# TEACHING SOFTWARE REQUIREMENTS INSPECTIONS TO SOFTWARE ENGINEERING STUDENTS THROUGH PRACTICAL TRAINING AND REFLECTION

Anurag Goswami, Gursimran Singh Walia
Computer Science Department
North Dakota State University

## Introduction

There is a growing demand for software developers that is expected to grow even more in the coming years [1,2] It is important to ensure that students graduating are prepared for their future careers in the software industry. However, multiple researchers have reported that software engineering (SE) graduates lack the necessary skills or abilities to find employment in the software industry [3-5]. For example, Simmons et al., reported that students are not familiar with software development processes when beginning their jobs in industry and that curricula should put more emphasis on requirement gathering and elicitation techniques [6]. Additionally, an extensive systematic literature review empirically evaluated knowledge deficiency of graduating CS/SE students and revealed "Software Engineering Practices" (e.g., requirements, life cycle, and quality assurance) as a major category [7].

In software industries, requirements development is one of the earliest phases of Software Development Life Cycle (SDLC). It is a critical phase where software requirements are gathered from different stakeholders (both technical and non-technical), and written using Natural Language (NL) in a formal document known as Software Requirements Specification (SRS) [8]. Due to the inherent nature of NL (i.e. ambiguity, imprecision, and vagueness) [9], faults are committed during the development of SRS. Therefore, industries are focused on detecting and fixing faults at early phases of SDLC to avoid any rework effort and costs to fix faults at later stages of development [10]. To do so, software inspections [11], are used wherein skilled individuals review a software artifact to find and report faults.

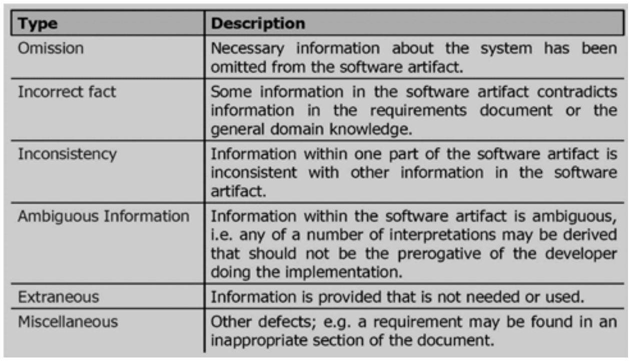
Software industries (e.g., Microsoft) routinely provides inspection training to educate their developers about the process, importance, and benefits of inspections. Due to the importance of inspections in software industry (i.e. to save rework cost, effort, and time), academia should also prioritize training students with early quality assurance skills (i.e. inspections) during SDLC. Therefore, this research reports the results from a practical training experience to help students improve their understanding of inspection which in turn, would improve their inspection performance. This paper presents results of an academic study on the effect of reflection (training) technique on thirteen graduate and twenty-six undergraduate students on their inspection performance. The participants individually inspected two different requirement documents using fault-checklist method and recorded faults pre and post reflection. We analyzed the impact of reflection by calculating individual pre and post reflection inspection performance and by taking class average for undergraduate and graduate students. The results show that post reflection, inspection understanding and performance increases for both undergraduate and graduate students.

## Background

This section describes the fault-checklist based inspection technique and its steps along with various other fault detection techniques that are used to detect and report faults.

Inspection, as described by Fagan [12], is a systematic technique to examine a software artifact in detail. Evidence showed the benefits of inspection on artifacts developed at different phases of software development (e.g., requirement, design, code, interfaces) [13]. Inspections takes place in different steps which involves: a*) Selecting skilled individuals/ inspectors, b) Individual review to find faults, c) Team meeting to consolidate faults, d) Follow-up and repair*.

There are many variations on Fagan's original concepts [14,15] that emphasize different parts of the process (e.g. placing more emphasis on the individual preparation phase and less emphasis on the team meeting phase). Regardless of whether there is a team meeting, the effectiveness of the individuals significantly impacts the overall effectiveness of the inspection [16]. During the use of inspection technique, inspectors are given a set of checklists and printed form which guides them on how to perform inspection [17]. Based on the knowledge from checklist regarding the type of faults (example in Figure 1), inspectors read through the software artifact (here requirements document) to detect and report faults in fault form.



| Type | Description |
|---|---|
| Omission | Necessary information about the system has been omitted from the software artifact. |
| Incorrect fact | Some information in the software artifact contradicts information in the requirements document or the general domain knowledge. |
| Inconsistency | Information within one part of the software artifact is inconsistent with other information in the software artifact. |
| Ambiguous Information | Information within the software artifact is ambiguous, i.e. any of a number of interpretations may be derived that should not be the prerogative of the developer doing the implementation. |
| Extraneous | Information is provided that is not needed or used. |
| Miscellaneous | Other defects; e.g. a requirement may be found in an inappropriate section of the document. |

Figure 1. Fault types in the fault checklist form.

## Experiment Design

The goal of this study was to investigate whether experiential learning of reviewing software artifact aided by an individual's reflection on their inspection results can lead to an improved understanding of requirements inspection process and an improvement in their abilities to find real software faults during the inspection. To accomplish this goal, a controlled empirical study involved undergraduate and graduate students enrolled in SE courses at North Dakota State University (NDSU). During the course, the subjects performed inspections of two industrial strength software requirement artifacts that were seeded with real software defects. For first inspection, students were trained on fault checklist based requirements inspection. The students then performed an individual inspection of requirements document and reported the faults found during the inspection. Next, the subjects were provided an actual list of seeded faults and were asked to read through the fault descriptions and reflect upon the faults found (reported) and missed during the inspection. For second inspection cycle, each participant performed an individual inspection using the same fault checklist on a different requirements document and reported faults. The fault data (e.g., reported and missed faults, true faults and false positives, fault descriptions) were collected and analyzed pre and post reflection to understand the nature of improvement in their inspection abilities and their understanding of the inspection process. The remainder of the section provides details of the study goals and metrics, requirement artifacts, students, and study procedure in the following sub sections.

*Research Goal:* The major goal of this study was to investigate whether students' understanding of requirement faults and their inspection performance improve after hands-on practice and reflection. Our research questions are postulated in the GQ format [18] (Figure 2) and briefly described below.

RQ1 investigates whether the students are able to detect a larger number of faults post reflection (i.e., during the second inspection)? RQ2 investigates whether students' find faults faster (i.e., increase in fault rate) during the second inspection? Finally, RQ3 evaluates the quality of the description of the faults reported by the students' pre and post reflection? RQ4 investigates the improvement in terms of the percentage of true faults vs. fault positives post reflection?

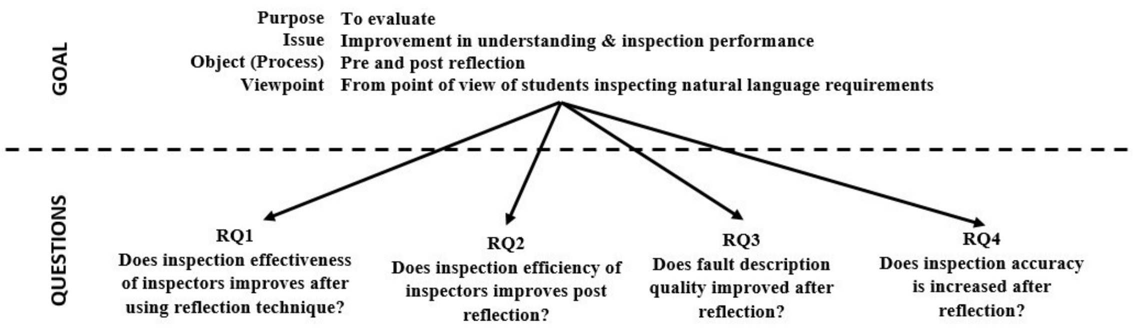

GOAL
Purpose To evaluate
Issue Improvement in understanding & inspection performance
Object (Process) Pre and post reflection
Viewpoint From point of view of students inspecting natural language requirements
QUESTIONS
RQ1
Does inspection effectiveness of inspectors improves after using reflection technique?
RQ2
Does inspection efficiency of inspectors improves post reflection?
RQ3
Does fault description quality improved after reflection?
RQ4
Does inspection accuracy is increased after reflection?

Figure 2. Research questions in Goal-Question (GQ) format.

*Participating Subjects:* Twenty six under-graduate students enrolled in System Analysis and Design course along with thirteen graduate students enrolled in Requirements Engineering course at NDSU participated in this study. System Analysis and Design course covers the requirements and design development and the required skills for planning, analysis, and design of software system. Similarly, Requirements Engineering course focuses especially on requirement development tasks and technique along with requirement inspection technique. Both the courses required the students to learn about software inspections and their impact on the software quality improvement. Students in both the courses had an average of two years of software development experience in past (i.e. classroom projects, assignments, and industry).

*Artifact:* Two externally developed industrial strength requirement documents (Table I), Loan Arranger System (LAS) and Parking Garage Control System (PGCS), were inspected by each participant during two inspection cycles. Both the documents were written in plain English, developed by Microsoft developers, and have been used in several inspection studies (for comparing different inspection techniques) as well as by Microsoft to train their employees on the inspection process at Microsoft [19,20]. In terms of the length of documents, LAS was 11 pages long seeded with 30 realistic faults whereas PGCS was 14 pages long seeded with 34 faults. The seeding of the faults was done by Microsoft researchers to represent realistic faults committed by Microsoft developers. Both the documents were selected because both came from the same organization (Microsoft) and were similar in size and fault density (i.e. 2.72 and 2.42 faults per page for LAS and PGCS respectively).

*Experiment Procedure:* The experiment steps are described below and shown in Figure 3:

*Training 1 - Training on inspecting SRS for faults:* During this step, students in both classes were trained by the same instructor during an in-class session of 70 minutes on how to use fault- checklist technique to detect and report different types of faults in SRS in a fault list. During the training, students were provided a small subset of requirements for a Gas Station Control System (GSCS) and were asked to

Table I. Artifacts used for inspection.

| Inspection Cycle | Artifact name | Description | Number of seeded faults | Number of inspectors |
|---|---|---|---|---|
| 1 – Pre- reflection | Loan Arranger System (LAS) | Online system for loan bundling based on user characteristics | 30 | 26 (Undergrad) 13 (Grad) |
| 2- Post- reflection | Parking Garage Control System (PGCS) | Provides automated entry and exit of vehicles based on card/ticket | 34 | 26 (Undergrad) 13 (Grad) |

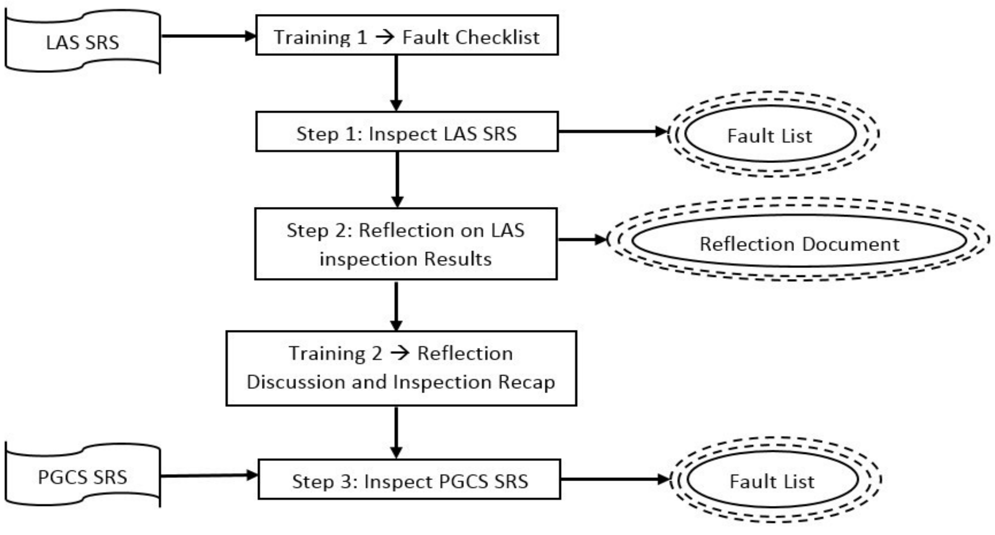

LAS SRS
Training 1 → Fault Checklist
Step 1: Inspect LAS SRS
Fault List
Step 2: Reflection on LAS inspection Results
Reflection Document
Training 2 → Reflection Discussion and Inspection Recap
PGCS SRS
Step 3: Inspect PGCS SRS
Fault List

Figure 3. Experiment Procedure.

find faults which were then discussed in class to prepare them for the first inspection cycle.

*Step 1 – First inspection: Inspecting LAS SRS for faults:* Next, the subjects individually inspected the LAS document (that was handed to each participant) using the fault checklist technique and reported faults along with timestamp when each fault was found. In addition, the fault reporting form required the subjects to classify the faults identified during the inspection into one of the following fault types: *Omission* (O), *Ambiguous Information* (A), *Inconsistent Information* (II), *Incorrect Fact* (IF), *Extraneous* (E), and *Miscellaneous* (M). At the end of the inspections, thirty nine fault lists (from 26 undergraduate and 13 graduate students) were collected for analysis.

*Step 2 – Reflection of LAS inspection results:* One of the researchers evaluated the faults reported by each participant in both the courses and provided them feedback about true faults and false positives. Students were also informed of the faults that lacked a clear and precise description of why (and where) it represented a problem in the requirements. Next, post inspection reflection document (sample in Table II.) was performed in-class wherein, participants were handed a complete list of the 30 faults in the LAS document. Students were asked to read through the actual fault descriptions and to comment on whether they agree (and explain if they disagree) on the fault? Whether they were able to find that fault? and if they were able to report them? Reading through the first row of Table II, each column is described as follows:

- *Defect#:* represents the defect ID in seeded fault list.
- *Req.#:* indicates the requirement ID(s) where fault is present.
- *Type:* denotes students about fault category to which a fault belongs. For example, A in the first row represents an ambiguity (A) in the requirements.
- *Description:* explains the fault in enough detail for readers to understand.
- *Is it a defect?:* this column required students to agree or disagree whether the fault described represents an actual requirement problem.
- *Did you see this?:* students reported whether they were able to see this fault (in the form of 'yes' or 'no') during the inspection.
- *Did you report this?:* students reported (in the form of 'yes' or 'no') whether they reported this fault during inspection of LAS document.
- *Explain:* this column needs a brief description by the students about their response.

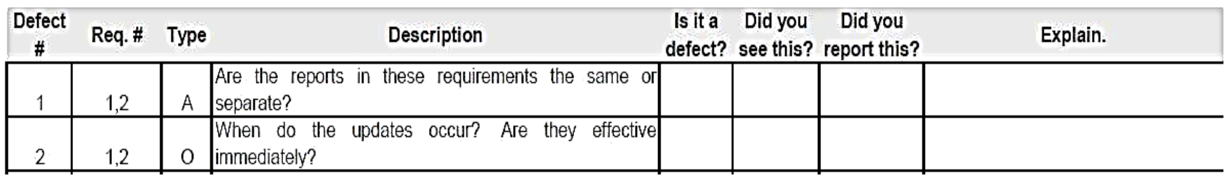
The goal of the reflection document was to enable participants to gain insights in the inspection process and to help them reflect on the reasons behind the faults they missed or they saw but not reported in their fault list. The students were also told to read through the fault descriptions to be able to improve their fault report quality.

*Training 2 – Reflection discussion and recap for re-inspection:* The students were asked to discuss any doubts in the reflection of faults with the trainer and were given a quick recap of fault-checklist based inspection technique.

*Step 3 – Inspecting PGCS requirements:* Next, each participant received the second PGCS document along with the fault form (that they had used during the first inspection) and were asked to perform an individual inspection to identify and record faults based on the feedback from reflection. Like the first inspection, participants were required to mention start and end time of inspection along with the timestamp when they found each fault and to classify the faults into fault types. At the end of the inspections, thirty-nine new fault lists (one per student) were collected for analysis.

## Data Collection

This section describes the raw data collected during the study along with the data that was computed from



| Defect # | Req. # | Type | Description | Is it a defect? | Did you see this? | Did you report this? | Explain. |
|---|---|---|---|---|---|---|---|
| 1 | 1,2 | A | Are the reports in these requirements the same or separate? | | | | |
| 2 | 1,2 | O | When do the updates occur? Are they effective immediately? | | | | |

Table II. Sample of reflection form for LAS document.

raw data to calculate inspection performance (shown in Figure 4) for each participant for both requirement documents (i.e. LAS and PGCS). The raw data variables are described below:

- *M1: Total faults ($T_{sf}$):* denotes the number of total faults seeded in the requirements document. In this experiment, LAS contained 30 and PGCS had 34 seeded faults.
- *M2: Total number of faults reported ($T_f$):* denotes the total number of faults (i.e. count of all faults reported) reported by the subjects in their fault reporting form. This is the raw count prior to any evaluation of the correctness of the faults reported by the subjects. This was done to compare the true and false positive counts when comparing pre and post reflection results.
- *M3: Inspection time ($I_t$)* - is the measure of total time (in minutes) taken by each participant to perform the inspection of an SRS document. M3 was calculated by comparing the starting and finish times for each participant and for each inspection cycle.

Below are the calculated variables from raw inspection data described above:

- *M4: Total number of false positives ($T_{fp}$):* one of the researcher's read through fault list of each participant to identify the number of false positives

- *M5: Inspection effectiveness ($T_e$)* - after removing fault positives ($T_{fp}$) from the total fault count ($T_f$), the number of actual faults for each participant was calculated. This was computed pre and post reflection to evaluate the improvement in their inspection accuracy (discussed next). $T_e = T_f - T_{fp}$
- *M6: Inspection Accuracy ($I_a$)* – is measured as the percentage of inspection effectiveness ($T_e$) in terms of the total fault count ($T_{sf}$). Inspection accuracy was computed pre and post reflection as; $I_a = (T_e/T_{sf})*100$
- *M7: Inspection efficiency ($I_e$)* - measured as the total number of faults ($T_e$) found per hour. This was done to evaluate if the subjects were able to find faults faster post reflection and computed as: $I_e = (T_e/I_t)*60$
- *M8: Fault description score ($FD_s$) and M9: Fault description quality ($Q_{fd}$)* — for each inspector, it is the summation of binary score of 0 (not well described) or 1 (well described) for a fault description of each fault out of total faults detected. The idea behind is that, the author of the document should be able to understand and correct the faults without discussing with the inspector(s). Using the same criterion, one of the researcher read through the fault descriptions to understand clearly where fault occurred in SRS and why it represented a problem without talking to the inspectors. If a fault was well described, then it was marked as 1 otherwise 0. For example, if out of total 20 faults, only 10 faults were described in well understood form (i.e. with a score of 1 for 10 faults and a score of 0 for other 10); then the fault description score will be 10 for that particular inspector. This was done to calculate M9 of each fault list pre and post reflection. For each inspector and each

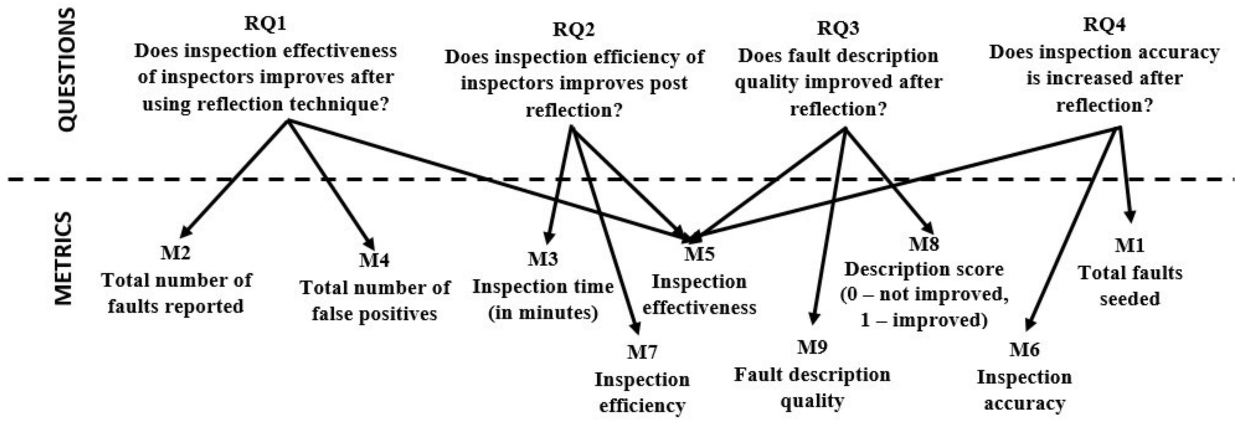

QUESTIONS
RQ1
Does inspection effectiveness of inspectors improves after using reflection technique?
RQ2
Does inspection efficiency of inspectors improves post reflection?
RQ3
Does fault description quality improved after reflection?
RQ4
Does inspection accuracy is increased after reflection?
METRICS
M2
Total number of faults reported
M4
Total number of false positives
M3
Inspection time (in minutes)
M5
Inspection effectiveness
M8
Description score (0 – not improved, 1 – improved)
M1
Total faults seeded
M7
Inspection efficiency
M9
Fault description quality
M6
Inspection accuracy

Figure 4. Research questions along with various metrics used.

inspection cycle, M9 was measured as the percentage of faults that are described in a well understood form out of total inspection effectiveness (Te). Qfd = (FDs/Te)*100

For example, out of total 20 faults (i.e Te) fault description score is 10 (FDs) then the fault description quality will be: (10/20)*100 = 50%.

We compared the average score of inspection performance from metrics (described above) during the first inspection (i.e. pre reflection) vs. during the second inspection (i.e., post reflection) for participants in both the courses to evaluate the improvement in the students' inspection performance. Table III represents a sample pre and post inspection data for one student. The columns are arranged (from left to right) in the same fashion as metrics are described above. Based on the data from one student, during the second inspection, he/she reported fewer total faults (10 vs. 17), spent less time to find those faults (i.e., 25 minutes vs. 70 minutes), yet found more true faults (4 vs. 2), and reported less fault positives (6 vs. 15). Inspection effectiveness, accuracy and efficiency and fault descriptions improved visibly after training and reflection. The next section analyzes whether similar patterns were seen across all the subjects.

## Analysis and Results

This section reports the improvement in the understanding of requirements inspections and fault detection abilities of the students from first to second inspection cycle. The results are organized around the four research questions (see Figure 4):

### *1) RQ1: Does inspection effectiveness of inspectors improves after using reflection technique?*

To provide an overview of the effectiveness results, students were able to find a larger number of true faults (Te) during the second inspection (PGCS) as compared to the first inspection (LAS document). Figure 5 compares the average inspection effectiveness (solid fill for graduate students and pattern fill for undergraduate students) pre (using LAS document) and post (PGCS document) reflection. The results show that, graduate students found an average of 4.85 faults during the second inspection (vs. 4.23 faults during the first inspection) and undergraduate students found an average of 5.04 faults (vs. 4.35 faults) during the second inspection. These results show that, effectiveness (the number of actual faults detected) during inspections increased for both graduate and undergraduate students. This was consistent across all the subjects. Additionally, the increase was larger for the undergraduate students which could have been due to the size effect (i.e., larger number of students). The results from paired samples t-test (p=0.49 for graduates and p=0.16 for undergraduate students) showed that the effectiveness increase was not statistically significant. Therefore, based on this result, while the experiential learning (and reflection) helped students detect a larger number of faults, the increase was not significant.

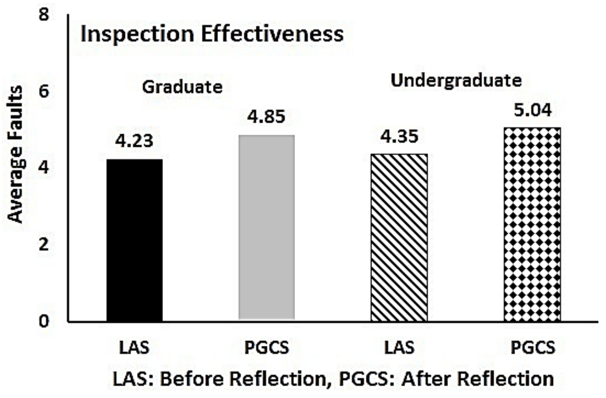
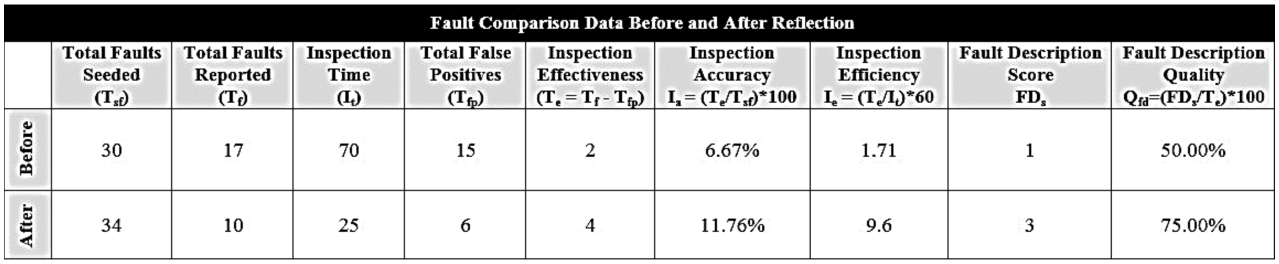


Figure 5. Comparison of inspection effectiveness before and after reflection.



| Fault Comparison Data Before and After Reflection | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Total Faults Seeded ($T_{sf}$) | Total Faults Reported ($T_f$) | Inspection Time ($I_t$) | Total False Positives ($T_{fp}$) | Inspection Effectiveness ($T_e = T_f - T_{fp}$) | Inspection Accuracy $I_a = (T_e/T_{sf})*100$ | Inspection Efficiency $I_e = (T_e/I_t)*60$ | Fault Description Score $FD_s$ | Fault Description Quality $Q_{fd}=(FD_s/T_e)*100$ |
| Before | 30 | 17 | 70 | 15 | 2 | 6.67% | 1.71 | 1 | 50.00% |
| After | 34 | 10 | 25 | 6 | 4 | 11.76% | 9.6 | 3 | 75.00% |

Table III. Sample data of one inspector before and after reflection.

*2) RQ2: Does the inspection efficiency is increased after reflection?*

This research question compares the rate at which students found faults (i.e. inspection efficiency – Ie) during the first and second inspection. Figure 6 shows the average inspection efficiency of graduate (solid fill) and undergraduate students (pattern fill) pre and post reflection. Results from Figure 6 shows that post reflection (i.e. during the second inspection), students found faults faster as compared to the first inspection. The results from a paired samples t-test showed that inspection efficiency significantly improved for both graduate (p=0.004) and undergraduate (p<0.001) students post reflection. This is a significant results and signify that, the students' learning curve was significantly enhanced after having performed an inspection and reflecting upon their mistakes and the fault they should have found.

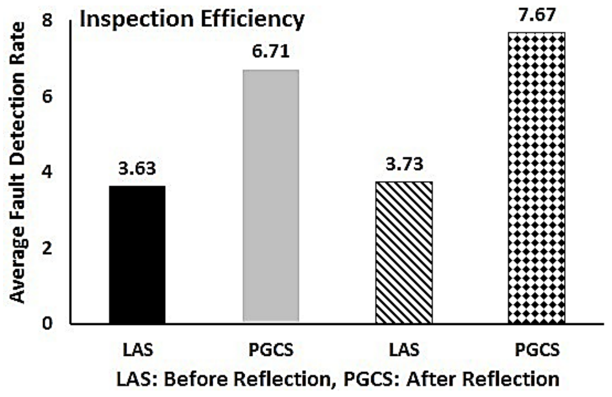


Figure 6. Comparison of inspection efficiency among graduate and undergraduate students before and after reflection.

*3) RQ3: Does fault description quality improves after reflection?*

This research question investigates whether inspectors described faults more clearly in fault reporting form during inspection after reading through the clear descriptions in the reflection document. Table IV is an example of fault description of one of the inspectors before and after reflection process. As seen in Table IV, the descriptions are structured in more understandable manner while still being concise post inspection. To quantify the description quality, we compared the average *Fault description score for both courses pre and post reflection (Figure 7).*

| Fault Description Quality | |
|---|---|
| Before Reflection | After Reflection |
| 6 loan analysts not 4 | Not every driver needs a ticket (only non-reserved) |

Table IV. Example of fault description quality before and after reflection technique.

Figure 7 compares the fault description quality among graduate (solid fill) and undergraduate students (pattern fill) as requirements inspectors before and after reflection. The results in Figure 7 shows that fault description quality of both graduate and undergraduate students increased after they went through reflection technique. To evaluate the statistical significance, we performed paired samples t-test which showed that reflection had a strong and significant impact on the fault description quality for both graduate (p=0.003) and undergraduate (p=0.004) students. Therefore, the experiential learning help students report more clear and understandable fault descriptions.

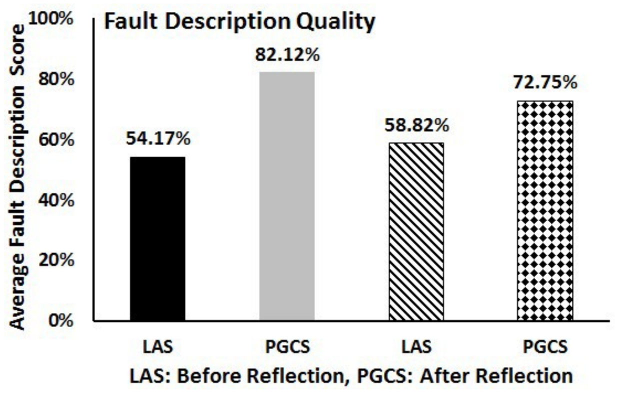


Figure 7. Comparison of fault description quality score among graduate and undergraduate students before and after reflection.

*4) RQ4: Does inspection accuracy of inspectors improves post reflection?*

As mentioned earlier, inspection accuracy (calculated as the percentage effectiveness out of total number of seeded faults) was compared during the two inspection cycles. The percentage was computed to normalize the comparison between two documents that had a different number of seeded faults. A comparison of the inspection accuracy is shown in Figure 8. The results show that, students reported higher inspection accuracy post reflection as compared to the first inspection. Yet again, the increase was higher for the undergraduates as compared to the graduate students.

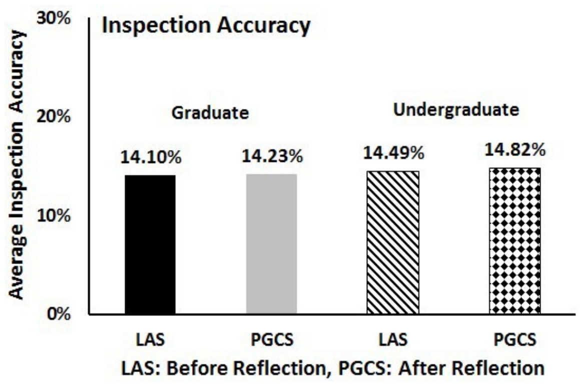


Figure 8. Comparison of inspection accuracy before and after reflection.

To gain more insights into the accuracy results, we calculated the percentage of inspection false positive data from pre and post reflection. It was the ratio of false positives (Tfp) to the number faults seeded (Tsf). It was found that, students reported a large number of false positives during the second inspection which impacted their results. This could have been biased by a couple of reasons. First, the students were told that they would be graded on their performance during the second inspection (since they have already done it once and have had a chance to review their mistakes). This might have negatively motivated them to report as many faults as possible to show that their effort during the inspection exercise. Second, the students still tend to think a lot in terms of the missing design details (which is outside the scope of functional requirements) and that needs to be talked more in class for them to be able to differentiate between true faults and false positives.

## Discussion of Results

The major focus of this study was to investigate whether experiential learning aided by the reflection technique can lead to a better understanding of requirements inspection which in turn, leads to an improved inspection performance. Based on the results, it is evident that reflection technique helped students understand the inspection process better which leads to an improved inspection outcome (i.e. effectiveness, efficiency, and description quality).

Inspection accuracy was almost equal for both graduate and undergraduate students which might be due to the fact that students learn more on how to design and code and not enough time is spent on helping students to read or write functional requirements. This makes it difficult for students to differentiate between missing information or ambiguity in the requirements description (a type of requirements fault) and missing design information (often outside the scope of requirements) during the review of information contained in the SRS. This was a big reason that the students still report (even post reflection) a larger frequency of false positive faults. Interestingly, undergraduate students performed better than the graduate students in terms of their inspection performance both before and after reflection. This is in accordance with the studies [21,22] at Microsoft, wherein level of the technical education (Bachelors vs. Masters vs. Doctorate) did not had a significant impact on the inspection performance of professional developers. Therefore, unlike other aspects of software development, inspections may rely more on the inherent abilities of the students to comprehend and process natural language information contained in requirements document. We plan to evaluate this aspect in future studies in hopes of further improving the performance of students learning software inspections in classroom settings.

## Conclusion and Future Work

Based on the results from our study, reflection techniques do help students in better understanding of fault-checklist based requirements inspection technique and can lead to higher inspection output. Results also exhibits that, reflection technique can be used by academicians for reducing skill gap between academia and industry by helping students acquire the required inspection skills in experiential form. While this paper reports the use of experiential learning in the context of teaching requirement inspections to the students, it can be used for training other needed software skills (e.g., writing quality code, developing requirements/design document, etc). These results motivate us for further investigation. Our immediate future work would include replicating the study for non-technical inspectors for generalizing our results. Another future work is how students' cognitive ability to comprehend information could have an impact on software development task(s).

## References

1. Samson, T.: 'Demand for software engineers keeps climbing -- and so do the salaries', InfoWorld, 2015.

2. Sayed, D.: 'Technology pay rates rising faster than the general labor market', Applied HR Strategies (AHRS) Client Alert, 2015.

3. Begel, A., and Simon, B.: ‘Struggles of new college graduates in their first software development job’. Proc. ACM SIGCSE Bulletin2008.

4. Haddad, H.: ‘Post-graduate assessment of CS students: experience and position paper’, Journal of Computing Sciences in Colleges, 2002, 18, (2), pp. 189-197.

5. Radermacher, A., Walia, G., and Knudson, D.: ‘Missed Expectations: Where CS Students Fall Short in the Software Industry’, CrossTalk Mag.-J. Def. Softw. Eng., no. Jan/Feb, 2015, pp. 4-8.

6. Simmons, C.B., and Simmons, L.L.: ‘Gaps in the computer science curriculum: an exploratory study of industry professionals’, Journal of Computing Sciences in Colleges, 2010, 25, (5), pp. 60-65.

7. Radermacher, A., and Walia, G.: ‘Gaps between industry expectations and the abilities of graduates’. Proceeding of the 44th ACM technical symposium on Computer science education2013.

8. Goswami, A., and Walia, G.: ‘An empirical study of the effect of learning styles on the faults found during the software requirements inspection’. Proc. Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on, Pasadena, CA, 4-7 Nov. 2013.

9. Berry, D.M.: ‘Ambiguity in natural language requirements documents’: ‘Innovations for Requirement Analysis. From Stakeholders’ Needs to Formal Designs’ (Springer, 2008), pp. 1-7.

10. Perry, W.E.: ‘Effective Methods for Software Testing: Includes Complete Guidelines, Checklists, and Templates’ (John Wiley & Sons, 2006).

11. Ackerman, A.F., Buchwald, L.S., and Lewski, F.H.: ‘Software inspections: an effective verification process’, Software, IEEE, 1989, 6, (3), pp. 31-36.

12. Fagan, M.E.: ‘Advances in software inspections’: ‘Pioneers and Their Contributions to Software Engineering’ (Springer, 2001), pp. 335-360.

13. Fagan, M.E.: ‘Design and code inspections to reduce errors in program development’: ‘Pioneers and Their Contributions to Software Engineering’ (Springer, 2001), pp. 301-334.

14. Martin, J., and Tsai, W.T.: ‘N-fold inspection: A requirements analysis technique’, Communications of the ACM, 1990, 33, (2), pp. 225-232.

15. Parnas, D.L., and Weiss, D.M.: ‘Active design reviews: principles and practices’, Journal of Systems and Software, 1987, 7, (4), pp. 259-265.

16. Porter, A., Siy, H., Mockus, A., and Votta, L.: ‘Understanding the sources of variation in software inspections’, ACM Transactions on Software Engineering and Methodology (TOSEM), 1998, 7, (1), pp. 41-79.

17. Parnas, D.L., and Lawford, M.: ‘The role of inspection in software quality assurance’, Software Engineering, IEEE Transactions on, 2003, 29, (8), pp. 674-676.

18. Van Solingen, R., Basili, V., Caldiera, G., and Rombach, H.D.: ‘Goal question metric (gqm) approach’, Encyclopedia of Software Engineering, 2002.

19. Shull, F., Carver, J., and Travassos, G.H.: ‘An empirical methodology for introducing software processes’, ACM SIGSOFT Software Engineering Notes, 2001, 26, (5), pp. 288-296.

20. Carver, J., Shull, F., and Basili, V.: ‘Observational studies to accelerate process experience in classroom studies: an evaluation’. Proc. Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on 2003.

21. Carver, J.: ‘The impact of background and experience on software inspections’, Empirical Software Engineering, 2004, 9, (3), pp. 259-262.

22. Carver, J.C., Nagappan, N., and Page, A.: ‘The impact of educational background on the effectiveness of requirements inspections: An empirical study’, Software Engineering, IEEE Transactions on, 2008, 34, (6), pp. 800-812

**Biographical Information**

Anurag Goswami is a Ph.D. Candidate in the department of Computer Science at North Dakota State University. His main research interests include empirical software engineering, human factors in software engineering, and software quality.

Gursimran S. Walia is an associate professor of Computer Science at North Dakota State University. His main research interests include empirical software engineering, software engineering education, human factors in software engineering, and software quality. He is a member of the IEEE Computer Society. Contact him at gursimran.walia@ndsu.edu