Comparison of Software Inspections and other Software Quality Assurance Techniques

Shama Shahzadi
Department of Information Technology
University of Lahore
Gujrat Campus, Pakistan
shamapugc@gmail.com

Muhammad Umair
Department of Computer Science
Virtual University of Pakistan
Pakistan
ms150200096@vu.edu.pk

ABSTRACT-In this paper, we will analyze and discuss how software inspections overweigh other software quality assurance techniques such as reviews, testing, inspection, etc. This will be done by providing the historical data of companies that used inspections while building their software products and comparing it with the data when software inspections were not used. Our results will show that software inspections reduce the development cost of the software product and also improves the quality of the underlying software. Software inspections, testing, reviews, inspections and other SQA techniques will also be discussed briefly. Besides this, we will also look at why companies still hesitate to use inspections and think that this is an overhead cost. We will try to answer the questions that companies put forward for not using Software inspections.

KEYWORDS- Inspections, Reviews, Testing

I. INTRODUCTION

The purpose of writing this paper is to compare different software quality assurance techniques. Our analysis will show that software inspections overweigh any other software quality assurance technique in terms of cost, quality, benefit and time. We will start with brief introduction of software quality assurance techniques i.e. Reviews, Testing, Inspections, etc. We will also have a look at the benefits of each. Also, we will look at the team structure for performing each technique. While discussing each software quality technique we will look at the historical data of different companies that how this technique either resulted in benefit or detriment for the company.

Organization of paper follows as under. In section II, we will give a brief overview of related work. Section III, IV, and V discuss brief overview of Reviews, Testing and Inspections respectively. Sub-section of each of above section debates on benefits, importance. Also, we will discuss the team structure for performing underlying software quality assurance technique. Historical data will be presented to see the cost of each technique. In section VI, a comparison will be done between Reviews, Testing and Inspections and we will see how Software Inspections overweigh the benefits as compared to other techniques. In section VII, we will try to answer the questions that companies put forward as the reasons for not using software Inspections. Section VIII discusses the future of Software Inspections.

II. RELATED WORK

Our work for this paper is to provide an analysis of different software quality assurance techniques such as Reviews, Walkthroughs, Testing and Inspections; and show how Inspections overweigh the advantages than any other Software Quality Assurance technique. Before jumping on to analysis of each, we start with discussion on each software quality assurance technique.

III. REVIEWS

Introduced by Fagan at IBM, Review is of one of those quality metrics which that is used to ensure the quality of project deliverable. Review involves examining the software, its documents and its related material to discover areas where software project is lagging behind and quality standards have not been followed. Usually all software documents such as SRS, design document, code, process models, test plans etc. are reviewed during the review process. Reviews ensure that all the documents are conforming to the standards and are complete in all respect.

Whenever problems are found during review process, they are written down and passed on to the person who is responsible for correcting these errors.

REVIEW PROCESS- Review process normally consists of three phases, i.e. Pre-review activities, review meeting and Post-review activities.

Pre-Review activities: Review preparation and review planning is done in this phase. Review planning includes tasks such as setting up a review team, arranging a specific data and time for review and distributing the documents that need review. On the other hand, review preparation activity prepares the participants of the meeting in order to hold effective meeting. Team members usually meet (either formally or informally) in order to get overview of the software to be reviewed. Individuals try to look into the software and its related documents in order to prepare themselves for the review meeting.

Review Meeting: In review meeting, program or document that is being reviewed is put forward before the
other members of the review team. Review meeting must not last more than two hours. Review decisions and actions that need to be taken are recorded.

**Post-review activities:** The issues raised during the review meeting are addressed. These issues can vary ranging from deficiencies in requirements to coding errors. Management review is followed up with this activity in case if it is to be decided whether more resources are needed to correct the issues raised during review meeting.

**REVIEWS: BENEFIT OR DAMAGE?**

Whether review meetings should be held or not is still questionable in software engineering domain. Fagan believed that review meeting is crucial as most defects can be identified during review meetings [1]. According to [2], review meetings don’t carry a significant impact. On the other hand [3] showed that reviewers were able to find 90% of defects while preparing for the software review meeting whereas only 10% defects were found during software review phase. Other studies such as [4] and [5] also don’t speak in favor of review meetings.

**IV. TESTING**

Software Testing is a formal procedure to detect and then remove the defects that incurred during the development of the software. Testing is usually to demonstrate that developed software meets its requirements. Also, testing tells us that behavior of the system where it does not behave as it would have in normal circumstances.

**TEAM STRUCTURE IN REVIEWS**

Review team comprise of 3-4 members who serve as principal reviewers. One of them serves as senior designer who is responsible for all significant decisions during the review meeting. Principal reviewers may also invite other members who are currently working on system development (an example is designer of the system) to contribute to the review. They may not review the whole document but may review that work which is related to them or affects them during development. Project manager is usually not involved in the review meeting, however, he/she will have to take part in the review meeting if some problems require change in the project plan.

**TEAM STRUCTURE IN TESTING**

Following two levels are often included in software test teams.

**Test Manager** is responsible for all responsibilities of test planning. He or she is also responsible to check if sufficient resources are available to carry out the testing activities. Test manager also needs to ensure that status report of testing activities is being prepared on regular...
basis and if interaction with customers is required on
 certain cases, then he or she must arrange for it. Test
 manager is also responsible for updating the project
 manager with test activities.

Test Engineers, Quality Assurance Testers and
Quality Control Testers need to read all the documents
so that they understand what needs to be tested and
deciding how it will be tested, developing test cases,
executing test cases and report about defects along with
their priority and severity. When code is changed to fix
defects, they are responsible for regression testing.

GUIDING PRINCIPLES FOR TESTING:

For successful software testing, following principles must
be followed. [6]

- Start the testing process as early as possible
  because when you will start early, it will lead to
  less cost. This will help fix more errors in the
  early stage of software development. Cost of
testing is shown in Figure 2.
- Testing is the process to find defects, therefore,
  our objective should be to find as many errors as
  we possibly can, only then true objective of
  testing can be achieved.
- Complete and to the point requirements are
  crucial to successful software. Requirements
  should be clear before testing. Test cases must be
  written prior to performing testing.
- Valid as well as invalid inputs must be given to
  the system to perform testing. This will make
  sure how the system behaves for invalid inputs.
- Errors usually come in clusters. When an error is
  discovered in a piece of code, probability
  increases that more errors will be discovered in
  that code. Therefore, additional testing may be
  required for such areas.
- Software Testing will not stop. It is a never
  ending process that has to be stopped
  somewhere.

![Figure 2: Cost of fixing defects](image)

TESTING: BENEFIT OR DAMAGE?

Studies have shown that Software Testing is that step in
software development process where most defects are
identified. Software Testing adds value to the
organization because defects identified during testing are
fixed afterwards and therefore quality product is shipped
to the customers. Faulty product leads to customer dis-
satisfaction and earns less revenue and value for the
organization. Also, when customers identify defects
while using the product, it takes more time and money to
fix defects at that stage. Testing is the only process in
software development that makes software useable
because it helps identify defects. If defects are not fixed,
then customers will stop using the product.

V. INSPECTIONS

Software inspection is a process where team
members of software development team meet to find out
bugs in the program and where product has not been
conforming to the standards. Software inspections can be
applied to any complete deliverable or incompletes deliverable of software development process
[7]. Software Inspection is somewhat opposite to
Software Testing in the way that software does not
necessarily needs to be tested during Software Inspections
in order to find out bugs. If Testing is not performed then
how defects are found? The answer to this question is that
a checklist of common bugs to a specific domain are used
to find bugs in the program. Also, we can use checklist of
different programming errors that occur in a specific
language to help identify bugs in the product. Some
common type of errors that are looked during inspections
are as follows:

- Are values of all variables are initialized before
  their use?
- Is upper bound of Array is one less than its size?
TEAM STRUCTURE IN INSPECTIONS

Minimum team size in Software Inspection comprises of three members, however, this number can vary from four to seven. High level documents may require larger teams. Members can be added when their point of view needs to be taken into consideration. Inspector, Moderator, Reader, Recorder and Author play vital role in Software Inspections. We take a look at role of each briefly in the following.

Inspector performs the duty of finding defects in the product. Besides performing duty as an inspector, an inspector may perform duty as moderator, author, reader, and recorder whenever possible.

Usually those personnel are considered good for inspector who have done some similar work in the earlier projects. People who wrote requirements can be a good choice for design inspection. Outside inspectors can also be brought in if they have better knowledge of the system being built.

Moderator is the one who leads the inspection meeting and ensures that good inspection is carried out. This role is the most critical to formal inspection process therefore, moderators must be trained. Moderator remains active during all inspection activities. Moderator performs duties such as:
- Selecting members for the inspection team
- Assigning roles to team members individually and together
- Leading the team throughout the process
- Collecting inspection data on inspection report forms

Reader is responsible for reading the document. He also tries to explain the document with different examples so that everyone in the meeting can understand it clearly and only one interpretation is made by all the participants of everything that he or she reads.

Recorder is responsible for recording the defects and issues that were raised during the inspection meeting. Recorder acts as a reader too as he after recording the defect reads it in the meeting to make sure to everyone that he has written only what was reported and no error has been done by him while recording the defect.

Author creates the underlying work product to be inspected. He or she also answers questions during the meeting so that misunderstanding may not lead to mark something as a defect when actually it isn’t. Author also acts an inspector during inspection meeting as he corrects all the defects found during the inspection meeting.

INSPECTION PROCESS

Previous discussion highlighted roles of participants of the inspection meeting. Now we take a look at the inspection process.

Planning: In this phase, participants for the inspection meeting are selected. This selection is done by the moderator. In this step, moderator also takes important inspection material from the author and distributes it to the participants.

Overview Meeting facilitates the moderator in the sense that he uses this meeting to introduce important features of the inspection to the participants.

Preparation allows each member of the team to review its assigned work product. Defects and issues are noted down by him that he will raise during the inspection meeting.

Inspection Meeting is led by the moderator. Recorder records all the defects and open issues that will be raised during the inspection meeting.

Casual Analysis: In this activity, we try to take long term benefits of the inspection process. Casual analysis help us find ways to improve software quality assurance.

Rework: In this activity, issues are settled that were highlighted during the inspection meeting. Issues that couldn’t be resolved are held until further meeting or as decided by the moderator.

Follow-up is done to make sure that all the defects that were reported during the inspection meeting have been fixed. If same has been achieved, the inspection meeting is finished.

GUIDING PRINCIPLES FOR INSPECTIONS

For successful software inspections, following principles must be followed. [8]

- Ego should not be brought in inspection meeting. Since producers of the work-product remain an active part of the meeting, it’s difficult to put your own work for open criticism. Being a team member, everyone should strive for finding defects before the product is delivered to the customer.
- Criticize the product and not the producer. If someone will start criticizing the producer, then...
it will lead to unmanageable behavior during the meeting and the purpose of the meeting will be destroyed.

- Try to find defects during the meeting and never discuss the ways how to fix them.
- Inspection meeting must not last more than two hours as tiredness and boredom becomes obvious after two hours. Long duration will fail the purpose of the inspection meeting.
- Style issues should not be discussed. Style issues could be that someone gives two spaces for indenting the code and someone gives three spaces. These should never be discussed as they are not defects.

INSPECTIONS: BENEFIT OR DAMAGE?

Software inspections have a key benefit that they can be applied earlier in the software project unlike other quality assurance attributes. Since purpose of inspection is to find defects in the work product as early as possible, therefore, earlier defect recovery leads to low cost. Also, software inspections is that quality assurance metric that can be applied on all software development phases i.e., we can apply software inspections on design, requirements, coding, testing, etc. etc. Historical data shows that inspections never lead to negative impact on software products rather high quality software is produced when inspections are used. Figure 3 gives a comparison of the situations when inspections were not used and when inspections were used during software development.

![Figure 3 Comparison - Inspections vs No Inspections](image)

Table 1 Comparison of Errors

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Inspection</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module interface errors</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Excessive code complexity</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Unrequired functionality</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability problems</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Performance problems</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Badly structured code</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Failure to meet</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boundary value errors</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

From above table, it is clear that more errors can be figured out by using Inspections rather than Testing.

If we take a look at Figure 4 (adopted from [9]), then we can clearly see when inspections are not performed then cost to fix defects rises abnormally, whereas when inspections are performed then cost is reduced. Another important thing to be noted here is that when inspections are used, then time to develop the software system is also reduced.
Now, we take a look at the cost of fixing defects in testing phase and compare this with cost of fixing defects with inspections. To do a comparison, we take a look at the historical data of companies. Table 2 shows that data.

<table>
<thead>
<tr>
<th>Company</th>
<th>Inspections Cost</th>
<th>Test Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T</td>
<td>1.40 hrs/defect</td>
<td>8.50 hrs/defect</td>
</tr>
<tr>
<td>JPL</td>
<td>$110/defect</td>
<td>$1715/defect</td>
</tr>
<tr>
<td>IBM</td>
<td>01 unit of cost</td>
<td>09 times more cost</td>
</tr>
<tr>
<td>IBM</td>
<td>$47/defect</td>
<td>$60-$1230/defect</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>01 unit of cost</td>
<td>19 units of cost</td>
</tr>
<tr>
<td>ICL</td>
<td>1.2~1.6 hrs/defect</td>
<td>8.39 hrs/defect</td>
</tr>
<tr>
<td>Shell</td>
<td>01 unit of cost</td>
<td>29 units of cost</td>
</tr>
<tr>
<td>Thorn EMI</td>
<td>01 unit of cost</td>
<td>7~25 units of cost</td>
</tr>
<tr>
<td>Applicon, Inc.</td>
<td>01 hour</td>
<td>--</td>
</tr>
<tr>
<td>Infosys</td>
<td>01 unit of cost</td>
<td>3 – 6 units of cost</td>
</tr>
</tbody>
</table>

Table 2: Comparison of Inspection cost and testing cost

From the table, we see that inspection cost is much less as compared to cost of fixing defects later in the testing stage.

In this section, we looked at the comparison between software testing and software inspections. We concluded that software inspections outweigh the benefits gained from any other quality assurance technique.

VII. WHY COMPANIES DON’T PERFORM INSPECTIONS?

In the preceding sections, we looked how inspections overweight the advantages when they are used as compared to the situations when we don’t use them. Even with these advantages, not too many companies use it. In this section, we take a look at the questions that companies put forward that why they don’t use inspections for their software projects. We will also try to answer their questions. Some questions/statements that such companies put forward are:

- Inspections are not easy
- Inspection is an unenjoyable task for software engineers
- They are labor-intensive
- Inspections do take about 5% to 15% budget. However, this cost is reduced when seen in terms that when defects will be uncovered during later stages, then they will add more to the cost. Inspections will definitely take money in the beginning, but when a company will be used to inspections, it will realize that inspections save money.
- When something new is adopted, it is difficult. Inspections will be difficult for companies who never used it. But when they will start using inspections in their every software project, they will become expert. Also, new task seems to be boring in the beginning and also requires more work. When companies will become expert in software inspections, then boredom will be diminished.

VIII. CONCLUSION

Software development is becoming a monster with the passage of time because software systems are becoming more complex and unmanageable. To tackle down this monster, we looked how quality assurance attributes can contribute to the quality of software system. We gave analysis and showed how software inspections can be used to build high quality software system. We concluded that those organizations that want to stay in the software industry must take inspections seriously and adopt it in their software projects.

REFERENCES:


[7] National Aeronautics and Space Administration, Washington, DC 20546, SOFTWARE FORMAL INSPECTIONS GUIDEBOOK

[8] Karl E. Wiegers, Improving Quality Through Software Inspections

[9] Roger S. Pressmann, Software Engineering – A Practitioner’s Approach, (pp. 422)