Prosperity in Software Inspection: Improvement of Software Development Process Using an Inspection Smart Boost

1Navid Hashemi Taba (IEEE Member)  
2Siew Hock Ow  
3Ahdieh Sadat Khatavakhotan (IEEE Member)  
Department of Software Engineering,  
Faculty of Computer Science and Information Technology,  
University of Malaya  
Kuala Lumpur, Malaysia  
nhtaba@siswa.um.edu.my  
show@um.edu.um  
khotan@siswa.um.edu.my

Abstract—Software engineering specialists unanimously agree that software inspections are the right technique for managing software production costs as well as improving their quality. However, software testing is mostly used instead of inspections in development process, to achieve software desired quality, inspections must be highlighted and focused more in software production cycle. The model presented in this paper has improved inspection process through software development based on life cycle as the most popular development model. In this article, some discussions to do with the revision of ingenious inspection models are raised. In order to evaluate the model, a case study has been done in an IT company. The founded facts show that IT projects inspection models can have an appropriate efficiency and officially just by having a suitable array both superficially and profoundly.

Keywords-Software Testing, Software Inspection, Development Test, Validation and Verification, Software Release, Software Development

I. INTRODUCTION

The design of test and inspection models of project and software products does have its own specific intricacies [1]. The considerations to do with software projects inspection are noticeable on account of the fact that software products and services are more a collection of rules, programs, data and documents than have a physical quality [2]. Therefore, the inspection of projects and products based on the inspection of procedures, trends, logical characteristics of work products, final product is much more difficult than the inspection of other product’s physical characteristics of products, and final product is much more difficult than the inspection of other product’s physical characteristics or the physical progress of other types of projects.

In this article, some discussions to do with the revision of ingenious inspection models are raised and it does emphasize that software projects inspection models can have an appropriate efficiency and officially just by having a suitable array both superficially and profoundly. A deep inspection leading to discovering any error can create a generative inspection that if the findings are registered systematically and making cause and effect logical rules and adding to inspection process can give the inspection models a kind of learning property, which increases the ingenuity of such models.

II. QUALITY IMPROVEMENT THROUGH DEFECT MANAGEMENT

When the inspection process faces a problem, the unfinished work being concerned, requires an action plan which exhibit what the next steps are. Ramler at al. in reference [3] stated: “Defect prediction promises to indicate defect-prone modules in an upcoming version of a software system and, thus, allows focusing the effort on those modules.” Therefore, inspection models which only rely on a report of defect and do not follow the causes or root of a problem profoundly, try to nab people involved and routines instead of cooperating in solving problems fundamentally. Finding a defect in a product which has met quality control checkups and versatile tests show that designers and people involved in process have had some weaknesses in one case or some cases [4] but quitting them will not make any progress in future.

The concept of discovered problems with inspection implies that quality control and quality assurance models do not work properly [5]. Although quality control routines and quality assurance standards utterly determine that in case tests or controls are not met, what repairs or reworks must be done, after finding a problem by inspection activities, it will not be determined what process must be taken to solve it [6]. In other words, a revision resulting in the discovery of a defect shows that diagnostic routines and solving the errors of structural and development models are not fully responsive and it is necessary to manifest needed measures and recommend essential solutions.

III. INSPECTION, AN ACTIVITY BEYOND DEVELOPMENTAL MODEL

Usually inspections come during production process and in some cases after producing a work product. The existence of inspection in development model or out of that, each bear
some advantages and disadvantages. Mostly, inspections include some activities out of development model so that the problems of the model are justified. To complete the aforementioned points is that work products are evaluated via pre-determined criteria during production and development process usually called quality control, and is assigned to some processes and people out of production process [7].

On one hand, some inspectional activities out of production process and production development including software products are dealt with so that both process and people involved and also product is assessed by taking a different perspective and without any bias to production trend. On the other hand; being out of development model in software and IT products is the most important reason for making a problem in projects’ planning and coordinating activities [8].

The inspections can disrupt or defer the process of project which in both cases although quality is improved, project and product planning suffers. Kurt and Kharoufeh in reference [9] stated: “If the system is found to be failed at an inspection, then it must be either repaired or replaced with a new one at an additional penalty cost.” Therefore; in development models design, a prediction of program inspection must be considered despite the fact that inspectional activities are designed and planned out of development model so that there is a decrease in negative effects of inspection in program being executed and the deadline of final product [10] [11].

IV. OBJECTIVE OF RESEARCH

The objective of this research is improving the performance of software testing by proposing a novel inspection based development model. Therefore, the objective of this research is increasing the quality of software explicitly [12].

V. INSPECTION ORIENTED SOFTWARE DEVELOPMENT

According to F. Shull et al. [13], Test-Driven Development (TDD) delivers higher quality software. Figure 1 is a pictorial view of Inspection Oriented Software Development (IOSD) model, as an specific form of TDD presented in this article. The IOSD model comprises nine distinguished major phases. Each phase has independent activities and its output is transferred to central Smart Inspection Boost (SIB) core to determine its validation, verification, and certification. The IOSD is an iterative life-cycle based model. The output of each phase could be assumed as the gross input for next phase that after validation in central boost could be used. The IOSD model has a clockwise movement. Figure 2 shows the major documents and artifact, which are candidates for inspection during inspection process in each development phase.

VI. SMART INSPECTION BOOST (SIB) CORE

Meyer in reference [14] stated, “Testing strategy’s most important property is the number of faults it uncovers as a function of time.” After a completing each phase, inspection team makes an accordance between performed activities and phase plan. The aforementioned comparison is the responsibility of Smart Inspection Boost (SIB) core. Two checklists are provided for confirmed matters and encountered defects respectively. The origin of such defects are dugout. A cause-and-effect tree is generated and illustrated for each defect. Eliminated defects and project documentation are updated. A description on defects is provided and its causes and elimination is well covered in learning documentation so that it can be applied for other similar projects.

VII. SIB ARCHITECTURE

Figure 3 shows the main part of inspection core called SIB. This core comprises three parts as hierarchy, knowledgebase, and an integrated database.
A. Defect determination

The first part can pinpoint defects. Independent experts by referring to knowledgebase, database, and each phase documents can find out potential errors and possible defects with conventional and innovative routines. Such defects are analyzed and a cause-and-effect chart is illustrated for each defect. The information to do for each defect will be recorded in relevant forms. Inspection core entails three independent but relevant parts. Defect detection, defect elimination together with logging, recording, and categorizing detected defects and the discovery of their logical relation has gifted the model with a learning property.

Decreasing the failures and defects using the proposed SIB core will mitigate the development lateness as the most significant weakness of SDLC models [15].

B. Defect removal

In the second part, inspection team announces the result of each inspection together with their elimination approaches to phase developer team. Inspection team is responsible for supervising precise administration of such changes and removing defects. Reference [16] suggested the semi-automated method to improve the test efficiency. Therefore, second part results will transfer to the third part of core through pre-defined on-line or conventional forms.

C. Defect Learning

In the third part, the information to do with each extracted defect can be inserted into project database, logical regulation related to defect occurrence will also be registered in the knowledge base of project. Master knowledge base and the main database will be updated according to each phase results. The learning feature has a significant positive impact on the total performance of software development [17].

VIII. PERFORMANCE CALCULATION AND EFFECTIVENESS EVALUATION OF IOSD MODEL

The performance of proposed software development model is gained by adding up the amount of saved effort and the reduces resources because of encountering defects [18]. Table 1 conducted the performance as well as efficiency evaluation formulae. Formula number one to six show the inspection performance calculation as well as the effectiveness evaluation of the model for each phase. Total performance and total effectiveness of the model for all phases can be calculated by formula 2 and 6. It is worth mentioning that the required resources as well as saved resources have to be translated to a unified unit such as time or currency unit. Formula number 4 shows the inspection effectiveness of the model for each phase. Total performance and total effectiveness of the model could be calculated through formula 5 and 6 respectively.

<table>
<thead>
<tr>
<th>Formula no.</th>
<th>Formula</th>
<th>Parameter description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula 1</td>
<td>IP = Re_{i+1} - Ie_i</td>
<td>IP: Inspection performance in phase i, Re: Rework effort in phase (i+1), Ie: Inspection effort in phase i</td>
</tr>
<tr>
<td>Formula 2</td>
<td>Te = Pe_i + Ie_i</td>
<td>Te: Total efforts in phase i, Ie: Inspection efforts in phase i, Pe: Efforts for phase i completion</td>
</tr>
<tr>
<td>Formula 3</td>
<td>Ic = 1 / Te_i</td>
<td>Ic: Inspection coefficient in phase i</td>
</tr>
<tr>
<td>Formula 4</td>
<td>Ep = IP * Ic_i</td>
<td>Ep: Effective Percentage for phase i</td>
</tr>
<tr>
<td>Formula 5</td>
<td>TIP = \sum_{i=1}^{9} IP_i</td>
<td>TIP: Total Inspection Performance</td>
</tr>
<tr>
<td>Formula 6</td>
<td>TIE = \sum_{i=1}^{9} IE_i</td>
<td>TIE: Total Inspection Effectiveness of the IOSD model</td>
</tr>
</tbody>
</table>

If the performance of phase i is more than zero, the inspection performance is positive. If the performance amount of phase i is less than zero, the inspection performance in that phase is negative. The performance is being zero implies that applying model has not had any materialistic harm. The effectiveness of model (formula 4) could be evaluated based on the performance and the allocated resources to the inspection activities.

Such performance calculation and effectiveness evaluation only covers project management process and the positive consequences of discovering and eliminating defects are not considered [19]. Making mutual assurance between customers and organization, punctual product delivery and deducting maintenance costs are some of the consequences of discovering and eliminating concealed flaws. Aforementioned items can intensify the performance of the model.
IX. Model Implementation

Proposed software development model was applied in website system development of an IT company. The candidate company has focused on website designing to make a closer relationship with clients. The most important of company website is taking orders on-line and supplying products for clients. The chosen model to develop a website for company was adopted SDLC model. The project managers agreed that parallel to website development, inspection core of IOSD model could be applied. Hence, proposed inspection processes are done after each phase. Acquired results from the application of the IOSD model show the efficiency of the model.

A. Interpretation of case study result

Adopted formula 1 to 6, effectiveness percentage of phase number i in the studied case equals performance amount in phase i multiple by inspection coefficient. The subtraction of the amount of reworking from the effort made to discover and eliminate defects in phase i could be assumed as the performance of applying the inspection core in each phase. The inspection coefficient for each phase could be calculated by dividing the inspection efforts by total required efforts to completion that phase.

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

<table>
<thead>
<tr>
<th>Phase</th>
<th>Phase no.</th>
<th>Inspection efforts</th>
<th>Total efforts</th>
<th>Rework efforts</th>
<th>Inspection coefficient percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>8</td>
<td>32</td>
<td>9</td>
<td>8.03</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>3</td>
<td>23</td>
<td>10</td>
<td>7.04</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>14</td>
<td>4</td>
<td>2.07</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>15</td>
<td>8</td>
<td>5.07</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>3</td>
<td>23</td>
<td>5</td>
<td>2.04</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>5</td>
<td>21</td>
<td>7</td>
<td>2.05</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>7</td>
<td>3.06</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>128</td>
<td>32</td>
<td>160</td>
<td>61</td>
<td>36.06</td>
</tr>
</tbody>
</table>

The total effort for each phase completion is well portrayed in column four. Column five describes the sum up of required reworking during next phases or maintenance phase in the case of not finding defects addressed by professionals. Column six shows the inspection performance for each phase according to formula 1. inspection coefficient is calculate using formula 4 and presented in seventh column. The last column portrays the percentage of effectiveness of each phase according to formula 4.

B. Performance of the model

Due to the fact that software development in the studied case has done in the first iteration, it bears no indication and usage for maintenance phase of the model. Figure 4 presents an efficiency diagram to compare the phases’ efficiency for evaluate the total performance of the model. Using modified performance formula has shown that the utmost amount of efficiency is attributed to the application of model by smart inspection core with 42% and is related to phase one. Afterwards, phase 5,3,2 with 35, 28, and 24 percent respectively has had the highest percentage of efficiency to model application. The total amount of model efficiency for all phases during first iteration is 22%, which for a large project is not negligible.

![Figure 4. The efficiency of model implementation for all phases](image)

X. Conclusion

The IOSD model with SIB central core has covered SDLC typical models weak points. The initiation of model with preliminary study is called initializing phase, which can make planning capability independent from detail analysis possible. The proposed model has allocated an integrated inspection core to all phases to support V&V, and certification all along life cycle.

A case study has been done to verify a model on a website development project. The proposed model of this article simultaneous with a SDLC traditional model based on life-cycle has been used in a website development project. Pinpointed defects in each phase are recorded and analyzed in relevant tables. Eliminating hidden defects of project with proposed model, which was not detected in traditional SDLC model shows a 22% saving in human resources. The wasting or saving about one-quarter of vital resources can lead to the failure or success of a large software project. Because of variety of software types as well as different characteristics
of implementation environments, model customization is highly recommended.

REFERENCES


