A Comprehensive Model to Improve the Efficiency of Software Inspection: A Case Study

1 Navid Hashemitaba
2 Ahdieh Khatavakhotan
3 Siew Hock Ow

Department of Software Engineering,
Faculty of Computer Science and Information Technology,
University of Malaya
Kuala Lumpur, Malaysia

Abstract:

Recently researches show that customizing inspections can increase their efficiency up to 40%. Scholars studies proved that correcting a defect in the early stages of software development is from 10 to 100 times more reasonable in cost than correcting and redoing at the final stages of development. This paper presents a novel conceptual model for software inspection. The proposed model in this article provides professional facilities to make the software inspection process more efficient. Using collaborative system enables internal and external inspectors to work together under a virtual roof and find the defects faster. Classification of defects and dynamic checklists are the core technologies for the proposed model and made it more effective comparing the traditional approach. Reduction in inspection preparation time and increasing the number of detected defects are proved through a real case study.

Keywords- Software Inspection; Inspection Model; Analysis Inspection; Design Inspection; Software Test

I. INTRODUCTION

Software engineering specialists unanimously agree that software inspections are the right technique for managing software production costs as well as improving their quality [1]. Some Scholars define inspection in details as a systematic validation and verification of a program [2]. According to them, the aim of inspection is evaluating the software quality and not the quality of software development process.

Reference [3] regard inspection as a static verification as well as a validation technique. Inspections are applicable to all software products due to the fact that they do not need a dynamic execution. Software inspection can be executed when necessary, unlike manual and automatic test tools, which are applied only after code completion. In other words, it can be initiated from the first documentation in preliminary investigation phase and can be continued until the maintenance as the final development phase.

II. THEORITICAL FRAMEWORK

Porter et al. (1997) show that software inspections have the most reasonable cost for software qualitative improvement during the last two decades [4]. As a result of applying quality–based approach, 240% improvement in detecting failures and 140% improvement in accelerating the failures detection have obtained in comparison with common inspection methods.
A. Three Key Concepts for High Quality Inspection

Denger, and Shull in 2007 presented a practical approach for inspection with high quality [5]. Three key concepts of their presented approach are as following:

Precise identification of constraints and limitations

1) Precise identification of constraints and limitations

2) Explicit involvement of stakeholders and updating them

3) Constant focus on the quality of final product and ignoring petty details

Figure 1 depicts quality-based inspection approach. Two basic questions, which can guide inspectors in all phases, are:

1) Which parts are less comprehensible and their testing is more difficult, and what appropriate conduct we should have?

2) Which missing data are of great use for testing and inspection?

The suggested approach by Denger team says that qualitative inspections or all kinds of software must be tailored according to the characteristic of each of them. The studies and experiments by Denger showed that customizing inspections can increase their efficiency up to 40%.

![Quality-Based Inspection Approach](image)

Figure 1. Quality-Based Inspection Approach adapted Denger, and Shull (2007)

<table>
<thead>
<tr>
<th>Process Association</th>
<th>Defect Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Function</td>
</tr>
<tr>
<td>Low Level Design</td>
<td>Interface, Checking, Timing/Serialization, Algorithm</td>
</tr>
<tr>
<td>Code</td>
<td>Checking, Assignment</td>
</tr>
<tr>
<td>Library Tools</td>
<td>Build, Package, Merge</td>
</tr>
<tr>
<td>Publications</td>
<td>Documentation</td>
</tr>
</tbody>
</table>
C. Inspection Metrics

Vodde in 2007 has presented a research on the metrics of software inspection. Measuring the time is sometimes absolute but sometimes applied by combining human forces (person–month) [9]. The statistical analysis of gathered data provides a clear image of inspection process for managers and people involved in this process and can facilitate the forthcoming inspection for next software.

1) Efficiency and Effectiveness of Software Inspection

Different models are developed to evaluate software inspections and motivating their usage. One of simple models presented by Gilb and Graham is evident in Formula 1 [10].

\[ \text{Efficiency} = \frac{\text{Number of defects}}{\text{Inspectors efforts}} \quad (\text{Formula 1}) \]

2) Efficiency and Effectiveness of Software Inspection

The second model defines the economic effect of inspections according to their concept, expense and the effect made for them.

\[ CE = \frac{\text{Cost saved by inspections}}{\text{Cost incurred by inspections}} \quad (\text{Formula 2}) \]

III. PROPOSED SOFTWARE INSPECTION MODEL

Figure 2 is a pictorial view of proposed inspection model. Using collaborative intelligent system is a core technology of proposed model. Defect classification, cause and effect dependency protocol, and smart inspector qualification are some of important feature of the proposed model. Next section provides a brief explanation for the core components of the model.

![Figure 2. A Proposed FTRM Model for Risk Mitigation in Software Systems](image)

IV. MODEL COMPONENTS

The proposed model has two core components besides the common functions: 1) Defect management, and 2) Dynamic check list management.

- **Defect Management**: Managing the defect is so important for succession the inspection process. Finding defects based on the predefined defect patterns is the common essential task for any inspection model. Nair,
and V. Suma (2010) do a professional research in defect managing [11]. In the aforementioned research, versatile projects are under survey during an experimental study so that inspection process is analyzed for the sake of an effective management to eliminate software defects. Using Orthogonal Defect Classification (ODC) is suggested by proposed model. Chillarege et al. (1992) have presented an orthogonal defect classification (ODC) for this process measurement, which facilitates defect prevention process (DPP) [12]. DPP determines the route of defects and takes some measures to prevent their recurrence. ODC makes an environment that provides error reports analysis automatically and without the necessity of the presence of any human beings. In this technique, defect type and defect trigger are used for development process and verification process respectively. Figure 3 lists defect types and illustrates defect triggers together with corresponding experiences. Defect trigger is a condition that reveals defects and this condition are not made in applied tests in software development process.

![Figure 3. Corresponded Defect Triggers and Defect Types](image)

- **Dynamic Check list management**: Ackerman, Buchwald, and Lewski (1989) emphasized that providing appropriate checklists is one of the first steps of inspection. In describing inspection roles, they emphasize that the presence of a developer in inspection meetings can decrease the undue secrecy of the product being inspected and facilitate review. Appendix 1 shows some fault modes together with the rate of their incidence [1].

- **Inspector Duties**: An independent person apart from artifact producer usually does inspection technique. Besides inspection supervisor, the proposed model defines two types of inspectors: Internal, and external. Both groups of inspectors have to consider the following functions

V. MODEL IMPLEMENTATION

Proposed software inspection model was implemented in a large industrial company. The candidate company has focused on vehicle parts production with a focused collaborative relationship with clients. The performance of proposed software development model is gained by adding up the amount of saved effort and the reduced resources because of encountering defects [1].

Table 2 displays the outcome of proposed model application in the pilot company web development project. Column 1 shows the number of phases. Column two contains the main resource, total time spent to
complete each phases including the inspection process time based on person-per-week. Column 3 shows the extent of effort made by the inspection experts to pinpoint defects based on one person per week.

VI. NEW MODEL EFFECTIVENESS AND EFFICIENCY

To verify the proposed model efficiency, a comparison experiment is launched. The experiment inspected nine artifact of commercial software using traditional inspection technique and proposed model by two groups of inspectors. All artifacts were produced during design phase as work products. A pretest survey is done to ensure that the related skill, and experience of both teams are the same. The inspectors use common checklists to prevent any biases. Each data set presented by inspectors has three subsets: (1) Number of found defects, and (2) the number of false positive defects.

1) Inspection Effectiveness: The results of experiment by using two traditional and proposed models are shown in table 1. The captured defects by two groups and for each artifact are represented in table 3. Fig. 4 reflects the total defects founded by traditional and proposed inspection models.

The results reflect more than 26% improvement in effectiveness. This means proposed inspection team has found 20 defects more than traditional inspection team.

2) Inspection Efficiency: According to reference [13], time reduction is a major criterion to approve the efficiency of an inspection model.

Table 2 presents the effort of inspectors in person/hour metric. The diagram in Figures 4 to 8 portrays that using proposed model had reduced inspection time. In other words, the proposed model has increased the efficiency 40% by reducing 20 hours elapsed time to inspect all artifacts. The average time for each defect in traditional approach is:

- \( \text{Average Defect for Traditional} = \frac{362}{376} \approx 54 \text{ minutes for defecting each defect} \)
- \( \text{Average Defect for new model} = \frac{277}{416} \approx 40 \text{ minutes for detecting each defect} \)
### Table II: The Results of Performance Calculation of Case Study

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Inspected Artifact</th>
<th>No. of Art.</th>
<th>Traditional Approach</th>
<th>Proposed Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Effort (ph)</td>
<td>Efficiency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IP</td>
<td>DD</td>
</tr>
<tr>
<td>1.</td>
<td>Failure mode system</td>
<td>4</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>2.</td>
<td>Measurement system analysis</td>
<td>3</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>3.</td>
<td>Process Audit</td>
<td>5</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>4.</td>
<td>Product Audit</td>
<td>2</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>5.</td>
<td>Structuring the feasibility</td>
<td>6</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>6.</td>
<td>Inventory control</td>
<td>2</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>7.</td>
<td>Project control</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>8.</td>
<td>Customer satisfaction</td>
<td>3</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>9.</td>
<td>Timing system</td>
<td>4</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>10.</td>
<td>Identifying and tracing system</td>
<td>8</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>11.</td>
<td>Product planning and realization</td>
<td>3</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>12.</td>
<td>Calibration</td>
<td>2</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>13.</td>
<td>Quality control</td>
<td>6</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>14.</td>
<td>Document and history control</td>
<td>5</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>15.</td>
<td>Product review</td>
<td>4</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>16.</td>
<td>Internal auditing</td>
<td>4</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>17.</td>
<td>Maintenance</td>
<td>6</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>70</td>
<td>125</td>
<td>251</td>
</tr>
</tbody>
</table>

![Figure 1: Efficiency comparison for traditional and proposed model](image1)

![Figure 2: Number of Defects](image2)

![Figure 3: Efficiency comparison for traditional and proposed model (FP)](image3)
VII. CONCLUSION

The proposed model in this article provides professional facilities to make the software inspection process more efficient. Using collaborative system enables internal and external inspectors to work together under a virtual roof and find the defects faster. Classification of defects and dynamic checklists are the core technologies for the proposed model and made it more effective comparing the traditional approach. Reduction in inspection preparation time and increasing the number of detected defects are proved through a real case study. Eliminating hidden defects of software artifacts with proposed model, which was not detected in traditional inspection approach, shows a 26% saving in human resources.

REFERENCES


