A Scenario-Based Model to Improve the Quality of Software Inspection Process

1Navid Hashemi Taba  
2Siew Hock Ow  
Department of Software Engineering,  
Faculty of Computer Science and Information Technology,  
University of Malaya  
Kuala Lumpur, Malaysia  
1nhtaba@siswa.um.edu.my  
2show@um.edu.um

Abstract—Independent researches of expert groups and scientific papers show that, although some testing models and automated and manual tools are mostly used, but still some software and software projects deal with important defects that may lead to project failure. Most of software inspection models are independent of software developing process and concentrating on inspection artifacts. In this article, the proposed model is based on software developing phases adapted predefined scenarios. This inspection approach removes some possible defects in each of software development phase and avoids amplifying these defects in each of the next phases. Defect learning is the interesting point of scenario-based proposed model. Learning is a basic factor for a model to be intelligent. In this model a learning plan is created and executed through which, according to founded results, inspection process is modified and updated. The efficiency of model is evaluated through a case study.

Keywords—Software Inspection, Scenario Based Model, Software Quality, Software Development, Analysis and Design

I. INTRODUCTION

The experiments made by Genuchten and his team [1] shows that well–executed inspections can discover between 60% to 80% of the defects in software lifecycle, before test phase. Ackerman, Buchwald, and Lewski [2] 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.

Weinberg and Freedman [3] have done a comprehensive study for review, walk-through and inspection. They have emphasized that Formal Technical Review (FTR) must be in the scope of responsibility of those not involved in software production. Hence, they attribute the main role of FTR for providing trust worthy information from software products to managers.

The most important researches on software testing and specially software inspection approaches in the last three decades showed that it is necessary to improve inspection models and at the same time we must develop software such that their working products could be inspectable. The objective of this article is improving the performance of software testing by proposing a novel scenario-based inspection model. Therefore, the objective of this research is increasing the quality of software explicitly.

II. TRADITIONAL INSPECTION PROCESS

Figure 1 shows the traditional software inspection processes [4]. Formal inspection plays an essential role in software quality. There are different combinations of review committees, various routines and frame works for formal technical reports.

A. Inspection Common Problems

The inspection case study by Kollanus [5] presents that two important common problems could be seen in all inspection processes in analysis and design phases of several studied organizations:

1. Inspection process in none of considered organizations has been inspected and this lack of inspection especially inspection meetings as a critical part of process decreases its reliability.
2. In all those organizations, inspection training has been the weak point and individuals taking part in inspection is the most important part of process and their not being skillful renders the inspection with no quality.

III. THE PROPOSED INSPECTION MODEL

Figure 2 is a pictorial view of the proposed model for scenario based software inspection. As it can be seen in the figure, the inspection activities are located among two phases and make transition from one phase to another, changing from one simple shift to an intelligent semi process. Although there are some defects detection and removing activities, using defects databases and knowledge bases containing defects producing rules, changes traditional development model to an intelligent model that improves the performance and efficiency of SDLC specially in delivering and releasing final product.

A. Model Capabilities

This model has some capabilities like, document based, staged, milestone and deliverables are identified in each phase, but has some disadvantages too. One of the most important disadvantages of the approach is that it takes some costs and in some cases there is no way to return to the previous phases. Clearly speaking, the problems found in a phase, if have some reasons in the previous phases, must be a return to the previous phase that itself has some bad feedbacks like repetitive control actions and document reforms.

According to the above comments, some researchers talk about amplifying some defects from one phase to the next. And some experts and researchers, know that improving final tests in each phase specially validation and verification actions, are an approach for avoiding from defects to be amplified in the phases of the project. But all the tests natures are insurance of subsystems performance and correct artifacts documents, work products and deliverables. Most of the testers are some bodies who were system developers so if they forget something in the software development phase, they will not pay attention to them whenever they test the software. Specially that, formal statics and independent researches of expert groups and scientific paper show that, although some testing models and automated and manual tools are mostly used, but still some software and software projects deal with important defects that may lead to project failure. Software inspection concentrates on some working products, documents, deliverables, artifacts and whole of the software, to find possible defects. We mean by defect, the errors that cannot be detected by failure detection procedures or quality control activities and quality guarantee models. These failures that are hidden after doing some tests or V and V activities can be the source of some hidden and clear defects in the next phases of software development. We can detect and remove the rest of the defects and reduce the defects increasing, by adding inspection to the SDLC model specially using some independent expert inspectors. The scenario based inspection model not only leads to quality improvement of produced software, but also leads to production process improvement and detecting some possible defects. Each phase give a better position to the next phases activities that are finally more beneficial to the risk managers and project managers.

IV. UNIQUE CHARACTERISTICS OF THE PROPOSED MODEL

Most of software inspection models are independent of software developing procedure and concentrating on inspection artifacts. In this article, the proposed model is based on software developing phases adapted to external SDLC methodology. At the end of each of analysis and design phases, the recommended actions are not only

Figure 2. Scenario based inspection model
concentrating on documents, deliverables and non-deliverable working products, but also do the software inspection process implicitly and gradually.

A. Gradually Inspection

The most important aspect of the proposed methodology is to concentrate on software inspection as a very important factor for improving software engineering process, in other words, the most important characteristics of this proposed methodology is that, it inspects gradually and step by step software developing procedure in all of its steps. This inspection approach removes some possible defects in each of software development phase and avoids amplifying these defects in each of the next phases.

B. Work Product Quality Improvement

The model improves the quality of software work products during development process. The inspection activities during software development process not only leads to quality improvement of each phase working products, but also prepares a base for final inspection of a completed software product that is ready to release. The other advantages of this gradually inspection are:

1) Transition facilities. Defects that are discovered and removed in each phase (or being discovered and removed), are a practical help to prevent the same mistakes in each next phases.

2) Scheduling improvement. In each iteration of the process the discovered defects in the previous iterations and the same previous phases facilitates a more consciously scheduling to produce.

3) Cause and effect paradigm. The rules of cause and effect established in the process of gradually inspection activities, leads in to complementary test models and improving test cases.

4) Facilitate Risk management. The data and information gathered for the detected defects and removing the defects actions, improves risk management and evaluations in maintenance and production phase of the same products with the same methods.

V. MODEL STAGES

The most prominent property of inspection model that is used in early phases of software development, is its consistency and uniformity that is on the top level of abstraction. That is, although there are some completely different activities and some various working products, in each of analysis and design phases of SDLC model, there are three golden steps suggested by the model that are applicable:

1) Golden step one: defect determination. In this step the inspectors learning process starts with preparing the information related to potential kinds of defects, in hand tools and collaboration framework. The important expected action in this phase is defect mapping that identifies which defects with which properties are related to what section, location, component, document or artifact. To detect the defects we use inductive and deductive methods. The common methods of step by step inspection artifacts and innovative methods of detecting the defects are done throughout using automated tools, simulation and stub.

2) Golden step two: defect removals. Something that distinguishes the proposed model from other common methods of inspection and test is trying to detect report and remove the defects. In this method, by considering the previous step stored information, it is clear that: which actions are done by whom with which skills and using which instruments and resources in how many times and in which order to clear the defect.

3) Golden step three: defect learning. The most interesting point of this step brought to inspection model is creating an intelligent model. Learning is a basic factor for a model to be intelligent. In this step a learning plan is created and executed through which, according to founded results, inspection process is modified and updated. Strictly speaking, in this step three following actions are done.

   a) Database updating. By finding some defects, the relations and fields of defects databases are updated.

   b) Knowledgebase updating. Defects logical rules are extracted and then inserted into the knowledgebase.

   c) Training program enhancement. Inspectors training programs, that are preparation programs or defects discovering and registering methods, will be improved. Depending on number and type of defects or the created rules and by using committee chairs idea, we may use some new training programs or make some reforms to the current programs.

VI. PERFORMANCE CALCULATION AND EFFICIENCY EVALUATION OF IOSD MODEL

The performance of proposed software inspection model is gained by adding up the amount of saved effort and the reduces resources because of encountering defects [6]. Table 1 conducted the performance and 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 [7].
Total performance and total effectiveness of the model for all phases can be calculated by formula 5 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 (Person-Week). Formulae number 4-a and 4-b show the inspection efficiency of the model for analysis and design phase. Total performance and total efficiency of the model could be calculated through formula 5 and 6 respectively.

### VII. CASE STUDY

Proposed software development model was applied in analysis and design phases of a software project in an international general company. The candidate company has focused on customer relationship services to make a closer relationship with clients. The proposed inspection model is done during each phases of earlier development stages. Acquired results from the application of the model show the performance and efficiency of the model.

#### A. Interpretation of Case Study Results

Table 2 displays the outcome of model application in aforementioned company customer relationship project. Column 1 shows the name of phases. Column 2 has the main human resources for each phase, total time spent to complete each phases exclude 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>No.</th>
<th>Formula</th>
<th>Parameter Description</th>
</tr>
</thead>
</table>
| 1   | a) $IP_a = SR_a - IR_a$ | $IP_a$: Inspection performance for analysis phase  
$SR_a$: Saved resources by inspection in analysis phase  
$IR_a$: Inspection resources of analysis phase |
|     | b) $IP_d = SR_d - IR_d$ | $IP_d$: Inspection performance for design phase  
$SR_d$: Saved resources by inspection in design phase  
$IR_d$: Inspection resources of design phase |
| 2   | a) $TR_a = ER_a + IR_a$ | $TR_a$: Total resources of analysis phase  
$ER_a$: Expected resources of analysis phase  
$IR_a$: Inspection resources of analysis phase |
|     | b) $TR_d = ER_d + IR_d$ | $TR_d$: Total resources of design phase  
$ER_d$: Expected resources of design phase  
$IR_d$: Inspection resources of design phase |
| 3   | a) $IC_a = 1/TR_a$ | $IC_a$: Inspection coefficient for analysis phase |
|     | b) $IC_d = 1/TR_d$ | $IC_d$: Inspection coefficient for design phase |
| 4   | a) $IE_a = IP_a * IC_a$ | $IE_a$: Inspection efficiency for analysis phase |
|     | b) $IE_d = IP_d * IC_d$ | $IE_d$: Inspection efficiency for design phase |
| 5   | TIP = $IP_a + IP_d$ | TIP: Total inspection performance |
| 6   | TIE = $IE_a - IE_d$ | TIE: Total inspection efficiency |
TABLE II. THE RESULTS OF CASE STUDY

<table>
<thead>
<tr>
<th>Phase</th>
<th>Phase efforts</th>
<th>Inspection efforts</th>
<th>Total efforts</th>
<th>Rework efforts</th>
<th>Inspection performance</th>
<th>Inspection coefficient</th>
<th>Effective percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>18</td>
<td>6</td>
<td>24</td>
<td>14</td>
<td>8</td>
<td>0.04</td>
<td>32%</td>
</tr>
<tr>
<td>Design</td>
<td>32</td>
<td>4</td>
<td>36</td>
<td>11</td>
<td>7</td>
<td>0.03</td>
<td>21%</td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td>10</td>
<td>60</td>
<td>25</td>
<td>15</td>
<td>0.02</td>
<td>30%</td>
</tr>
</tbody>
</table>

Note: Table note.

* Unit: person per week

The total effort for each phase completion is well portrayed in column 4. Column 5 describes the sum up of required reworking during next phases or maintenance phase in the case of not finding defects addressed by professionals. Column 6 shows the inspection performance for each phase according to formula 1. Inspection coefficient is calculate using formula 3 and presented in seventh column. The last column portrays the percentage of efficiency of each phase according to formula 6.

B. Performance of the Model

Figure 3 presents an efficiency diagram to compare the phases’ efficiency for evaluate the total performance of the model. Using performance formula has shown that the utmost amount of efficiency is attributed to the application of model by proposed approach 32% and is related to analysis. Design phase with 21 percentage has had the second rank of efficiency to model application. The total amount of model efficiency for all two phases is 30%, which for a large project is not negligible.

VIII. CONCLUSION

The proposed inspection model in this research is concentrating on two aspects of software inspection and developing models. Software developing aspect, prepares an approach that by concentrating on common methodologies based on lifecycle in each phase, the documents, working products and deliverables would be inspectable. The other aspect is to propose compatible unified and integrated approach that suggests suitable inspection techniques in each of software developing phases. Some quantitative and qualitative indices measures and metrics are proposed in this model to evaluate reliability and efficiency of the proposed model. To implement this model, a big productive industrial company is selected and its quality control system is developed by the proposed methodology. The results shows the efficiency and effectiveness the model comparing the traditional inspection process.

REFERENCES