ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 9(45), DOI: 10.17485/ijst/2016/v9i45/101581, December 2016*

A Fault Prediction Approach based on the Probabilistic Model for Improvising Software Inspection

B. Dhanalaxmi[1]*, G. Apparao Naidu[2] and K. Anuradha[3]

[1]Department of Information Technology, Institute of Aeronautical Engineering, Dundigal, Hyderabad –500 043, Telangana, India; dinnu18@gmail.com.
[2]Department of Computer Science and Engineering, J. B. Institute of Engineering and Technology, Bhaskar Nagar, Moinabad Mandal, R. R. District, Hyderabad – 500075, Telangana, India; apparaonaidug@gmail.com
[3]Department of Computer Science and Engineering, Gokaraju Rangaraju Institute of Engineering and Technology, Nizampet Road, Bachupally, Kukatpally, Hyderabad – 500090, Telangana, India; kodali.anuradha@yahoo.com

Abstract

Objective: Software development is a multitask activity performed by a team. Each activity involves with different tasks and complexity. To achieve quality of improvement it is important that each activity task should be fault free. But, finding and correcting faults are most expensive and time consuming. **Methods**: Software inspection is a static analysis technique which does not required program execution, instead it use inspector to make decision during the development. **Findings**: But it was observed in literature that inspection has bad records in finding accurate defects in software development. In this paper, we present a novel Fault Prediction Approach (FPA) based on the probabilistic model to improvise the software inspection to detect the defect accurately and cost effective for the quality software development. **Application/ Improvement:** Inspection is an effective activity to find the defects using empirical data in the initial stage of development. The proposed FPA investigate a probabilistic methods using modified Naive Bayes classification to estimate the probable faults in an experiment context to suggest fault controlling development. Further, the analysis investigates how FPA effectively identifying the faults during the inspection and impact in the quality development performance..

Keywords: Fault Prediction, Probabilistic Model, Software Inspection, Software Quality

1. Introduction

Software Engineering process organized the software development methodologies, disciplined and quantifiable measures for systematic development, which requires a lot of human efforts. Exceptional defects that deviate from the quality characteristics of the software development which might requires human-based activities for the inevitable defects and it should be automatically predicted, monitored and resolved. Quality assurance software with just a test is not enough because it has the effect of delay in development, and is quite expensive. Software inspection immediately after the development of the product can be used early in the software development life cycle, as well as to determine the quality of the product at the same time to save the project and later worked to find gaps in

*Author for correspondence

the development of software products, has established an impressive track record.

The purpose of evaluating the quality of a product is to invest in quality assurance from an economic point of view to discover as many product defects as possible and to optimize the associated costs and benefits[1,2]. Inspection is a wide range of practical software verification and validation of software used for the design and application of static analysis techniques[3–5]. In this technique the human inspector reads and checks the program to reveal shortcomings and faults. Since the inspection does not require the launch of the program, but makes use of human judgment instead, it can be applied at any time before or after the code is completed.

Many researchers have contributed to the development of the methodology of inspection as defined different reading techniques[6–8] and inspection procedures[9], but many of the techniques focus on the administrative aspects, such as, meetings and management[10–12] but it can be difficult to support with software tools, except for lowing or syntactic level[13]. There are some tools available which successfully provide automated inspection program to detect bugs, such as "Coverity static analysis Tool" by Engler groups[14,15] and the tool developed by[16], but these tools primarily seems to handle the implementation of related errors that can be defined without reference to the requirements of the software.

It would be difficult to achieve a fully automated process of inspection to find errors related requirements, but a high level of automated support for inspection based on the specifications is desirable because it will help increase efficiency and reduce human error in the controls. In general, the construction of a highly automated tool aid must be based on accurate descriptions of requirements specifications, checklists for inspections and control processes, but the lack of precision in the normal control procedures makes it difficult to achieve the objective.

This paper, examines the FPA based on a descriptive empirical study of probabilistic model testing and the relationship between test factors for effective detection of failure factors and nominal cost benefits, reading skills and test duration. The efficiency of the review process for various inspection periods is an important aspect to investigate because the inspection period is an important cost factor. A recent survey[17], suggest empirically measured using a probability distribution of the statistical probability of the model which has been deployed for the defects use for a nominal hypothesis inspection team to calculate the efficiency of identifying costs and benefits of technology scars. The purpose FPA will support the identification of a problem with the software development process and the FPA probabilistic model analysis the economic program and is to determine whether the organization will help to improve the quality of products. The study focus in the failure and its prevention through detect inspection in the usefulness of the detection as a solution.

The remaining papers are structured as follows, Section 2 presents related research works, Section 3 presents limitations of the inspection, FPA probability model inspection, and FPA prediction cost. Section 4 presents case study data and measurement methods. Section 5 describes the analysis of the results and Section 6 describes the conclusions and future work.

2. Related Works

An important focus of research is the efficiency and effectiveness of the defect detection process[6,14,17]. Reading techniques used in the steps of detecting the fault inspection showed that a significant effect on the individual fault detection efficiency and effectiveness.

There are several ways to identify defects such as inspection, testing the validity of the evidence. Formal audits are more effective and expensive quality[3] method for identifying defects in the early stages of development. Through prototypes several requirements are clearly understood what helps to overcome limitations. Testing is one of the least effective methods. Defects that can be avoided during the initial phase can be detected during testing. The accuracy of the evidence is a good tool to detect mainly in coding phase. Precision in the construction industry is the most efficient and economical way to build software.

Many organizations follow a three-step inspection to see some failures, lack of collection, failure and repair[13]. These process software inspections typically include management and editing steps for inspection planning and reporting, which contrasts with the detection and collec-

tion of operational failures by inspection teams. A recent review of research approaches to improve and support discovery failures and individual steps of an inspection team meeting to optimize inspection results [6,13]. The success of an inspection is defined as a function found in the total number of faults in a given class for a defect in the product. Studies show that there is a significant change in efficiency, effort, and duration of inspections associated with the approach used in [15]. Several studies based on the detection efficiency of the meeting to move the collection of defects. The technique for fault detection is designed to detect flaws in the inspector associated with meta-class errors.

In[16] to explain the importance of access to error prevention, noting that 15% of the project and it is necessary to reduce development costs and time to characterize the quality of the product of the fertile aspect of the software update. Failure removal efficiency and prevent errors in the orthogonal defect classification techniques are helpful. Through training and the use of known standards, organizations can move to better places and improve quality standards. The strength of this research is that you can compare different mechanisms and techniques to prevent development loss and helps in accurate analysis of the different levels of organization.

In[16], focuses on disability prevention and inspection techniques evaluated for five projects of various sizes. The study found that 70% of the attributes in the inspection and developer unit testing and 29% in the phase validation were detected. These metrics are useful in increasing confidence for the product development. Evaluation of the project yields calculated called the efficiency of the removal of the defect is calculated; the number of state tested the actual number of fault features identified.

According to[18] in meetings, we can hardly detect the impact or significantly extend the duration of your inspection. In all cases, individual work-intensive defect detection is an important prerequisite for a successful solution. So far, these techniques have had little or no effect on the various steps of the actual testing effort invested and empirical information on the effort required achieving the best results.

In[19], it highlighted the importance of preventive measures to control defects early in the project development. "Causal analysis", is a widely used process for identifying the cause of a feature and taking corrective action. However, when the incidents reported are increased, it is very difficult to handle and take measures to stop. Foretelling the case is very important to protect against the failure of software process improvement. The study represents the failure of the prediction based on the laws of the association, which applies to the mining association techniques. The proposed mechanism of action that may lead to higher levels of the software development process can be applied to predict the feature faults.

In[20] proposed a framework to protect tentative breakdown of starting "0.85 to 0.1 per Kilogram Line of Code (KLOC)". Designing and monitoring defined processes by means of data analysis techniques. An error in order to achieve the highest levels of quality control measures are performed at different stages of the project life cycle. Defect in order to achieve the highest levels of quality control measures are performed at various stages of the project life cycle. Works for the protection of the mistake are: preparation, definition of a failure of this type, the process of prevention and support for the protection of ignorance.

3. FPA

We use the term views here in the broad sense for all kinds of documents to read errors detection with FPA. A limitation of software inspection is discussed to understand the need of FPA initially and the Naive Bayes based probabilistic model is discussed below.

3.1 Limitation in Software Inspection

Static analysis technology that relies on visual inspection is to identify gaps in product development, the development is a classic example of violation of the standards and other issues [6,7]. Use most static analysis tools to eliminate exceptions that can be eliminated using static analysis, such as standard, "uncaught runtime", "redundant code", "using invalid variables", "division by zero", and "coding in possible memory leaks". An outstanding example is the test of an official[10] which has a well-defined process and prepare the meetings and different roles defined. Another

process that is often used is the Walkthrough. In this process, the president leads through the code, but there is no need to prepare.

Inspection software is used to successfully detect defects in various types of documents, such as specifications, designs, test programs, and source code [14]. Several reviewers independently examined the same document. All defects mentioned in the inspection are then collected. More than one reviewer cannot detect some defects therefore; the inspection result was zero-one matrix that shows where the reviewer found which fault.

Not all defects found in this document are often found during testing. After the inspection, the administrator must decide whether to correct the errors in the document or to forward the document to the next stage of development. The normal way to determine the degree of defect-free documents. For example, management may require the document to be above 90 percent defect-free before being utilised for the development. In fact, many defects are actually included in the document. Therefore, this is a major problem in software engineering practices.

Current techniques standard assessment test after the failure content falls into two categories: clutch find ways[2,19] and curve-fitting [17]. The standard method uses the "Buddy Zero One matrix", as the only way to calculate estimates. Some studies show that "capture-capture-capture statistics" and "curve-fittings" are too inadequate to be practical [12]. Methods reflect the excesses and make a huge difference and fault interpretation likely indicates that these methods did not consider the standard made during the last inspection.

We describe the use of probabilistic analysis model for predicting failure. In the future, the concept of inspection might be suggested for the guided control. FPA can enable software engineers to correct errors before they are on the surface of the more public inspection or test failures customer reports.

3.2 FPA Probabilistic Model

The proposed FPA model can be used in software development as a filter to find faults in the process. It can automate the FPA to identify specific types of exceptions and to scan and parse the source text of the program to find a sample set of code. FPA analysis utilized for "control flow", "data flow analysis", "interface analysis", "information analysis" and "time analysis software".

There are number of errors in the software development which can automatically detect the FPA. At the same time, failure prediction tools indicate that each tool is different, and sometimes it does not overlap, or that bug is found [2]. The anomalies found are not always due to actual faults, but they are often an indication of faults.

Critical issues related to the use of the fault prediction tool cannot be ignored by a number of false reports or errors that do not include a deeper analysis of the context in the system. There may be more than 50 percentage false positives for all actual bugs. Some static analysis tools report a low 50% error [15,16]. Often, static analysis tools are filters that can be customized and set out some of the errors will be reported, and reduce the number of false positives. Other organizations virus services pre-screening to eliminate false positives analysis of the magnetic field exit before the involvement of their own groups.

We classify defects based on the probability, which can be detected by using the FPA established fault type taxonomy. Taxonomy consists of trained data failure knowledge related to the kind of level applications that can be identified by the FPA. To build the taxonomy we integrate the five defects classes identified by[22] as, "Omission", "Incorrect Fact", "Inconsistency", "Ambiguity", and "Extraneous Information". Alogrithm-1 describes the probable model to identify the possible defects using the taxonomy knowledge base.

Alogorithm-1: ***Fault Prediction***

Input:
$C[\,] \rightarrow$ Set of Test Code
Faults_TaxonomySet[ ][ ] → *n-dimensional vector*

For each block of code in c_i of $C\,[\,]$
For of each fault taxonomy data t_i of *Faults_TaxonomySet* $[c_i\,][\,]$
Calculate the Naive Bayes probabable similarity β in *Faults_TaxonomySet* $[c_i\,][\,]$
End for
End for
If β >=1 **then**
c_i, *classified fault as* → t_i
End if
End for

Probable model determines the classification system malfunction with the intention to identify the categories of different faults. To perform FPA Probabilistic Model inspection we measure the performance metrics as, the number of defects detected by the inspection, the number of errors found by the FPA, the preparation and meeting time, and the pre-screening cost. The probabilistic computation case study is discussed in section-4.

In addition, keep in mind that many authors offer guidance in order to achieve the full extent of the testing, i.e., how fast testers to read the documents. This is evident impact on the effectiveness of the test. For example, [8] the evaluation of a quality inspection is between page 1 to 0.8 per hour, when the document contains 300 words. If some authors to provide the same value assassination, then we can summarize this easily with a statement that the inspection is about one page per hour. However, the result of a large deviation from fully understood. So, we may review our view point in the faults found.

4. A Case Study

To determine the cost of finding fault with the test, we manually examined inspection records different version releases V1 and V2, a total of approximately 45 thousands Lines Of Code (LOC) being used for a design, development & implementation of Human Resources Management System (HRMS) as shown in Figure 1.

4.1 Analysis Data

We collect and analyze fault data for the HRMS system. Data analysis performed by 10 inspectors and testers to provide customer errors reports, for more than 45,000 LOC written in Java, JavaScript and SQL developed for an business data processing company. As it will be explained that each of these projects is subject to FPA or other combinations of inspections and tests. FPA or inspections cannot be conducted, and the FPA to be done before the inspection, before the test, or during the test.

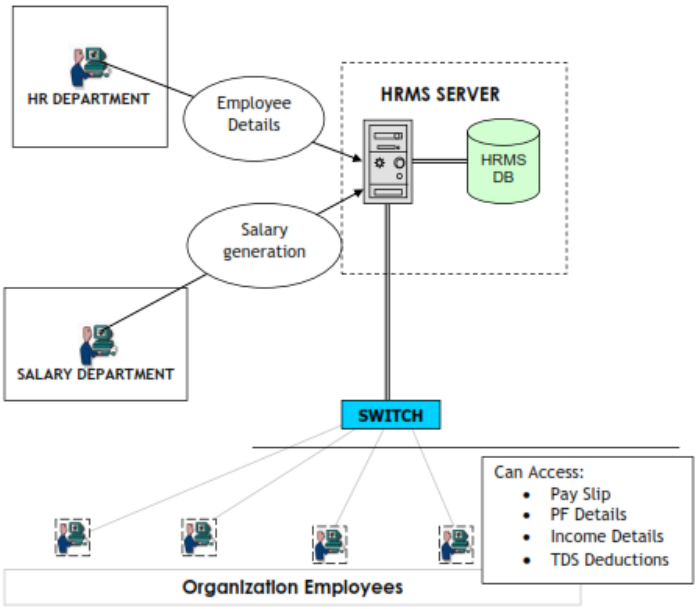

Figure 1. HRMS system model.

Table 1. Inspection vs. FPA data analyzed

System Versions	Inspection	FPA	Change Request (CR)	% of CR
A	Performed	No	Yes	71%
B	Performed	Yes	Yes	24%

The first two versions of the system were analyzed we term as System-A and System B, and both underwent inspection and FPA. However, the inspections have not been conducted on the System-A. We analyzed the first version (V1) which is underwent inspection only because it was developed before. The next release (V2) passed both the FPA and inspection.

For System-A and System-B, FPA errors are sent to the service of pre-screening. Errors that were analyzed in this study are good and true that remained after the pre-test. To each issue, we are carefully classified and FPA multitude of failures, inspection documentations, and Change Requests (CR) record. Each test failure was reported by a CR to the customer. The Table-1 shows data analyzed summary for each product.

4.2 Prediction Cost Measures

Inspection records for System-A and System-B, contains quantifications time to prepare for the time of each inspection participants and profile errors, including the type, complexity and explanation of the test set by the convention. To get the measure of the failure to find, we have added a correction and meeting all the participants and the number of errors is divided by time that are detected during the inspection, as given in equation-1. Here, n defines the number of participating inspector. We computed the costs by the annual average cost of wages 2.5 Lakhs per inspection participants.

$$Avg.Cost\ of\ Fault\ Detection(Inspection) = \frac{\sum_{i=1}^{n}(Minute\ Time\ Meeting + Time\ Preparing)_n\left(\frac{Salary}{Time}\right)}{Quantitiy\ Fault\ Found} \quad (1)$$

We computer average price of FPA for fault detection based on the cost of the certificates, the price of prescreening to eliminate false positives based on the LOC and a good number of real errors may occur. Some additional costs can be difficult to cover the cost of learning how to run using FPA and computing resources. The lack of information is a limitation of knowledge, but I do not think it will be evident in our results.

To use FPA and to learn how to use computing resources to run may be difficult to cover the some additional cost. The lack of information is the limit of our knowledge, but we do not believe and the results can be an evident. To purchase no additional computing resources are required to use the FPA. The cost of computer detection errors shown in the equation- (2).

$$Avg.Cost\ of\ Fault\ Detection(FPA) = \frac{FPA\ Certificate + (Cost\ Per\ Line)(LOC)}{Quantitiy\ TP\ Fault\ Found} \quad (2)$$

To measure the value we provide cost benefit, as shown equation- (3).

$$Cost\ Benifit = \frac{Avg.\ Cost\ of\ fault\ Detection\ by\ FPA}{Avg.Cost\ of\ fault\ Detection\ by\ Inspection} \quad (3)$$

Based on our data, the calculated cost benefits ratios are 0.86 for the V1 and V2 to 0.53, which means that the cost of the FPA by the detected fault is less in compare to inspection detection. These results show that the FPA is relatively affordable techniques for debugging.

5. Result Analysis

The empirical results are analyzed using a number of features found in the inspection system, the amount of features available to customers testing, it cause Thousands of Lines of Code (KLOC). Table 2 provides a comparison of the quality of the final product. This measurement applies to the final quality of the product; the number of errors after KLOC has been stirred (both tested and failed customers reported).

The KLOC-specific defects are the quality standards of the final product because they reflect the effects of prod-

Table 2. Relative quality analysis

	Process Cycle-1	Process Cycle-2	Relative Quality (Failure/KLOC)
System-A	Inspection		1.84
System-B	Inspection	FPA	0.35

Table 3. Percentage of defect types found

Defect Type	Customer (%)	Inspection (%)	FPA (%)	Test (%)
Assignment	0	6.81	84.28	4.12
Interfaces	0	1.68	0	1.09
Functions	72.52	2.05	0	62.81
Validations	0	25.82	38.82	1.25
Algorithms	41.31	39.10	0	45.04
Documentation	0	42.81	0	0

uct changes. Using System-A as main product, because this version was originally developed without FPA. We normalize fault omissions by KLOC metric relative to that System-A to protect the quality of information. This gives quality ratio of the Relative Quality comparison of both the versions, as shown in Table 2.

As mentioned earlier, there is a significant difference in the relative quality of the system. As a result, our analysis indicates better production of better quality products.

Comparison between different types of defects is shown in Table 3. The results show that the FPA largely able to identify errors in the three types of errors in Assignments, Validations and Functions. A larger range of errors have been identified in the validation algorithm and documentation can be seen all these activities are done in prior of FPA and even fewer validation faults are identified by inspection and testing compared to the FPA.

6. Conclusion

A key feature of the inspection design is the size of the inspection team and a set of technologies that use the team to detect errors. To schedule the inspection, the leader seeks to determine the possible effort, efficiency, and cost-effectiveness of a particular test design and to choose the design that best suits his or her project goals. To analyze the value of FPA (predictive error approach), we analyzed automated inspection errors, manual inspection errors, and change request (CR) data for both development HRMS products. Our analysis provides some results that can help you understand and use FPA according to its limitations. The FPA expenses by fault was found out in the same order of importance as the cost of assessing the errors found by errors discovered, indicating that the FPA is relatively affordable techniques for debugging. The number of FPA errors in the module can be a pretty good indicator of identification module failure-prone before the test. In conclusion, our results show that FPA is a cost-effective adds advantage to other techniques of verification and validation.

7. References

1. Shivaji ES, Whitehead J, Akella R, Kim S. Reducing features to improve code change-based bug prediction. IEEE Transactions on Software Engineering. 2013Apr; 39(4):552–69.
2. Cinque M, Cotroneo D, Pecchia A. Event logs for the analysis of software failures: a rule-based approach. IEEE Transactions on Software Engineering. 2013 Jun; 39(6):806–21.
3. Parnas DL, Lawford M. The role of inspection in software quality assurance. IEEE Transactions on Software Engineering. 2003 Aug; 29(8):674–6.
4. Porter A, Votta L, Basili V. Comparing detection methods for software requirements inspections: a replicated experiment. IEEE Transactions on Software Engineering. 1995 Jun; 21(6):563–75.
5. Zheng J, Williams L, Nagappan N, Snipes W, Hudepohl JP, Vouk MA. On the value of static analysis for fault detection in software. IEEE Transactions on Software Engineering. 2006Apr; 32(4):1–14.
6. Liu S, Chen Y, Nagoya F , McDermid JA. Formal specification-based inspection for verification of programs. IEEE Transactions on software engineering. 2012 Sep–Oct; 38(5):1100–22.
7. Padberg F, Ragg T, Schoknecht R. Using machine learning for estimating the defect content after an inspection. IEEE Transactions on Software Engineering. 2004 Jan; 30(1):17–28.
8. Ackerman A, Buchwald L, Lewski F. Software inspections: an effective verification process. IEEE Software. 1989; 6(3):31–36.
9. Laitenberger O, DeBaud JM. An encompassing life cycle centric survey of software inspection. Journal of Systems and Software. 2000 Jan; 30(1):5–31.
10. Fagan ME. Design and code inspections to reduce errors in program development. IBM Systems Journal. 1976; 15(3):182–211.
11. Briand LC, Emam KE, Freimut BG, Laitenberger O. A comprehensive evaluation of capture-recapture models for estimating software defect content. IEEE Transactions on Software Engineering. 2000 Jun; 26(6):518–40.
12. Hamill M, Popstojanova KG. Common trends in software fault and failure data. IEEE Transactions on Software Engineering. 2009 Jul–Aug; 35(4):484–96.
13. Bush W, Pincus J, Sielaff D. A static analyzer for finding dynamic programming errors. Journal of Software: Practice and Experience. 2000 Jun; 30(7):775–802.
14. Suma V, Nair TRG. Effective defect prevention approach in software process for achieving better quality levels. World Academy of Science, Engineering and Technology. 2008 Aug; 45:1–5.

15. Challagulla V, Bastani F, Yen I, Paul R. Empirical assessment of machine learning based software defect prediction techniques. Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems; TA; 2005. p. 263–70.
16. Johnson PM, Tjahjono D. Assessing software review meetings: a controlled experimental study using CSRS[Internet]. 1996 [cited 1996 Jun]. Available from: csdl.ics.hawaii.edu.
17. Chang CP, Chu CP. Defect prevention in software processes: An action based approach. The Journal of Systems and Software. 2007; 80(4):559–70.
18. Hovemeyer D, Pugh W. Finding bugs is easy. Proceedings of Conference in Object Oriented Programming Systems Languages and Applications (OOSPLA) Companion; 2004. p. 132–5.
19. Basili VR, Green S, Laitenberger O, Lanubile F, Shull F, Soerumgaard S, Zelkowitz M. The empirical investigation of perspective-based reading. Empirical Software Engineering journal. 1996; 1(2):133–64.
20. Sathyaraj R, Prabu S. An approach for software fault prediction to measure the quality of different prediction methodologies using software metrics. Indian Journal of Science and Technology. 2015 Dec; 8(35). Doi no: 10.17485/ijst/2015/v8i35/73717.
21. Paramasivan R, Santhi K. APPFPA based best compromised solution for dynamic economic emission dispatch. Indian Journal of Science and Technology. 2016 Feb; 9(6). Doi no:10.17485/ijst/2016/v9i6/81015.
22. Kang SJ, Lee KM, Lee KM. Context-aware abnormality monitoring service for care-needing persons using a probabilistic model. Indian Journal of Science and Technology. 2016 Jun; 9(24). Doi no: 10.17485/ijst/2016/v9i24/96112.
23. Kim YY, Kim MH. What are software developers and medical expert's priorities for adopting a healthcare software platform? Indian Journal of Science and Technology. 2016 Jun; 9(24). Doi no:10.17485/ijst/2016/v9i24/96018.
24. Ramasamy S, Kumaran AMJM. Dynamically weighted combination of fault - based software reliability growth models. Indian Journal of Science and Technology. 2016 Jun; 9(22). Doi no:10.17485/ijst/2016/v9i22/93967.
25. Samuel S, Kovalan A. A design level optimization approach for functional paradigm software designs considering low resource devices development. Indian Journal of Science and Technology. 2016 Jun; 9(21). Doi no:10.17485/ijst/2016/v9i21/95208.