Analytical Phase wise Analysis of Defect Removal Effectiveness to Enhancing the Software Quality

Arun Kumar Marandi¹, Danish Ali Khan²
¹arunmarandi.ca@nitjsr.ac.in, ²dakhan.ca@nitjsr.ac.in
¹,²Department of Computer Applications, National Institute of Technology, Jamshedpur – 831014 (INDIA)

Abstract: In this paper, applications of the Defect Removal Effectiveness method for finding the solution of Phase wise Analytical Analysis are present. Phase-based Defects Removal, method is novel technique to analysis and summarizes the relationships among three metrics defect injection, defect removal and effectiveness. The paper deals proposed method an iterative connection is formulated, which allows determining the Phase wise Defects Removal effectiveness (PDRE). In this methodology little modify the values of the defect rate parameter are improve the effectiveness of the product and enhancing the Software Quality.

Keywords: software quality, software defect analysis, software inspection, defect tracking, phase-based defects removal.

1. Introduction

A Long running project meets successful compilation to reducing the Defects during software Development life cycle. In software development, lot of defects would come out during the development process [1]. Software is a unique entity that has laid a strong impact on all other fields [2]. Now the present day the organization faces with greater competitive pressures and skyrocketing costs of software breakdown have pushed the need for testing to new heights. Defect removal effectiveness is a direct indicator of the capability of a software development process in removing defects before the software is delivered. It is one of few, perhaps the only, process indicators that bear a direct correlation with the quality of the software's field performance [3][4]. The paper deals several aspects of defect removal effectiveness including overall effectiveness, inspection effectiveness, test effectiveness, phase specific effectiveness and the role of defect removal effectiveness under quality planning [5][6]. High quality software attributes to a defect-free product, which is competent of producing predictable results and remains deliverable within time and cost constraints [4]. It should be manageable with minimum interferences. It should also be maintainable, dependable, understandable and efficient to increased competitiveness in today’s business world, Thus, a systematic approach towards high quality software development [1] is required due technological advances hardware complexity and frequently changing business requirements [5][7][8]. Software Engineering is a discipline that aims at producing high quality software through systematic, well-disciplined approach of software development. It involves methods, tools best practices and standards to achieve its objective [4]. The three main phases of software development life cycle (SDLC) are requirement analysis, design and coding phases. The defect introduced into the code and to reduce the overall number of delivered defects per KLOC by 85 percentages. The Proposed Framework illustrates the method used to eliminate phase-by-phase defects removal. To deploy high quality software, it is essential to develop a defect-free deliverable at each phase. A defect is any blemish, imperfection, or undesired behavior that occurs either in the deliverable or in the product. Anything related to defect removal is a continual process and to removing defects, process and technique are used to improve the software quality. Defect analysis at early stages of software development reduces the time, cost and resources required for rework. Early defect detection prevents defect migration
from requirement phase to design and from design phase into implementation phase [3][6][9]. There are several studies conducted by different researchers for producing reliable software through error removal in code lines and software testing. But there are only few researchers who have given time in defect detection and removal in the development phase for delivering the reliable requirement specification [1]. Few authors have given four way of detect defects a) Checklist Based Detection b) Scenario Based Detection c) Perspective Based Detection d) Traceability Based Detection by [6], some author depend upon Defect Density Model and Design Phase Analysis for defect detection [7][9].

The rest of the paper is as follows: section 2 describes about the proposed framework, follows by the result and discussion in section 3.

2. Proposed Framework

Our paper aims to understand the impact of inspection and review on the defect removal effectiveness analysis and enhancing the Software Quality [10]. The paper first presents a simplified model of defect removal, and develops the details of the model at the level of individual defect removal at each step in software developing phase. In this section we, describe about the proposed model. In our proposed model there are three basic stages such as defect injection, defect detection and defect removal during software development life cycle. The model gets the Defect Removal Efficiency [5], competitively analysis, and reducing cost; improve on software quality and time effectively in Iterative Phase Defect Removal Model.

![Fig.1: Iterative Phase Defect Removal Model](image)

2.1. Defect Injection during SDLC

An empirical study conducted across several project from various service-based and product-based organizations reveals that requirement phase contain 50% to 60% of total defects. 15% to 30% of defects are at design phase. Implementation phase contains 10% to 20% of defects [1][2]. Remaining are miscellaneous defects that occurs because of bad fixes. Bad fixes are injection of secondary defects due to bad repair of defects. The common cause for defect occurrences at requirements phase are requirement incompleteness, inconsistency, ambiguity, requirement change and requirement presentation. The common reasons for defect occurrences at design phase are non-conformance to external specification, internal specifications, logical specifications, interface specifications, component specification, security, organizational policies and standards in addition to non-conformance to design with requirement specification. The common sources for defect occurrences at implementation phase are improper error handling, improper algorithm, programming language short comings and wrong data access and novice developers [2]. The paper discusses weather the errors are mostly introduced or root cause during development process. In this section error are injected in different phase, the software engineers or tester are resolve this problem into the testing phase [3]. Defect occurs all the way through the development process. Hence, defect prevention becomes an essential part of software process quality improvement. In this study, in order to improve software process quality, defects are first identified from a given set of projects where the defects are mostly injected into the software development process. In this below figure-2 (a) describe the phase wise defect analysis with respect to cost
and another figure shows that percentage of occurs into the phase wise development. The phase wise defect injection analysis with cost, the first stage of SDLC cost should be low but the defect injection little bit higher than the cost level.

Show that initially defects are injected into little bit higher, the another Figure-2 (b) shows that initial percentage defects are higher. In the both figure 2 (a) and (b) analyze defects injected during SDLC process.

### 2.2. Defect Detection Process

In this stage detection, process did phase wise detection through detection metrics, finding through inspection the core document, inspection the code, review and defect analysis. The process starts with a requirement and from this point onwards, there is gradually more and more effort being applied to the creation or modification of an application. More people become involved and the cost of the solution increases. This is the natural course of events. It is therefore a natural conclusion, that if a mistake is made in the requirement, the cheapest time to fix the problem is as the requirement is being defined, or immediately afterwards resolve defect prevention and reduced defects from the requirement phase. In the above figure 1 (a) phase-wise defect injection analysis with cost it clear that low level phase has less amount spent during the software development life cycle. The paper deals detection method Defects are introduced throughout the software development life cycle and the art of testing is to find as many of them as possible when they are inserted. It is widely recognized that there is a parabolic curve of defect insertion. The starting point is the requirement specification, which begins by inserting 60% of the defects. The curve terminates with live or production, where the intended result is to find 0 defects. The tester should report on completion of the project, the defect detection efficiency. This looks at understanding for each defect, where it was inserted and where it was detected. A perfect test process would look to identify each defect as it is inserted. This is highly unlikely and the reality is that some defects will be found in later phases of testing. It is therefore important that testing is involved in the project from the outset, not as something that is included if there is the time, the budget and the inclination. Only by the application of systematic testing throughout the project is the quality level going to be understood and the opportunity to remedy problems in a timely manner presented. On the observation of figure 2-(a), (b) and analysis of defect finding table-1.

<table>
<thead>
<tr>
<th>Table-1: Historical Project Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECTS/Modules</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>p1</td>
</tr>
<tr>
<td>p2</td>
</tr>
<tr>
<td>p3</td>
</tr>
<tr>
<td>p4</td>
</tr>
<tr>
<td>p5</td>
</tr>
<tr>
<td>p6</td>
</tr>
<tr>
<td>p7</td>
</tr>
<tr>
<td>p8</td>
</tr>
<tr>
<td>p9</td>
</tr>
<tr>
<td>p10</td>
</tr>
<tr>
<td>p11</td>
</tr>
<tr>
<td>p12</td>
</tr>
<tr>
<td>p13</td>
</tr>
<tr>
<td>p14</td>
</tr>
<tr>
<td>p15</td>
</tr>
<tr>
<td>p16</td>
</tr>
<tr>
<td>p17</td>
</tr>
<tr>
<td>p18</td>
</tr>
<tr>
<td>p19</td>
</tr>
<tr>
<td>p20</td>
</tr>
<tr>
<td>p21</td>
</tr>
<tr>
<td>p22</td>
</tr>
<tr>
<td>p23</td>
</tr>
<tr>
<td>p24</td>
</tr>
<tr>
<td>p25</td>
</tr>
<tr>
<td>p26</td>
</tr>
<tr>
<td>p27</td>
</tr>
<tr>
<td>p28</td>
</tr>
</tbody>
</table>
2.3. Defect Removal Process

Defect, as defined by software developers, are variances from a desired attributes. These attributes include complete and correct requirements and specifications as drawn from the desires of potential customers. Thus, defect cause software to fail to meet requirement and make customers unsatisfactory. First, understand the activities in the development process that are related to defect injection and to removals. Defects are injected into the product or intermediate deliverables of the product at the various stages of SDLC. Defect Removal is used to determine the effectiveness of testing term’s defect removal efforts [11]. It represents as an indirect indication to the quality of the product. Higher the percentage of defect removal effectiveness, the more positive impact on product quality. This is because it represents the ability of testing team to find product defects prior to product delivery. The more possible value of defect removal effectiveness is 1 % or 100%, through it is quite unlikely to achieve. It can also be measured during various stages of software development life cycle. In such a way that a lower report analysis of design phases for instance, may signify that the technical review process needs to be strengthened. Manipulation of DRE post release of a product and analyzing the root causes behind various defects reported by end customers may help strengthen the testing process in order to achieve higher DRE levels for future releases. Defect removal is equal to defects detected minus incorrect repairs. Show that the paper deals iterative defect removal model and analysis its results using different distribution method. The model divided into five stages. First is SDLC phases are introduced to defect origin and where the defect location are traceable more. Defects are mostly introducing at requirements, design, code and testing phases. In each phases the defect location are critically unidentified. Second stage is Defect or error injection during the developing process. In this stage, identify the root cause of defect injection at different levels of software development cycle. There are some Metrics to be used and implement to enhancing the quality of the products. Third stage is Defect detection, in this stage detected the defects and counts the error percentage. Fourth stage is Defect Removal Process, in this stage the model verify and analysis the defect or unidentified defect (Bad fixing) go to re-refine the third stage. If the entire defects are reduced it redirect to the next SDLC phases and iteratively. The paper deals a simple model of the defect removal process to develop formulas for enhancing the quality and reducing the work effort.

Notation,

\[ P_k = \text{SDLC Phase to step } k. \]
\[ DI_k = \text{the number of defects injected in the phases on entry to step } k. \]
\[ DD_k = \text{the number of defects detection during the phase development to step } k. \]
\[ DR_k = \text{the count of defect removed from each step } k. \]
\[ Q_k = \text{Number of defect in the released software.} \]

The defect removal efficiency for each software development life cycle, relative to the defects present is the fraction found \( P_k \) ‘times the fraction of good fixes \((1 - e_k)\). Hence the defect removal efficiency (DRE\(_{ck}\)) relative to the defects currently into the each step \( k \) is.

\[
\text{DRE}_{ck} = (1 - e_k) P_k
\]

Let us assume that \( e \) and \( p \) are constants:

\[
\text{DRE}_c = (1 - e) P
\]  

(1)
The fraction of defects removed relative to the total lifetime defects (DRE) is:

\[ \text{DRE}_c = (1 - \varepsilon).p \]  \hspace{1cm} (1')

The fraction of defects removed in the review/inspection for each step \( k \) [1].

\[ \text{DRE}_k = \frac{\text{MP}}{\text{TD}} \]  \hspace{1cm} (2)

Where Phase 1 effectiveness (\( E_1 \)) and Phase 2 effectiveness (\( E_2 \)) [1],

\[ E_1 = E_2 \]  \hspace{1cm} (3)

(from Remus and Zilles “mathematical relationships of defect removal effectiveness)

From the equation (2) and equation (3), the result can be obtained directly.

\[ \text{DRE}_1 = \frac{p_{1,0D}}{\left( \frac{1}{1 - c_1} \right)0D} \]  \hspace{1cm} (4)

\[ = (1 - \varepsilon_k).P_k \] (Let ‘OD’ is the original defects introduced into the SDLC Phases.)

\[ \text{TD} = \text{MP} + (E_2 \times (\text{TD} - \text{MP})) \]

\[ Q_k = \text{TD} (1 - \text{DRE}_k) n \]

or:

\[ Q_2 = \text{TD} (1 - \text{DRE}_k) 2 \]  \hspace{1cm} (5)

Where,

\[ Q = \text{TD} (1 - E) (1 - E_2) = \text{TD} (1 - E) 2 \]

\[ = (\text{TD} - \text{MP}) E_2 \times (\text{TD} - \text{MP}) \times \frac{1}{\text{MP}} \]

from equation (5) and equation (6),

\[ Q_2 = \frac{\text{TD}}{\mu^2} \]  \hspace{1cm} (7)

Note that \( Q_2 \) is inversely proportional to \( \mu^2 \). The quality improves as the value of \( Q \) is decreases and the value of \( \mu \) is increases.

3. Result and Discussion

An Analytical Model is given a short overview of an analytical model of defect-detection techniques and refer to [12] for details. The model relates the discussed economic factors and other technical factors with the aim to analysis the defects factors at each level. The defect-detection techniques used to plan the quality assurance in a development project. Later on it used the model as a basis for reviewing the empirical literature and hence describes only briefly the assumptions and equations. A simplified version of this model is available that can be used to plan the quality assurance of a development project using historical data. It summaries the empirical knowledge available for the quality of defect-detection techniques introducing the approach in general and then describing the relevant studies and results for each of the model factors for different types of techniques and defects in general. The field of quality assurance and defect-detection techniques in particular has been subject to a number of empirical studies over the last decades. These studies were used to assess specific techniques or to validate certain law and theories about defect-detection. The first category of defect-detection techniques its look at is also the most important one in terms of practical usage. Dynamic testing is a technique that executes software with the aim to find failures [13]. The second categories of defect-detection techniques under consideration are review and inspections, i.e. document reading with the aim to improve them. The term inspection here in a broad for all kind of document reading with the aim of defect-detection [14]. In most cases review is used interchangeably. It can be identify differences mainly in the process of the inspections. In this techniques it start with analyzing the effectiveness of inspections and reviews that is later used in the approximation of the difficulty [12][14]. It also summaries these results using the lowest, highest, mean and median value in figure 3.
It observes a stable mean value that it is close to 30%. However, the range of values is huge. This suggests that an inspection is dependent on other factors to be effective. The efficiency relates the effectiveness with the spent effort. Again, this is not directly usable in the analytical model but nevertheless can give further insights into the relationships of factors. Analytical analysis techniques are evaluated in [13][14]. Interface consistency rules and anomaly analysis revealed 2 and 4 faults of 28, respectively. It also analyzed the effectiveness of programming code bug finding tools in [1][4]. After eliminating the false positives, the tools were able to find 95% of the known defects over several projects. However, the defects had mainly a low severity. For the severest defects the effectiveness reduced to 12% - 15%, for the second severest defects even to 10%. For lower severities the effectiveness lies between 80% - 95%.

4. Conclusions

The paper presents application of the defects origin iterative method for finding the approximated solution of the Phase wise Defects Removal effectiveness (PDRE) problem. The software companies spend more amount of cost on finding the defects and bad fixing. At that averages below 95% in cumulative phase wise defect removal efficiency (PDRE) is not adequate in software quality methods and needs immediate improvements. Any company or government group that does not measure PDRE and does not know how efficient they are in finding software bugs or defect prior to release is in urgent need of remedial quality improvements. When companies that do not measure PDRE are studied by the author during on-site benchmarks, they are almost always below 85% in PDRE and usually lack adequate software quality methodologies, inadequate defect prevention and inadequate phase wise defect removal are strongly correlated with failure to measure phase defect removal efficiency. For software quality is not only free but leads to shorter development schedules, lower development costs, and greatly reduced costs for maintenance and total costs of ownership.

References


