistic Regression Based Approach for Software Test Management

Yue Zhou
School of Information and Engineering
Communication Univercity of China
Beijing, China
Email: masonzhy@gmail.com

Jinyao Yan
Key Lab of Media Audio & Video, MOE
Communication Univercity of China
Beijing, China
Email: jyan@cuc.edu.cn

***Abstract*—Software test management is a hot field of software engineering both in academia and industry. However, traditional research are focused on software quality rather than test quality which could be evaluated by test management. Meanwhile, previous works on evaluating software test management were mostly based on statistical methods. In recently years, machine learning algorithms and techniques are developing rapidly. Logistic Regression is one of the important and often used algorithms. It is widely used in many domains such as credit investigation, book classification and so on. Therefore, our objective is to establish a logistic regression based approach for software test management to evaluate test quality. In this paper, we introduce our methodology to build metrics framework for test management, and enumerate the definition, type and range of each metric. We also show some results of our experiments using some data samples from a huge data sets.**

***Keywords*-test management; logistic regression; test quality;**

I. Introduction

Software testing is an important process in a software development project. It is usually a subset of all the stages as in the modern software development life cycle (SDLC) models, and the testing activities are mostly involved in all the stages of SDLC. In order to improve the quality of software products, it always costs a lot of resources, as [1] reported that "half the labor expended to develop a working program is typically spent on testing activities". However, testing cannot identify all the defects within the software, and it leads to big losses. According to a report of National Institute of Standards (NIST), the national annual costs of an inadequate infrastructure for software testing is estimated to range from $22.2 to $59.5 billion.

To improve software test quality and efficiency, a number of test management strategies have been proposed, such as [2], [3] and [4]. But previous works on evaluating test quality are most based on the traditional statistical methods. In the recent years, machine learning is a very powerful approach to data analysis, modelling and visualization. It is developing rapidly for applications in different fields, such as data mining, natural language processing, image recognition, and expert systems. Therefore, in this paper, our objective is to build metrics framework for software test management and to predict test quality based on logistic regression model.

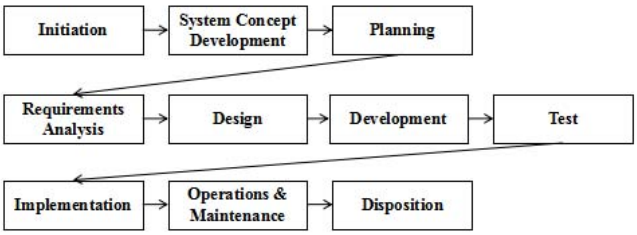

Figure 1. Model of SDLC

We classified metrics by involving total quality management theory and chose WEKA [5] as our experiment tool. We also present some exemplary results by using some data samples from a huge data sets.

The rest of this paper is organized as follows. In Section 2 we introduce our metrics framework for test management. We illustrate how to evaluate software test quality by using logistic regression model based on our metrics framework in Section 3. Finally, conclusions and future work are made in Section 4.

II. Metrics Framework for Test Management

In the field of software engineering, SDLC is a classic term to describe a process for planning, creating, testing, and deploying an information system. Fig 1 shows a typical stages of the SDLC which divided in ten steps from definition to creation and modification of the software development project. In this paper, requirements analysis, design and development are collectively called Development Stage (DS). And we subdivided Test Stage (TS) into requirements testing, unit testing, integration testing, system testing and acceptance testing.

Total quality management (TQM) is a set of classic management theory. As shown in Fig 2, it represents a typical process stream made up for manpower, machines, materials, methods and environments [6]. By introducing TQM into software test management, we considered several classes and subclasses of metrics separately in DS and TS. We designed test management metrics framework from three aspects: people involved, project property and test process.

978-1-5090-5154-0/16 $31.00 © 2016 IEEE
DOI 10.1109/CyberC.2016.59

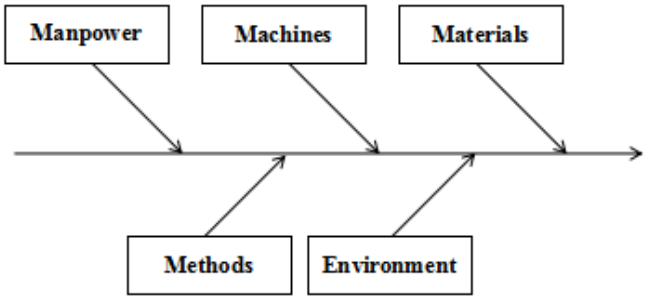

Figure 2. A typical process stream in fishbone diagram

We considered nineteen metrics and each metric was given the definition, type (discrete or continuous) and range. All the metrics we mentioned are shown in Table I.

A. People Involved

Testers and developers are the most important factor for software test management in an IT project. To a large extent, testers play the decisive role to software quality assurance. Developers have a very important effect on test management because they write codes. In general, researchers only concerned with the their skills and methods. However, we believed that many aspects of people have a significant impact on software quality. We separated metrics into two class: basic info and work ability.

- Basic Info
  It includes number of people, average age, total number of children and so forth.
- Work Ability
  It includes average work experience, average job performance, average skills training time and so forth. Among them, work experience measured by length of service for software testing; job performance measured by level of year-end assessment; skills training time measured by duration of test skills training.

B. Project Property

Test project is the object of software test execution and management which contained many significant important factors of software test management. We separated metrics into two class: project scale and requirements.

- Project Scale
  It could be measured by several ways such as Function Points (FP) or Source Lines of Code (SLOC) which also known as Lines of Code (LOC). In this paper, We recommend the IFPUG's (International Function Point Users Group) functional size measurement method to do FP estimation. Other metrics to describe project scale are number of test cases, number of modules and degree of coupling. Degree of coupling could be measured by number of associated system (or modules).
- Requirements
  It includes number of requirements and certainty of requirements. Let's give the definition of certainty of requirements. Support a test project has R requirements. In which, number of functional requirements is F and number of non-functional requirements is N. Therefore, $R = F + N$. If requirements based on review, a measurement method is as Equation 1 :

$$C = \frac{M}{R} = \frac{M}{F+N} \tag{1}$$

  where C is the certainty of requirements, M is the requirements which all the reviewers do the same explanation. When the requirements more specific, C gets more closer to 1.

C. Test Process

Test Process is the core content in the field of software test management. There are numerous metrics to measure test quality such as test coverage, defect and test resource.

- Test Coverage
  It includes test coverage rate and test passing rate. Test coverage rate shows that how many test cases are already do, and test passing rate means number of test cases which get correct result. Both of them can be calculated based on code or requirements.
- Defect
  It includes Defect Density (DD), average Defect Severity Level, Defect Age (DA) and so on. Defect Density is defined as Equation 2 :

$$DD = \frac{NumberofDefects}{FunctionPoints} \tag{2}$$

  Commonly, Defect Severity Level could be divided into critical, major, minor, trivial. Average Defect age can be measured in terms of time as Equation 3 :

$$DA = DefectFixDate - DefectDetectionDate \tag{3}$$

- Test Resource
  It contains numerous metrics such as Test Environment Failure Rate and Test Automation Rate. Test Environment Failure Rate (TEFR) is defined as Equation 4 :

$$TEFR = \frac{TestEnvironmentDowntime}{TestingTime} \tag{4}$$

  Test Automation Rate (TAR) can be calculated by Equation 5 :

$$TAR = \frac{NumberofAutomatedTestCases}{TotalNumberofTestCases} \tag{5}$$

Table I
METRICS FRAMEWORK FOR TEST MANAGEMENT

Class	Subclass	Metric	Type	Range
People Involved	Basic Info	Number of People	continuous	[0, ∞)
		Average Age	continuous	[0, ∞)
		Total Number of Children	discrete	positive integer
	Work Ability	Average Work Experience	continuous	[0, ∞)
		Average Job Performance	continuous	[0,100] (100 is full marks)
		Average Skills Training Time	continuous	[0, ∞)
Project Property	Project Scale	Function Points	continuous	[0, ∞)
		Number of Test Cases	discrete	positive integer
		Number of Modules	discrete	positive integer
		Degree of Coupling	discrete	positive integer
	Requirements	Number of Requirements	discrete	positive integer
		Certainty of Requirements	continuous	[0,1]
Test Process	Test Coverage	Test Coverage Rate	continuous	[0,1]
		Test Passing Rate	continuous	[0,1]
	Defect	Defect Density	continuous	[0,1]
		Average Defect Severity Level	continuous	[1,4] (trivial, minor, major, critical)
		Average Defect Age	continuous	[0, ∞]
	Test Resource	Test Environment Failure Rate	continuous	[0,1]
		Test Automation Rate	continuous	[0,1]

III. THE APPLICATION OF LOGISTIC REGRESSION

The principle of Logistic Regression (LR) rests on the analysis of a problem, in which a result measured with dichotomous variables such as 0 and 1 or true and false, is determined from one or more independent factors [7]. In the case of test management, the goal of LR would be finding the best fitting model to describe the relationship between test quality of software project and test management data collected during test process.

In this section, we illustrate the use of LR model to estimate test quality using data sets from a commercial bank in China. The bank has more than 200 information systems and the frequency of updating systems is once a month. By rough count, each year more than 2000 projects need to be tested. For commercial reasons, we choose some data samples that contain 52 instances to show the experiment results. Each instance contained 19 attributes the same as the metrics which mentioned in the last section.

According to the number of problems when the software is online, we defined a new attribute named Test Quality which would be predicted. If the number of problems less than 3 after it went into operation, Test Quality is defined as Good; on the contrary, it is defined as Bad. We selected a software "workbench" called WEKA as our experimental tool, which has incorporated several standard machine learning techniques into it.

Fig 3 shows the data visualization of distribution in the experiment. The summary of stratified cross-validation and confusion matrix are shown as Table II and Table III. The number of correctly classified instances is 45, and the correctly classified rate is 86.5385%. Before using this LR approach, this bank estimated test quality by reporting project risks manually. However, the accurate degree of test quality evaluation has greatly improved currently.

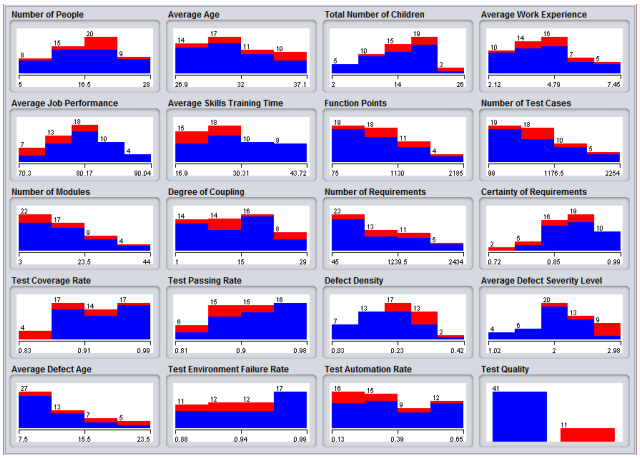

Figure 3. Data Visualization of Distribution

Table II
SUMMARY OF STRATIFIED CROSS-VALIDATION OF LR

Correctly Classified Instances	45 (86.5385%)
Incorrectly Classified Instances	7 (13.4615%)
Kappa statistic	0.6331
Mean absolute error	0.1339
Root mean squared error	0.3648
Relative absolute error	39.2813%
Root relative squared error	89.1574%

Table III
CONFUSION MATRIX OF LR

a	b		← classified as
36	5	|	a = Good
2	9	|	b = Bad

IV. SUMMARY AND CONCLUSIONS

In this paper, we took advantage of TQM and illustrated our metrics framework for test management including 19 metrics which could be measured during test process. After that, we introduced logistic regression model and made use of commercial data sets to inspect it. Actually, the focus of test management in an enterprise is not identical, so our metrics framework is flexible. For example, in a finance company, a new metric called Refer to Accounting (RTA) could be added which means a test project involving accounting or not. In the future, we hope to summarize more metrics of test management, and to compare effectiveness of different machine learning algorithms.

ACKNOWLEDGMENT

This work are supported by the Projects of NSFC (61371191, 61631016), and Research Project of China SARFT (2015-53).

REFERENCES

[1] E. Dustin, T. Garrett, and B. Gauf, *Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality*, 1st ed. Addison-Wesley Professional, 2009.

[2] K. D. Miller and E. W. K. Tsang, "Testing management theories: critical realist philosophy and research methods," *Strategic Management Journal*, vol. 32, no. 2, pp. 139–158, 2 2011.

[3] A. Lodhi, S. Wind, and K. Turowski, "Test management framework for managing it projects in industry," in *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, Sept 2013, pp. 509–514.

[4] S. Kukreja, A. Singhal, and A. Bansal, "A critical survey on test management in it projects," in *Computing, Communication Automation (ICCCA), 2015 International Conference on*, May 2015, pp. 791–796.

[5] G. Holmes, A. Donkin, and I. Witten, "Weka: A machine learning workbench," in *Proc Second Australia and New Zealand Conference on Intelligent Information Systems*, Brisbane, Australia, 1994.

[6] J. P. Pekar, *Total Quality Management: Guiding Principles for Application*. Astm Intl, 1995.

[7] D. W. Hosmer and S. Lemeshow, *Applied logistic regression*. Wiley, 2000.