EFFECTIVENESS OF HUMAN ERROR ABSTRACTION ASSIST TOOL AT IMPROVING THE QUALITY OF SOFTWARE REQUIREMENTS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Navneet Deosi

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

January 2020

Fargo, North Dakota

North Dakota State University

Graduate School

Title

EFFECTIVENESS OF HUMAN ERROR ABSTRACTION ASSIST TOOL AT IMPROVING THE QUALITY OF SOFTWARE REQUIREMENTS

By

Navneet Deosi

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia

Chair

Dr. Simone Ludwig

Dr. Limin Zhang

Approved:

January 13, 2019 — Date

Dr. Kendall Nygard — Department Chair

ABSTRACT

The successful completion of any given software project is dependent on error-free Software Requirement Specification (SRS) documents. SRS documents are prone to human errors and faults because their creation involves a large amount of human interaction. This paper reports an experiment study conducted to gauge the accuracy with which the users identify the root causes of software faults in external and self-developed SRS documents. This study primarily focuses on the effective use of the Human Error Abstraction Assist (HEAA) as a tool to help users abstract human errors. HEAA was chosen because it takes all real-life mishaps into consideration and maps them to the different human error classes – slips, lapses, and mistakes. The data collected from this study reports the students' performance in terms of accuracy and compares it with the results collected from a past study conducted with industry practitioners that also used the HEAA tool for abstracting errors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The process of software engineering facilitates the design, development, and maintenance of software programs or applications. There are several methodologies called the Software Development Life Cycle (SDLC) models that can be adopted to build software programs. Each SDLC model consists several different activities that guides a software project from the initiation phase to the completion phase [1, 2]. Building software applications is a creative process, but it also involves various challenges. One of the biggest challenges faced during software construction is the eagerness with which the software developers dive right into the project and begin coding without having a clear understanding of what needs to be done. The developers often argue that the project's scope and requirements would become clear with the initiation of the coding process. It is because they feel that the stakeholders of the project would be able to have a better understanding of their needs by seeing the initial workings of a software. However, these arguments are not completely true. It is important to emphasize on Requirements Engineering (RE) and its activities to avoid major software failures. Research has shown that the software developers spend around 80% of their time trying to fix those problems that should have been corrected in the phases prior to the initiation of software development [4, 16].

Requirements Engineering (RE) is the process of collecting, defining, documenting as well as maintaining all system requirements collected for a given project from its stakeholders. It is amongst one of the most important steps in any given Software Development Life Cycle (SDLC) model [7]. RE comprises of several different activities like – Requirements Elicitation, Requirements Specification, Requirements Verification and Validation, and Requirements Management [7]. Like most other software engineering activities, RE too must adapt to the project which is being worked on and on the scope of that project. Since, RE involves a large

amount of interaction between different stakeholders, this process is more prone to the insertion of faults and errors [10]. This paper analyzes how faults are injected into Software Requirement Specification (SRS) documents. It also aims at abstracting human errors from those faults with the help of the Human Error Abstraction Assist (HEAA). HEAA takes into consideration all real life mishaps and maps all errors to the appropriate human error classes of – slips, lapses, mistakes. It also uses a decision-tree based model for easy navigation and use of the tool [10, 11].

Requirements make up the key factors in the development of any software engineering project. It is a clear, concise, and specific set of instructions that are independent in nature. When two or more people read a requirement, they should all be able to interpret it in the same way. Requirements also need to be verifiable and testable in nature so that all required changes can be easily incorporated. Requirements describe the true business problem or the need. In other words, it describes what is needed and why is it needed? They are driven by the business needs and it is never considered as a complete solution.

Faults are often injected into the Software Requirements Specification (SRS) documents during the RE activities [8, 12]. These injected faults are identified by the inspectors reviewing the requirements documents. However, there are times when faults remain undetected, leading to critical system failures. Thus, a basic understanding of human error phycology is necessary, and it is used often to detect these faults and their underlying errors to help prevent major software defects [5].

Human Errors

Human errors (errors) are defined as the root cause of a specific problem. These are mistakes that take place when a given piece of information is misunderstood or an incorrect

solution to a problem has been thought of [7,9]. On the other hand, faults are the conditions that arise from the identified errors. Faults appear in the SRS documents in the form of incorrect requirements. Thus, when a human error occurs, it automatically causes the inclusion of faults in a software system. Research has shown that human errors are indeed the most frequent types of errors that are detected in the requirements documents. Due to this it is vital to not only identify the faults, but to also understand the basis of human errors. Research proves that the practice of focusing on human errors help reduce the number and frequency of faults that appear in requirements documents [9, 11]. This practice improves the overall quality of the SRS documents. Having a knowledge about the difference between faults and errors, and the reasons behind their injection creates an awareness which further helps reduce the occurrence of faults SRS documents [7].

Past research has developed and utilized tools such as the Requirement Error Taxonomy (RET) and the Human Error Taxonomy (HET) to assist the inspectors with the error abstraction process [5, 6, 15]. However, each of these tools had their individual shortcomings. The RET failed to take into account the basic human cognitive processes responsible for the injection of faults. The HET did overcome the shortcomings of the RET, but it failed to identify the specific requirements engineering activity where the errors occur [5]. In order to tackle this problem, we used the Human Error Abstraction Assist (HEAA) tool help guide the inspectors through the human error abstraction process.

Human Error Classes

Human errors occur due to the cognitive failures either during the planning phase or during the execution phase of a software system. The failures that arise during the execution

phase are divided into two classes – Slips and Lapses [10]. The failures that arise during the planning phase is divided into a class called – Mistakes [10].

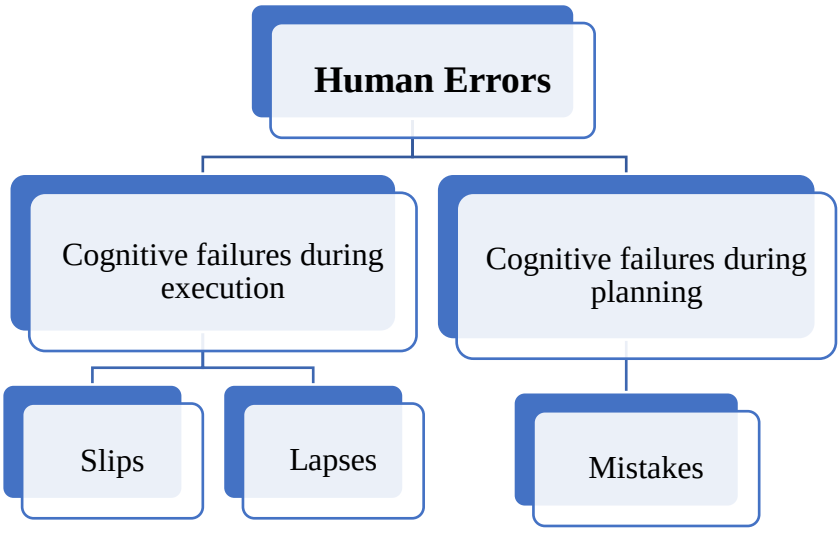

Figure 1. Human Error Classes.

Slips arise due to the lack of attention while performing a task, leading to the inferior quality of task completion. Lapses happen when an action is forgotten while performing a planned task. Mistakes occur due to insufficient planning of an action, typically arising during unfamiliar situations. All these classes of human errors thus, form the Human Error Taxonomy (HET).

Human errors are said to be present in various arenas like – medicine, aviation, etc. [3, 15]. So, it is important to learn how to reduce the amount of human errors that occur especially during the requirements elicitation phase. It is also helpful to learn how to detect the existing human errors to improve the overall quality of the software systems. It is better to detect the errors as early as possible. It is because, when errors are detected in the later phases of the SDLC model, it leads to increased overall project costs [1].

Problem Statement

In this paper, we try to address the problem of having better quality SRS documents by conducting an experiment study. The main goal of this study is to analyze the results and gauge the accuracy with which the study participants abstract human errors. The entire experiment study is centered around the following research questions:

RQ1: How do the students perform while abstracting errors from the PGCS SRS document?

RQ2: How do the students perform while abstracting errors from their own SRS document?

RQ3: Did the students benefit from the human error training provided to them in order to find additional faults that might have been missed during the traditional fault inspection method.

RQ4: What are the major human error types that are responsible for faults committed during the creation of the student's own SRS documents.

The next section of this paper describes the overall design of the study in details.

EXPERIMENT STUDY DESIGN

A research study was conducted in the Fall 2018 semester at the North Dakota State University (NDSU) to gauge the accuracy with which the study participants were able to identify faults and their underlying human errors in SRS documents. This section describes the overall experiment study design as well as the research questions that were formulated to report the study results.

Study Participants

The study was conducted in an undergraduate level Computer Science class - CSCI 413 (Principles of Software Engineering) at the North Dakota State University. A total of 45 students participated in this study. However, after detailed analysis it was found out that only 28 students were unique participants. There were some students that did not participate in the first task of this study, but they participated in the team activities. The data for those students were not considered, and hence, the study results are limited to those 28 students that participated in all task modules of this study experiment.

Research Questions

The following research questions were formulated for this study experiment. The study results were collected and organized around these research questions:

- ***RQ1:*** How do the students perform while abstracting errors from the PGCS SRS document?
- ***RQ2:*** How do the students perform while abstracting errors from their own SRS document?

- ***RQ3:*** Did the students benefit from the human error training provided to them in order to find additional faults that might have been missed during the traditional fault inspection method.
- ***RQ4:*** What are the major human error types that are responsible for faults committed during the creation of the student's own SRS documents.

Study Procedure

The entire study was broken down into smaller task modules. The study was initiated by first educating the study participants on some basic concepts and processes that the study was built upon. The training provided to the students was in a video format. It included introduction to concepts of RE and RE activities, and how to abstract errors using the HEAA tool. The video training listed some of the most commonly found faults in SRS documents and trained its viewers on the fault detection process as well. The video also provided training on how the detected faults can be traced back to its appropriate human errors which are the root causes of the injection of faults into SRS or other Business Requirement Documents (BRD). Lastly, the training provided a detailed explanation of the tasks and forms that the students were asked to complete as part of this experiment study.

Task 1: In the first part of the study, the students were provided with an industry standard SRS document which consisted of requirements for a Parking Garage Control System (PGCS). The PGCS enabled the automated entry and exit of vehicles into a parking garage. This SRS document was deliberately seeded with faults. Out of the many faults prevalent in the PGCS SRS document, the students were provided with 15 faults and they were asked to identify human errors pertaining to each one of those faults. This task was completed online through a Qualtrics survey form that was set up for the students.

The students were instructed to use the Human Error Abstraction Assist (HEAA) tool to abstract human errors and to identify the RE activity where the errors occurred. This tool consists of a three-step process that allows the better identification and analysis of the human errors associated with the provided faults. For each fault, the students were asked to answer three questions.

In the first question, students were asked to identify the Requirements Engineering (RE) activity in which the human error had occurred. The students had to pick between these options – Analysis, Elicitation, Specification, and Management. In the second question, the study participants were asked to pick a specific type of human error which they felt led to the injection of the provided fault. The options provided for this step were – Slip, Lapse, and Mistake. In the third question, the study participants were asked to pick the appropriate human error for each fault. To assist the students in picking the human error, a visual was provided to them. Below attached is a screenshot of the visual included in the final step of the first task that used the HEAA tool:

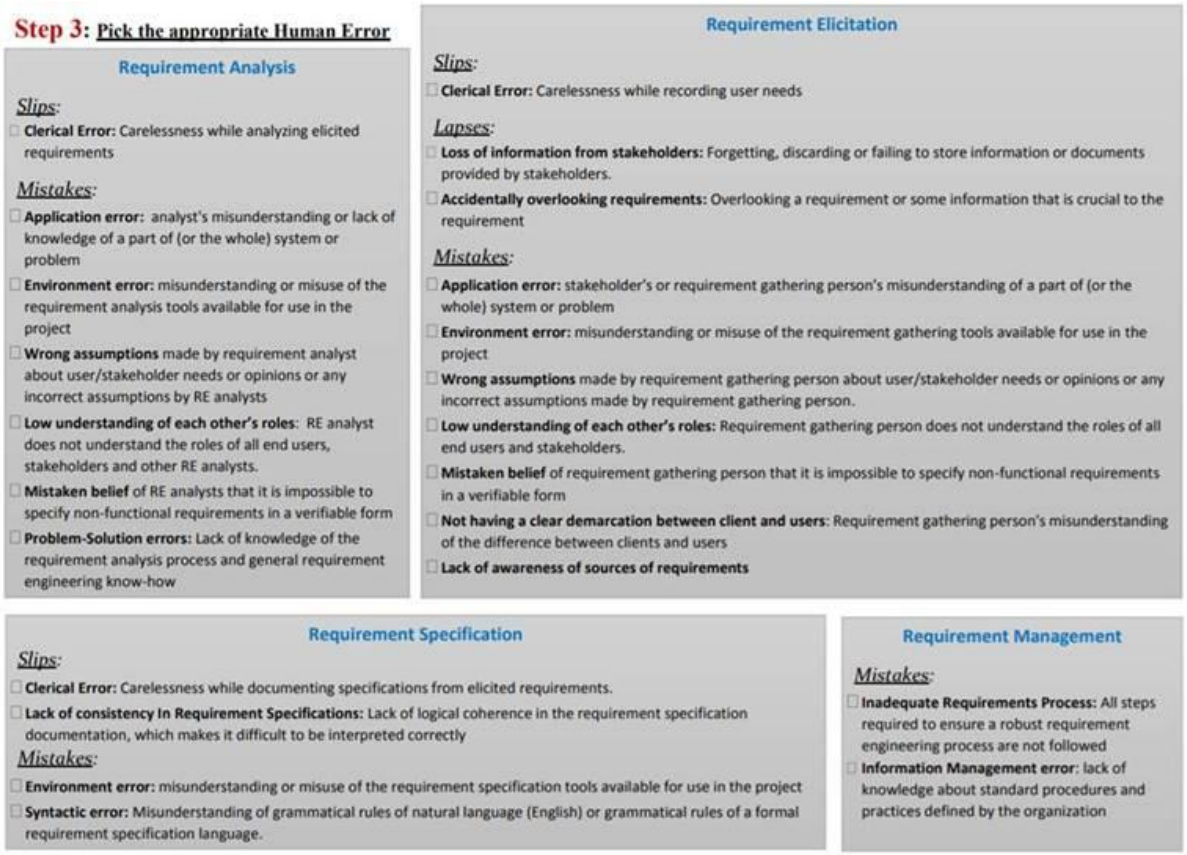

Figure 2. Human Error.

After the errors had been abstracted from the PGCS SRS, another training session was provided to the students where they reflected on their performance of the first task. A Qualtrics report was created with the students' response for each given fault. The Qualtrics report consisted of bar graphs which were copied into a PowerPoint presentation for comparing the students' results with the correct solutions. This discussion allowed the students to view their mistakes and better prepared them for the next tasks in the study.

Task 2: For the second task, the students of the CSCI 413 course were divided into five different teams. Each team was asked to prepare their own SRS document for a system of choice. Each of the five teams had different number of team members that comprised a particular team. The table below shows the details of each team that participated in this experiment:

Table 1. Team Listings.

Team #	Team Name	Total Number of Participants
Team 1	Capstone Course Management (CCM)	4
Team 2	Dissertation Calculator (DC)	2
Team 3	Science Olympiad (SO)	6
Team 4	Sugarbeet Research and Education Board (SBREB)	9
Team 5	Wonders of Weather (WOW)	9

The participants were asked to fill out a fault inspection checklist for their individually created SRS documents. After this task was completed, all the faults were consolidated in order to remove any false-positives derived from the results. This study excludes the results collected from Team 2 – DC because most of the team members did not participate in completing the second task of this study. The results obtained from this team was not very substantial in terms of data analysis and hence, the results were omitted.

Task 3: In the third task module, the students were asked to abstract human errors from the faults that the students had identified in their own SRS documents. In order to complete this task, the students used the HEAA tool. They followed the three-step process in the HEAA tool to zero in on the specific human errors that led to the injection of the identified faults. All the results from this step was collected and recorded in an Excel spreadsheet for data the purpose of data analysis

Task 4: For the final task, the study participants reviewed their personally created SRS document again to identify additional faults. These additional faults that the students were expected to find had to be different from the faults that they had previously identified in the second task of the study.

DISCUSSION OF RESULTS

This section provides details of the results that were collected from the execution of the experiment study described above. The data collected from this study is organized around the pre-formulated research questions.

RQ1: How do the students perform while abstracting errors from the PGCS SRS document?

A Qualtrics survey was set up for the first task where the students were asked to abstract human errors from the 15 faults that were provided to them. After the students completed the first task, the data from the Qualtrics survey was exported into an Excel sheet for the purpose of data analysis. A total of 28 unique students participated in the first task of the study. The table below shows the accuracy measured in terms of percentage (%) to evaluate the student's performance for each question or step involved in the human error abstraction process using the HEAA tool:

Table 2. Error Abstraction Accuracy Levels Using HEAA for PGCS SRS.

Fault #	Correct RE Activity (Step 1)	Correct Error Type (Step 2)	Correct Human Error Class (Step 3)	Overall Correctness for All Levels
Fault 1	5/28 (17.85%)	3/5 (60.00%)	0 (0.00%)	0 (0.00%)
Fault 2	17/28 (60.71%)	14/17(82.35%)	8/14(57.14%)	8/28 (28.57%)
Fault 3	11/28 (39.28%)	10/11 (90.90%)	5/10 (50.00%)	5/28 (17.85%)
Fault 4	5/28 (17.85%)	3/5 (60.00%)	3/3 (100.00%)	3/28 (10.71%)
Fault 5	5/28 (17.85%)	4/5 (80.00%)	1/4 (25.00%)	1/28 (3.57%)
Fault 6	4/28 (14.28%)	3/4 (75.00%)	1/3 (33.00%)	1/28 (3.57%)
Fault 7	7/28 (25.00%)	4/7 (57.14%)	3/4 (75.00%)	3/28 (10.71%)
Fault 8	3/28 (10.71%)	1/3 (33.33%)	1/1 (100.00%)	1/28 (3.57%)
Fault 9	17/28 (60.71%)	15/17 (88.23%)	10/15 (66.66%)	10/28 (35.71%)
Fault 10	12/28 (42.85%)	8/12 (66.66%)	5/8 (62.50%)	5/28 (17.85%)
Fault 11	3/28 (10.71%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Fault 12	13/28 (46.42%)	11/13 (84.61%)	7/11 (63.63%)	7/28 (25.00%)
Fault 13	2/28 (7.14%)	2/2 (100.00%)	0 (0.00%)	0 (0.00%)
Fault 14	4/28 (14.28%)	2/4 (50.00%)	1/2 (50.00%)	1/28 (3.57%)
Fault 15	11/28 (39.28%)	9/11 (81.81%)	0 (0.00%)	0 (0.00%)

The student's accuracy was measured in terms of percentages (%). Each row in the above table represents how the students performed for a specific fault. For instance, for Fault # 1, out of 28 students, only 5 students picked the correct RE activity where the correct human error occurs. This accounts for a total of 17.85% accuracy for the first step. For the second step, the results for only those 5 students that answered the first question correctly were recorded. As shown in the table above, 3 out of those 5 students picked the correct human error type which they felt led to the injection of the provided Fault # 1. A total accuracy of 60.00% was achieved for the second step. In the final step, none of the students out of those 3 students that picked the correct human error type were correctly able to identify the human error associated with Fault # 1. Hence, the overall accuracy for recorded at 0.00%. This process was repeated for all 15 faults that were provided to the students. It is important to note that all false-positives and duplicate results were ignored from being taken into consideration.

The average accuracies for each activity in the error abstraction process is documented in the table below:

Table 3. PGCS SRS Average Accuracies Comparison.

Error Abstraction Activities	PGCS SRS – CSCI 413 Students	Manjunath, Anu, and Walia (Control Group)
Step 1: RE Activity	28.33%	40.84%
Step 2: Error Type	67.34%	77.33%
Step 3: Error Class	45.55%	69.63%
Overall Correctness	10.71%	21.64%

The graphs below summarize the information depicted in the above attached table for a better visual representation:

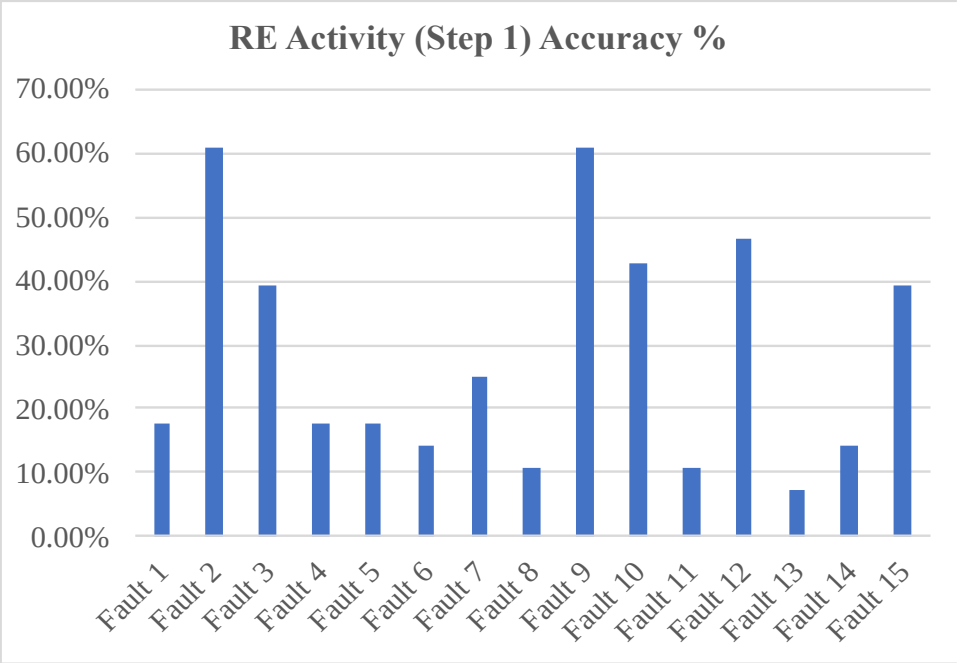

Figure 3. Step 1 Accuracy % for PGCS SRS Faults.

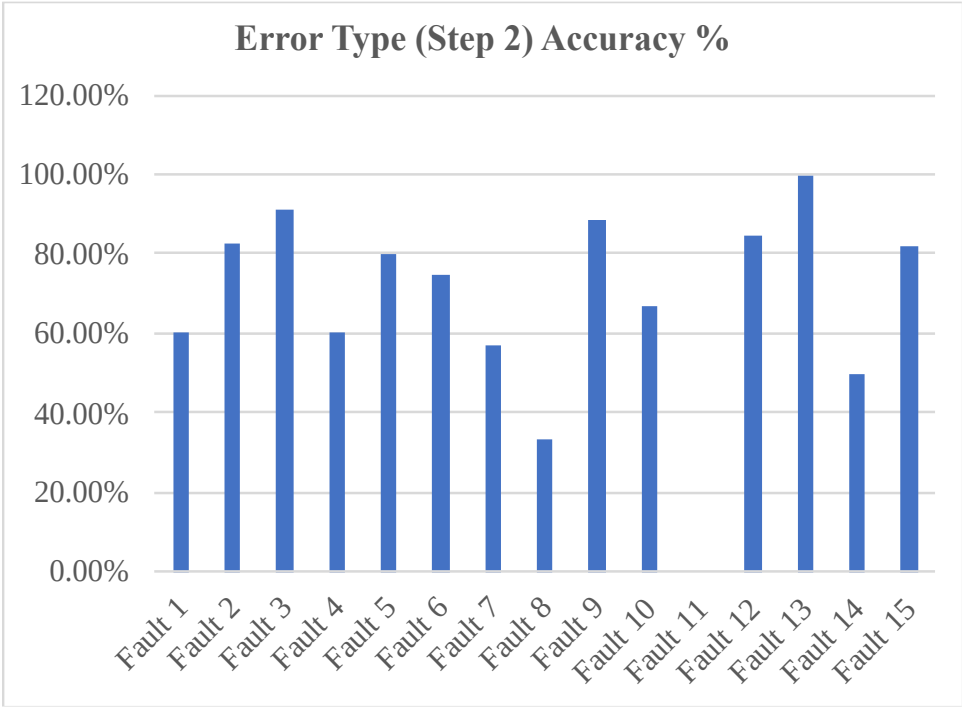

Figure 4. Step 2 Accuracy % for PGCS SRS Faults.

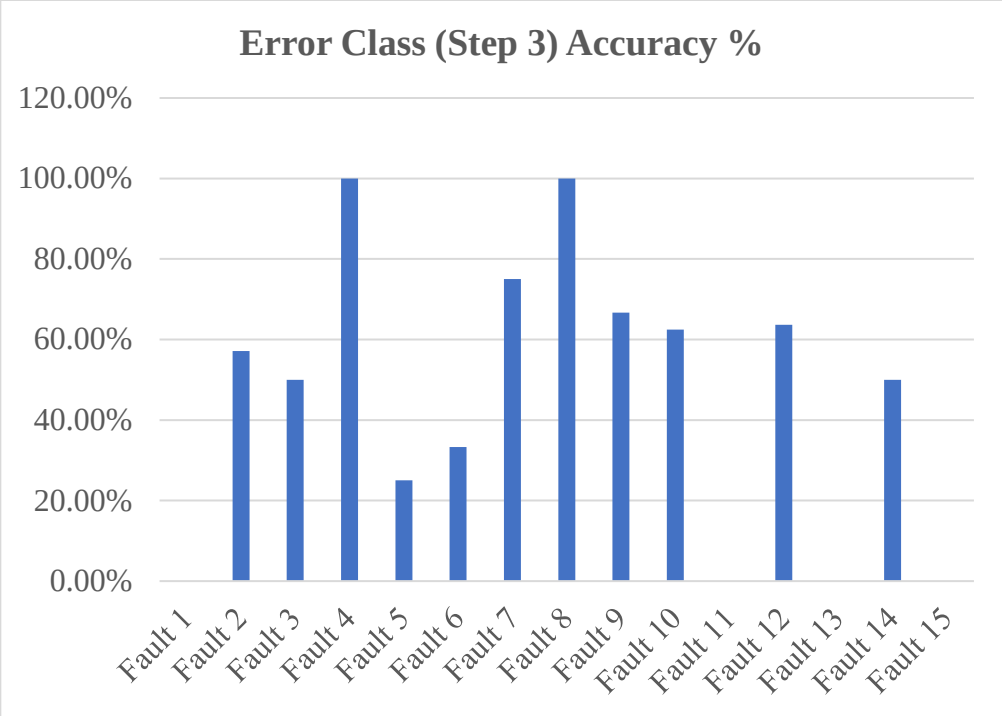

Figure 5. Step 3 Accuracy % for PGCS SRS Faults.

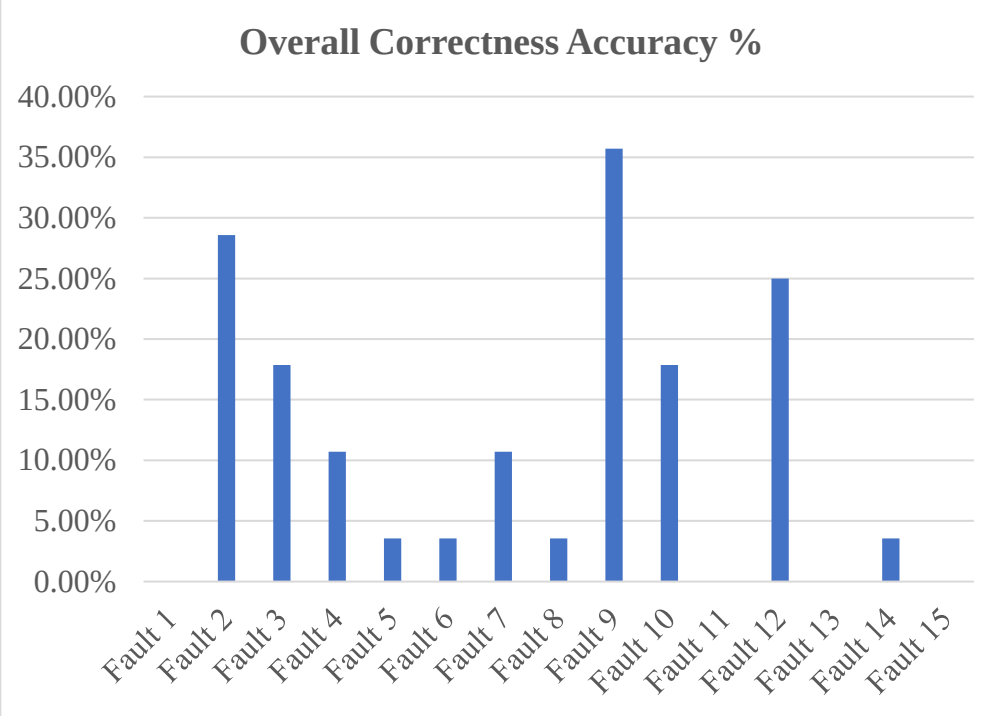

Figure 6. Overall Accuracy % for PGCS Faults.

RQ2: How do the students perform while abstracting errors from their own SRS document?

After the students documented the faults and abstracted the human errors from their own SRS documents, the results were compiled together in an Excel spreadsheet. Attached below are their results:

Table 4. Error Abstraction Accuracy Levels Using HEAA for Each Team.

Team	Fault #	Correct RE Activity (Step 1)	Correct Error Type (Step 2)	Correct Error Class (Step 3)	Overall Correctness for All Levels
Team 1: CCM	Fault # 1	3/4 (75.00%)	3/3 (100.00%)	2/3 (66.66%)	2/4 (50.00%)
	Fault # 2	3/4 (75.00%)	3/3 (100.00%)	2/3 (66.66%)	2/4 (50.00%)
	Fault # 3	3/4 (75.00%)	2/3 (66.66%)	1/2 (50.00%)	1/4 (25.00%)
	Fault # 4	2/4 (50.00%)	2/2 (100.00%)	1/2 (50.00%)	1/4 (25.00%)
	Fault # 5	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
	Fault # 6	4/4 (100.00%)	3/4 (75.00%)	2/3 (66.66%)	2/4 (50.00%)
Team 3: SO	Fault # 1	5/6 (83.33%)	4/5 (80.00%)	3/4 (75.00%)	3/6 (50.00%)
	Fault # 2	4/6 (66.66%)	4/4 (100.00%)	2/4 (50.00%)	2/6 (33.33%)
	Fault # 3	6/6 (100.00%)	5/6 (83.33%)	4/5 (80.00%)	4/6 (66.66%)
	Fault # 4	3/6 (50.00%)	3/3 (100.00%)	2/3 (66.66%)	2/6 (33.33%)
	Fault # 5	2/6 (33.33%)	2/2 (100.00%)	1/2 (50.00%)	1/6 (16.66%)
	Fault # 6	4/6 (66.66%)	4/4 (100.00%)	3/4 (75.00%)	3/6 (50.00%)
	Fault # 7	3/6 (50.00%)	3/3 (100.00%)	3/3 (100.00%)	3/6 (50.00%)
	Fault # 8	3/6 (50.00%)	3/3 (100.00%)	3/3 (100.00%)	3/6 (50.00%)
	Fault # 9	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Team 4: SBREB	Fault # 1	5/9 (55.55%)	4/5 (80.00%)	2/4 (50.00%)	2/9 (22.22%)
	Fault # 2	9/9 (100.00%)	6/9 (66.66%)	5/6 (83.33%)	5/9 (55.55%)
	Fault # 3	7/9 (77.77%)	6/7 (85.71%)	4/6 (66.66%)	4/9 (44.44%)
	Fault # 4	3/9 (33.33%)	3/3 (100.00%)	3/3 (100.00%)	3/9 (33.33%)
	Fault # 5	5/9 (55.55%)	4/5 (80.00%)	3/4 (75.00%)	3/9 (33.33%)
	Fault # 6	5/9 (55.55%)	4/5 (80.00%)	2/4 (50.00%)	2/9 (22.22%)
	Fault # 7	4/9 (44.44%)	3/4 (75.00%)	2/3 (66.66%)	2/9 (22.22%)
	Fault # 8	6/9 (66.66%)	5/6 (83.33%)	4/5 (80.00%)	4/9 (44.44%)
	Fault # 9	8/9 (88.88%)	4/8 (50.00%)	2/4 (50.00%)	2/9 (22.22%)
	Fault # 10	7/9 (77.77%)	6/7 (85.71%)	5/6 (83.33%)	5/9 (55.55%)
	Fault # 11	8/9 (88.88%)	5/8 (62.50%)	2/5 (40.00%)	2/9 (22.22%)
Team 5: WOW	Fault # 1	6/9 (66.66%)	6/6 (100.00%)	4/6 (66.66%)	4/9 (44.44%)
	Fault # 2	5/9 (55.55%)	4/5 (80.00%)	4/4 (100.00%)	4/9 (44.44%)
	Fault # 3	6/9 (66.66%)	4/6 (66.66%)	4/4 (100.00%)	4/9 (44.44%)
	Fault # 4	5/9 (55.55%)	4/5 (80.00%)	4/4 (100.00%)	4/9 (44.44%)
	Fault # 5	5/9 (55.55%)	5/5 (100.00%)	3/5 (60.00%)	3/9 (33.33%)

Next, the mass data collected for how each student performed the error abstraction activity for the PGCS SRS was organized based off the different teams. This data was compared with how each team performed the error abstraction activity on their individual SRS documents. The series of graphs below depict the comparison for each team's performance based on the overall correctness achieved by them.

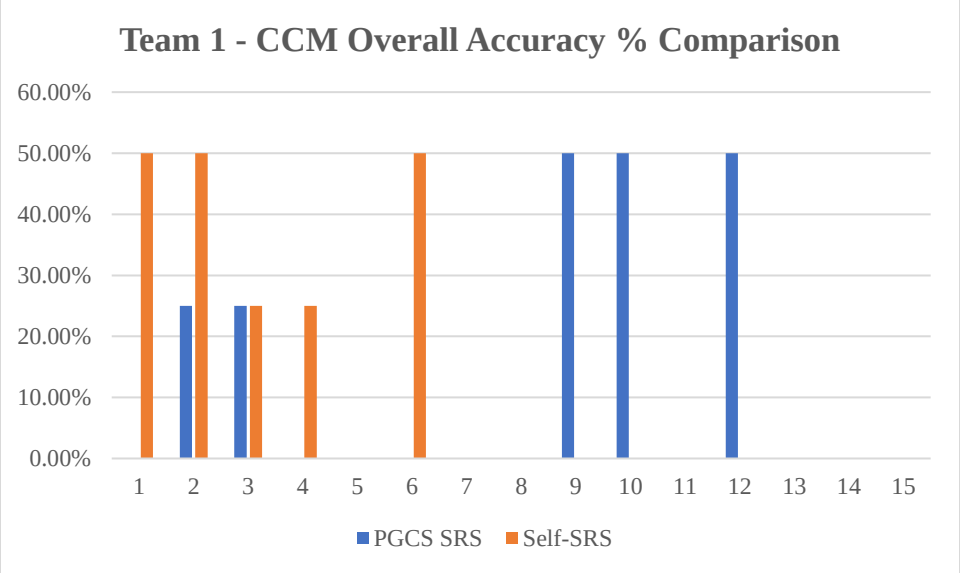

Figure 7. Team 1 – CCM Overall Accuracy % Comparison.

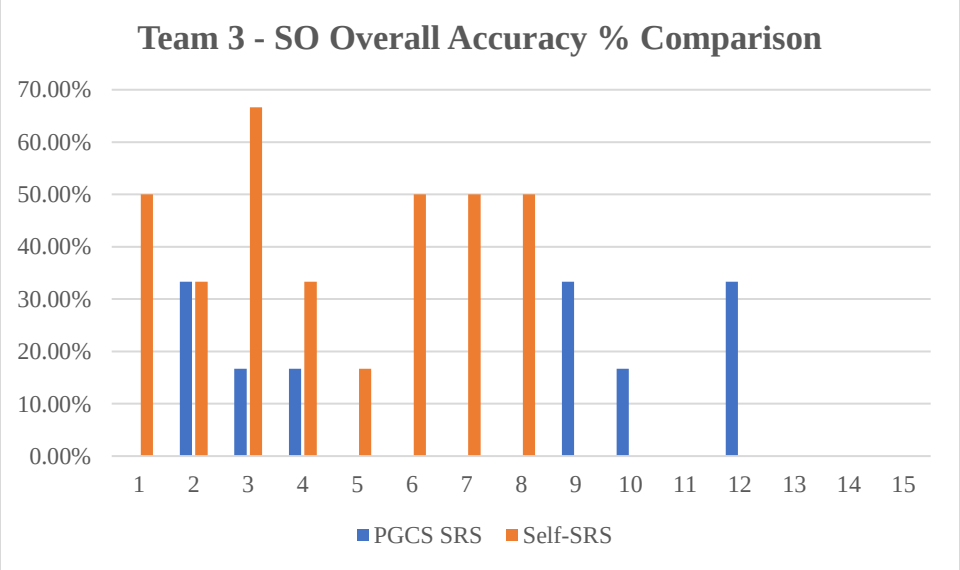

Figure 8. Team 3 – SO Overall Accuracy % Comparison.

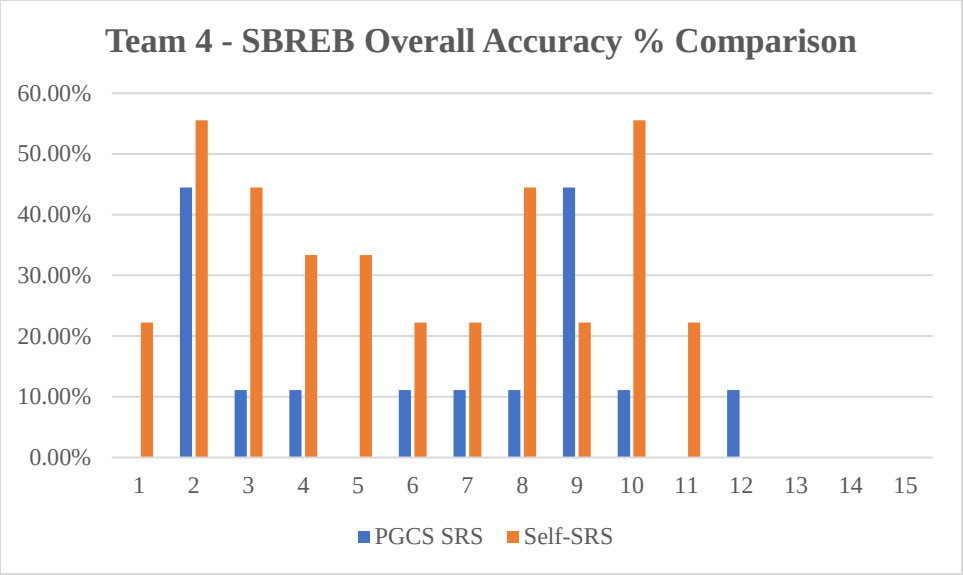

Figure 9. Team 4 – SBREB Overall Accuracy % Comparison.

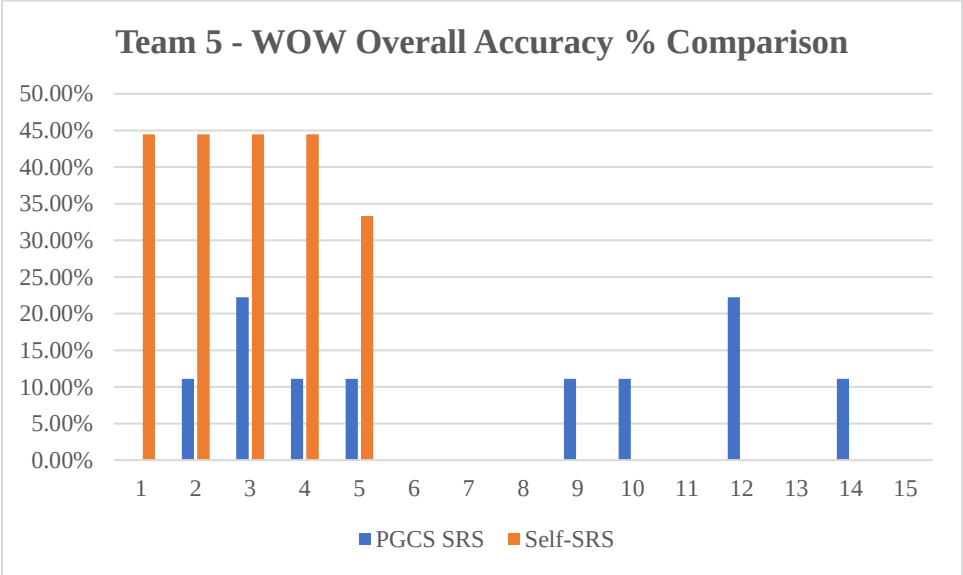

Figure 10. Team 5 – WOW Overall Accuracy % Comparison.

A t-test was conducted to know whether there was significant difference between how the teams performed the error-abstraction activity for the PGCS SRS document vs. their Self-SRS document. The data for each team was coded in the SAS software to run the t-test and the results are documented below.

t-test for Team_1(CCM) with equal variance

The TTEST Procedure

Variable: yield

trt	N	Mean	Std Dev	Std Err	Minimum	Maximum
PF	15	0.1333	0.2085	0.0538	0	0.5000
SF	6	0.3333	0.2041	0.0833	0	0.5000
Diff (1-2)		-0.2000	0.2073	0.1001		

trt	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
PF		0.1333	0.0179	0.2488	0.2085	0.1526	0.3287
SF		0.3333	0.1191	0.5475	0.2041	0.1274	0.5006
Diff (1-2)	Pooled	-0.2000	-0.4096	0.00961	0.2073	0.1577	0.3028
Diff (1-2)	Satterthwaite	-0.2000	-0.4228	0.0228			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	19	-2.00	0.0603
Satterthwaite	Unequal	9.4539	-2.02	0.0731

Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	14	5	1.04	1.0000

Figure 11. Team 1 – CCM t-test Results.

Team 1 shows no significant difference between the mean of the PGCS SRS (0.13) and the Self-SRS (0.33) because p-value is greater than 0.05 ($p > 0.05$).

t-test for Team_3(SO) with equal variance

The TTEST Procedure

Variable: yield

trt	N	Mean	Std Dev	Std Err	Minimum	Maximum
PF	15	0.1000	0.1372	0.0354	0	0.3300
SF	11	0.3418	0.1375	0.0414	0.2200	0.5600
Diff (1-2)		-0.2418	0.1373	0.0545		

trt	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
PF		0.1000	0.0240	0.1760	0.1372	0.1004	0.2163
SF		0.3418	0.2495	0.4342	0.1375	0.0960	0.2412
Diff (1-2)	Pooled	-0.2418	-0.3543	-0.1293	0.1373	0.1072	0.1910
Diff (1-2)	Satterthwaite	-0.2418	-0.3550	-0.1287			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	24	-4.44	0.0002
Satterthwaite	Unequal	21.679	-4.44	0.0002

Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	10	14	1.00	0.9680

Figure 12. Team 3 – SO t-test Results.

Team 3 has a p-value of $p \leq 0.05$ and hence, they show a significant difference between the mean of the PGCS SRS (0.09) and the Self-SRS (0.38).

t-test for Team_4(SBREB) with equal variance

The TTEST Procedure

Variable: yield

trt	N	Mean	Std Dev	Std Err	Minimum	Maximum
PF	15	0.1100	0.1440	0.0372	0	0.4400
SF	9	0.3889	0.2045	0.0682	0	0.6700
Diff (1-2)		-0.2789	0.1685	0.0711		

trt	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
PF		0.1100	0.0302	0.1898	0.1440	0.1054	0.2271
SF		0.3889	0.2317	0.5461	0.2045	0.1381	0.3917
Diff (1-2)	Pooled	-0.2789	-0.4263	-0.1315	0.1685	0.1303	0.2385
Diff (1-2)	Satterthwaite	-0.2789	-0.4469	-0.1109			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	22	-3.92	0.0007
Satterthwaite	Unequal	12.822	-3.59	0.0033

Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	8	14	2.02	0.2402

Figure 13. Team 4 – SBREB t-test Results.

Team 4 also shows a significant difference in their performance as $p \leq 0.05$.

t-test for Team_4(SBREB) with equal variance

The TTEST Procedure

Variable: yield

trt	N	Mean	Std Dev	Std Err	Minimum	Maximum
PF	15	0.1100	0.1440	0.0372	0	0.4400
SF	9	0.3889	0.2045	0.0682	0	0.6700
Diff (1-2)		-0.2789	0.1685	0.0711		

trt	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
PF		0.1100	0.0302	0.1898	0.1440	0.1054	0.2271
SF		0.3889	0.2317	0.5461	0.2045	0.1381	0.3917
Diff (1-2)	Pooled	-0.2789	-0.4263	-0.1315	0.1685	0.1303	0.2385
Diff (1-2)	Satterthwaite	-0.2789	-0.4469	-0.1109			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	22	-3.92	0.0007
Satterthwaite	Unequal	12.822	-3.59	0.0033

Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	8	14	2.02	0.2402

Figure 14. Team 5 – WOW t-test Results.

Team 5 also shows significant difference in its performance and has a p-value of less than equal to 0.05 ($p \leq 0.05$).

The average accuracy percentages for each team and for each step are summarized in the table below:

Table 5. Average Accuracy % for All Teams.

Team	Step 1 Average Accuracy %	Step 2 Average Accuracy %	Step 3 Average Accuracy %	Overall Average Accuracy %
Team 1: CCM	75.00%	86.66%	61.53%	40.00%
Team 3: SO	62.50%	66.66%	75.00%	43.75%
Team 4: SBREB	67.67%	74.62%	68.00%	34.34%
Team 5: WOW	60.00%	50.00%	82.60%	42.22%

The performance of each team cannot directly be compared with one another as the teams were reporting faults and abstracting human errors from different SRS documents. Each team reported a different number of faults for which they abstracted the errors. Hence, unlike the first task, there was no basis for direct comparison between how the teams performed.

RQ3: Did the students benefit from the human error training provided to them in order to find additional faults that might have been missed during the traditional fault inspection method.

We can say that the training provided to the students was beneficial because each team reported some additional faults. The table below summarizes the total number of additional faults that the teams reported.

Table 6. Additional Faults Reported by Each Team.

Additional Faults:	Team 1: CCM	Team 3: SO	Team 4: SBREB	Team 5: WOW
Fault # 1	4	7	5	4
Fault # 2	4	3	3	4
Fault # 3	4	5	6	4
Fault # 4	3	1	3	2
Fault # 5	4	1	3	2
Fault # 6	2	3	4	
Fault # 7		2	2	
Fault # 8		2	2	
Fault # 9		2	1	
Fault # 10			2	
Fault # 11			2	

RQ4*:* What are the major human error types that are responsible for faults committed during the creation of the student's own SRS documents.

After computing all the results together, it was found most of the study participants picked Slip as the type of human error that accounted for majority of the faults that were reported in individual SRS documents. After combining the results from all the faults identified by all the teams, it was found that Slip was chosen 109 times, Mistake was chosen 55 times, followed by Lapse which was chosen 44 times. These numbers were derived by doing a simple count for the number of times each specific error type was selected for each specific fault by every team. Hence, slips account for the majority of human errors that were responsible for the faults committed during the creation of the student's own SRS documents.

Threats to Validity

During this experiment study, we discovered external threats to validity. Although the students were trained on the different tasks involved in this study, their results ranked lower than that of the study conducted with industry practitioners described in [10]. It is because the study participants were a group of undergraduate students that lacked experience in creating industry

standard SRS documents and had a limited domain knowledge. The study participants had not been exposed to the concepts of RE and its activities, and human errors vs. faults in the past. Another major threat to validity includes the use of self-developed SRS documents as a means of comparing performance in terms of accuracy. It is because we are unaware of the total number of actual faults and errors that exist in those documents. Similar to past studies, the validity of all detected faults from self-developed SRS documents was accredited to the judgement of the researchers [5]. Another validity threat discovered was the lack of a control group which was similar in size as our treatment group. The control group used for this study consisted of 10 participants, whereas, the treatment group consisted of 28 participants. The control group participants had more experience as compared to our study participants.

CONCLUSION AND FUTURE WORK

Through the results gathered from the experiment that was conducted, we can conclude that the training provided to the students allows them to successfully identify faults in their individual SRS documents. The training also helps the students with the human error abstraction task where they were asked to pick the appropriate human error that they felt led to the injection of faults in their SRS documents. Without the training about the concepts of human errors, faults, and a description of the tasks that the study were expected to complete, the students probably would not have performed well. HEAA is a beneficial tool which helped the students in further narrowing down the human error by dividing the error abstraction into three simple steps.

Overall, this experiment study provided a great learning opportunity to all the students that participated in it. Having a knowledge about the difference between faults and human errors and how faults are injected into BRD documents, and how they can be traced back and prevented creates an awareness in the minds of the students. This awareness will allow the students to be mindful of such defects whenever they prepare other BRD documents in the future. It will also allow them the ability to view their own work as well the work of the others critically in order to make the BRD documents as error free and as consistent as possible.

In the future studies we aim at conducting pre-assessment knowledge tests before training the study participants in order to gauge and analyze actual improvement levels in terms of accuracy. We also aim at removing duplicates and false positives from the data collected regarding additional faults found by the participants in their self-developed SRS documents. The participants in this study performed poorly as compared to the study conducted by Manjunath, et al [10]. with the industry practitioners. Hence, we aim at improving the training video and modifying the teaching styles and methods used in the CSCI 413 class to better train the

participants. This would further help strengthen our studies by quality of data collected for analysis.

REFERENCES

1. Alshamrani, A., & Bahattab, A. 2015. A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. IJCSI International Journal of Computer Science. https://www.ijcsi.org/papers/IJCSI-12-1-1-106-111.pdf
2. Al-Zawahreh, H., & Almakadmeh, K. 2015. Procedural Model of Requirements Elicitation Techniques. International Conference on Intelligent Information Processing, Security and Advanced Communication. https://www.researchgate.net/publication/280154366_Procedural_Model_of_Requirements_Elicitation_Techniques
3. Anu, V., Walia G., & Bradshaw G. 2017. Incorporating Human Error Education into Software Engineering Courses via Error-based Inspections. 2017 ACM SIGCSE Technical Symposium. https://www.researchgate.net/publication/314293090_Incorporating_Human_Error_Education_into_Software_Engineering_Courses_via_Error-based_Inspections
4. Anu, V., Walia, G., Hu, W., Carver, J. C., & Bradshaw, G. 2016. Effectiveness of Human Error Taxonomy during Requirements Inspection: An Empirical Investigation. SEKE 2016. https://www.researchgate.net/publication/310417842_Effectiveness_of_Human_Error_Taxonomy_during_Requirements_Inspection_An_Empirical_Investigation
5. Anu, V., Walia, G., Hu, W., Carver, J. C., & Bradshaw, G. 2016. Using A Cognitive Psychology Perspective on Errors to Improve Requirements Quality: An Empirical Investigation. IEEE 27th International Symposium on Software Reliability Engineering. https://www.researchgate.net/publication/310418326_Using_a_Cognitive_Psychology_Perspective_on_Errors_to_Improve_Requirements_Quality_An_Empirical_Investigation

6. Anu, V., Walia, G., Hu, W., Carver, J. C., & Bradshaw, G. 2017. Issues and Opportunities for Human Error-based Requirements Inspections: An Exploratory Study. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. https://ieeexplore.ieee.org/document/8170135
7. Chakraborty, A., Baowaly, M. K., Arefin, A., & Bahar, A. N. 2012. The Role of Requirement Engineering in Software Development Life Cycle. Journal of Emerging Trends in Computing and Information Sciences. https://www.researchgate.net/publication/267631563_The_Role_of_Requirement_Engineering_in_Software_Development_Life_Cycle
8. Hu, W., Carver, J. C., Anu, V. K., Walia, G. S., & Bradshaw, G. 2016. Detection of Requirement Errors and Faults via a Human Error Taxonomy: A Feasibility Study. ESEM 2016. https://www.researchgate.net/publication/310417411_Detection_of_Requirement_Errors_and_Faults_via_a_Human_Error_Taxonomy_A_Feasibility_Study
9. Lanbuile, F., Shull, F., & Basili, V. R. 1998. Experimenting with Error Abstraction in Requirements Documents. Software Metrics Symposium, Proceedings Fifth International. https://www.researchgate.net/publication/3779940_Experimenting_with_Error_Abstraction_in_Requirements_Documents
10. Manjunath, K., Anu, V., Walia, G., & Bradshaw, G. 2018. Training Industry Practitioners to Investigate the Human Error Causes of Requirements Faults. 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). https://www.researchgate.net/publication/329064169_Training_Industry_Practitioners_to_Investigate_the_Human_Error_Causes_of_Requirements_Faults

11. Meister, D. 1989. The Nature of Human Error. 1989 IEEE Global Telecommunications Conference and Exhibition 'Communications Technology for the 1990s and Beyond'. https://ieeexplore.ieee.org/document/64071

12. Osman, M. H., & Zaharin M. F. 2018. Ambiguous Software Requirement Specification Detection: An Automated Approach. the 5th International Workshop. https://www.researchgate.net/publication/326072143_Ambiguous_software_requirement_specification_detection_an_automated_approach

13. Saarelainen K., & Jantti M. 2015. Quality and Human Errors in IT Service Infrastructures – Human error based root causes of incidents and their categorization. Conference: 2015 11th International Conference on Innovations in Information Technology (IIT). https://www.researchgate.net/publication/304294188_Quality_and_human_errors_in_IT_service_infrastructures_-_Human_error_based_root_causes_of_incidents_and_their_categorization

14. Walia, G. S., Bradshaw, G. S., & Carver, J. C. 2015. Workshop on Applications of Human Error Research to Improve Software Engineering. Conference: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE). https://www.researchgate.net/publication/308849521_Workshop_on_Applications_of_Human_Error_Research_to_Improve_Software_Engineering_WAHESE_2015

15. Walia, G. S., & Carver, J. C. 2009. A systematic literature review to identify and classify software requirement errors. Information and Software Technology. https://www-sciencedirect-com.ezproxy.lib.ndsu.nodak.edu/science/article/pii/S0950584909000111

16. Walia, G. S., Carver, J. C., & Philip, T. 2007. Requirement error abstraction and classification: A control group replicated study. Proceedings - International Symposium on

Software Reliability Engineering, ISSRE. https://ieeexplore-ieee-org.ezproxy.lib.ndsu.nodak.edu/document/4402198