Requirement Error Abstraction and Classification: A Control Group Replicated Study

Gursimran S. Walia, Jeffrey C. Carver, and Thomas Philip
Mississippi State University
{gw86, carver, philip}@cse.msstate.edu

Abstract

This paper is the second in a series of empirical studies about requirement error abstraction and classification as a quality improvement approach. The Requirement error abstraction and classification method supports the developers' effort in efficiently identifying the root cause of requirements faults. By uncovering the source of faults, the developers can locate and remove additional related faults that may have been overlooked, thereby improving the quality and reliability of the resulting system. This study is a replication of an earlier study that adds a control group to address a major validity threat. The approach studied includes a process for abstracting errors from faults and provides a requirement error taxonomy for organizing those errors. A unique aspect of this work is the use of research from human cognition to improve the process. The results of the replication are presented and compared with the results from the original study. Overall, the results from this study indicate that the error abstraction and classification approach improves the effectiveness and efficiency of inspectors. The requirement error taxonomy is viewed favorably and provides useful insights into the source of faults. In addition, human cognition research is shown to be an important factor that affects the performance of the inspectors. This study also provides additional evidence to motivate further research.

1. Introduction

Software quality is a major issue for software engineers. Various quality improvement approaches have focused on faults as an indication of problems early in the development process. To accomplish this goal, these approaches provide guidance to developers on the identification of faults [2, 3, 4, 5, 7, 8]. However, even with these approaches, the desired level of quality is not always achieved because identification of faults can not always reveal all the problems. There is a need to understand the actual underlying cause of the defects. To address this need, the research described in this paper goes a step back from faults to focus on the underlying cause of faults (i.e., errors) to improve software quality.

A detailed systematic literature review identified several methods that use the sources of faults as a means to improve quality. In analyzing the strengths and weaknesses of these methods, it became clear that most of them do not provide developers with an easy mechanism to use this information in practice [13].

Lanubile et al. provided evidence of the usefulness of error information in requirements inspection. They differentiated between error (a mistake in the human thought process) and fault (a concrete manifestation of the error(s)) based on standard IEEE definitions. . Lanubile, et al. described an error abstraction process for analyzing group of defects to determine their underlying cause. This information is then used to locate additional related defects. The process is heavily reliant on the creativity of developers to analyze and abstract errors and does not provide support for those activities [6]. Research in this paper builds on that work by providing developers a method for identifying requirement errors.

Another unique aspect of this work is the realization that human cognition research has focused on understanding human errors in a general sense for many years. Research into human errors needs to be integrated into the software quality process.

To understand the types of errors that software developers make when creating requirements, the systematic literature review included literature from both the software engineering and cognitive psychology domains. The results from this extensive literature search led to the development of a detailed taxonomy of requirement errors. The taxonomy quantifies the requirement error abstraction process by providing a list of error types for developers to focus on. This taxonomy is summarized in Section 2, with a full description already published [11, 13].

We conducted an initial study of this approach in a controlled experiment setting. The initial results were encouraging; however there were some unaddressed

1071-9458/07 $25.00 © 2007 IEEE
DOI 10.1109/ISSRE.2007.14

validity threats that motivated a replication of that study with a different experimental design.

Section 3 discusses the details of the previous study, the important results achieved, and the threats that motivated this study. Section 4 describes the experimental design. Section 5 describes the data analysis and results. Section 6 discusses the threats to validity. Section 7 talks about the relevance of the results. Section 8 compares the results of the replication to those from the first study. Section 9 contains the conclusions and future work.

2. Error Abstraction and Classification Process

The first step in the proposed approach is the error abstraction process. This step provides guidance to help developers analyze related faults and determine the underlying error(s). This process is similar to that introduced by Lanubile, et al. Understanding the errors that occurred during requirements development can help inspectors locate additional faults, related to those errors, in the requirements document.

After the error abstraction, the errors are classified into a requirement error taxonomy (RET). Because the error abstraction step is dependent on the ability of the inspectors to identify errors, it is likely that not all errors will be identified. The requirement error taxonomy provides developers with a list of the types of errors that may be present and focuses inspectors on related faults.

Errors are grouped into three major types: People Errors, Process Errors, and Documentation Errors. *People Errors* are errors caused by the individual fallibilities of the people involved in the development process; *Process Errors* are errors caused by selection of an inappropriate requirement engineering process; and *Documentation Errors* are caused by mistakes in organizing and specifying the requirements. Each of these high-level error types are composed of a set of more detailed error classes as shown in Figure 1. An example of a people error and the resulting fault is:

Error: An important stakeholder (e.g., bank manager in ATM system) was not involved while gathering requirements

Fault: Some functionality (e.g., handling multiple ATM cars simultaneously) was omitted. Similarly, different errors and faults are described for people, process, and documentation errors.

3. Background

The study described in this paper was motivated by the results of an earlier study. This section describes that study, and threats that motivated the replication. The original study evaluated the process described in Section 2 with a repeated factorial experiment design. The study consisted of sixteen senior software engineering students developing a real requirements document through interaction with clients. There were two teams, each developing a different system. After developing the requirements document, the subjects inspected the document using a fault checklist. Next, they were trained in the error abstraction process to use on those faults. After determining the errors, the subjects were trained on the classifying errors using the requirement error taxonomy. This error information was then used to reinspect the requirements document to locate additional faults.

The data analysis included comparison of the increase in defects found by each subject from first inspection (using fault checklist) to second inspection (using the error abstraction process). All analysis was done within the teams (i.e. there were no comparison between the teams). The results suggested that the error abstraction process provides a significant increase in number of faults, and that requirement error taxonomy was useful, well understood, and modular. Also *People Errors* were identified as the major causes of faults. There was a strong contribution from human cognition research. Finally, other independent variables showed an effect on the performance of subjects. Complete details about the study and its results have been published [11, 12].

A major validity threat in the original study was the lack of a control group. It is possible that some of the performance increase when using the error abstraction process could have been caused simply by the fact that the subjects were inspecting the document a second time. This threat motivated the need to perform a non-equivalent control group study to determine whether the increase in fault detection was due to the error abstraction process rather than to the reinspection.

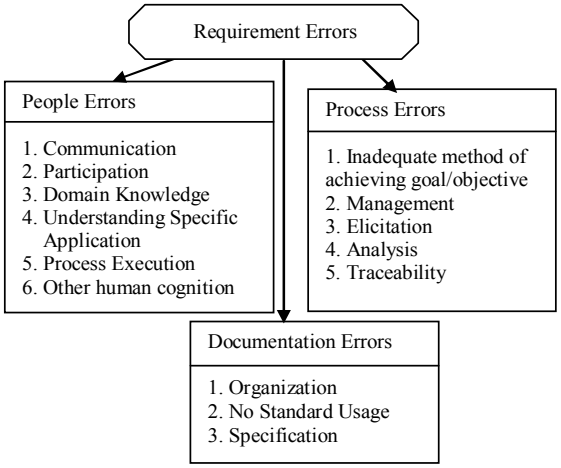

Figure 1. Requirement Error Taxonomy [12]

4. Experimental Design

To address the threat to validity described in Section 3, the major goal of this study is to understand whether the cause of additional faults found during a reinspection is the error abstraction and classification process or whether it is simply the reinspection process. The design of this experiment is a non-equivalent pretest posttest control group quasi-experiment design consisting of a control group and an experimental group. The details of the study are provided in the remainder of this section.

4.1 Methodology

This study was a replication of the original study described in Section 3 (with the addition of a control group). To enable the results of the two studies to be compared, the same hypotheses were used.

4.1.1 Hypotheses

Hypothesis 1: The error abstraction and classification process improves the effectiveness (number of faults) and efficiency (faults per hour) for teams and for individuals.

Hypothesis 2: The requirement error taxonomy is useful for improving software quality.

Hypothesis 3: The requirement error taxonomy provides important insights into the requirement phase of software development process.

Hypothesis 4: Research from the fields of human cognition, and psychology lead to additional faults.

Hypothesis 5: Individual performance during the error abstraction and classification process depends on a set of independent variables.

4.1.2 Variables

Independent Variables

1. **Process conformance –** measures how closely subjects follow the error abstraction, classification, and re-inspection processes.
2. The **pre-test –** measures the performance of subjects during an in-class exercise.
3. The **training procedures** (training 1, 2 and 3) – measure the perceived usefulness of the training by each subject.
4. **Effort** –amount of time spent during each phase of the error abstraction, classification, and re-inspection.
5. D**ifficulty level –** degree of difficulty faced by the subjects when performing the experimental tasks.

Dependent Variables

1. **Effectiveness** is the number of faults found by each subject.
2. **Efficiency** is the number of faults found by each subject per hour.

4.1.3 Subjects. Eighteen computer science graduate students participated in this study. The subjects were drawn from two full level graduate courses: Software Verification and Validation (V&V) and Empirical Software Engineering (ESE). The Software Verification and Validation course focused on various quality improvement approaches with a special focus on software inspections. The primary goal of the Empirical Software Engineering course was to teach the concepts related to the design of empirical studies and data analysis.

4.1.4 Artifacts. The software requirement specification used in the study was the Data Warehouse Functional requirements document produced by the Naval Oceanographic Office. The subjects in this study were not involved in the development of the requirements document nor did they have access to any of the people who were involved in its development.

4.2 Experimental Procedure

To evaluate the hypotheses posed in Section 4.1.1, the study was designed to contain a control group (Section 4.2.1) and an experimental group (Section 4.2.2). Figure 2 gives an overview of the procedure followed. Table 1 indicates the timeline. The groups were constructed by using the students in the V&V course as the control group and the ESE course as the experimental group. Of the 18 subjects, four were enrolled in both the courses. To balance the groups, these four subjects were allocated to one of the groups based on their experiences. (Those four subjects were not aware of what was occurring in the other group). Each started with nine subjects. However, one subject from experimental group opted out of the experiment reducing it to eight subjects.

To prevent any bias in favor of the error abstraction and classification process, the V&V course was chosen as the control group because it was already focused on the topic of software quality improvement and those students would likely be more motivated to perform well during the study.

4.2.1 Control Group. The procedure followed by the control group consisted of the following steps:

- *Training 1* – Fault checklist Technique: During this 50 minutes session, the subjects were given description of fault checklist and fault classes. Subjects were taught how to use it on an SRS document to locate faults and how to record faults. The fault checklist technique used in this experiment has been used in empirical studies for comparing detection methods for inspections [10].

- *Step 1* - Inspecting SRS for Faults: Using the information from Training 1, each subject inspected the requirements using a fault checklist. This step produced 9 individual fault lists (one per subject).
- Training 2: Re-inspection of SRS: During this 20-minute session, subjects were informed that additional faults remained in the document and motivated to re-inspect it to find the remaining faults missed during the first inspection.
- *Step 2*- Re-inspecting SRS: Using the same fault checklist as for Step 1, each subject re-inspected the requirements document. These faults were recorded in a new fault list. This step produced 9 new fault lists (one per subject).
- *Post-study Questionnaire*: The subjects were given an opportunity to provide feedback performing the inspection with the fault checklist.

4.2.2 Experimental Group. The following procedure was used by members of the experimental group:

- *Training 1* – Fault checklist Technique: Same as for the Control Group.
- *Step 1* – Inspecting SRS for faults: Same as for the control group. This step produced eight individual fault lists (one per subject).
- *Training 2* – Error Abstraction: During this 40-minute session, the subjects were trained on the error abstraction process. They were also instructed on how to use the error-fault form. A detailed description of the error abstraction training has already been published [11].
- *Step 2* – Abstraction of Errors: The subjects used the knowledge from Training 2 to extract the errors from the faults on their individual fault lists. These errors were documented in an Error-Fault List. The output of this step was 8 individual error-fault lists (one per subject).
- *Training 3* – Requirement Error Classification: This 90 minute session focused on the requirement error taxonomy and its use. The taxonomy was explained in detail. The students were taught how to classify errors and how to use error information to reinspect the requirements document. The students were then given a description of an example system and asked to classify 12 errors using the taxonomy. The students' classifications of these errors provide an idea of how well they understood the classification scheme. To combat a potential validity threat of learning, two error lists were prepared (A and B) with the same set of errors in different orders. Half the students received List A and half received List B.
- *Step 3* – Classification of Errors: Using their individual error-fault lists from Step 2 the subjects abstracted and classified errors. While doing this

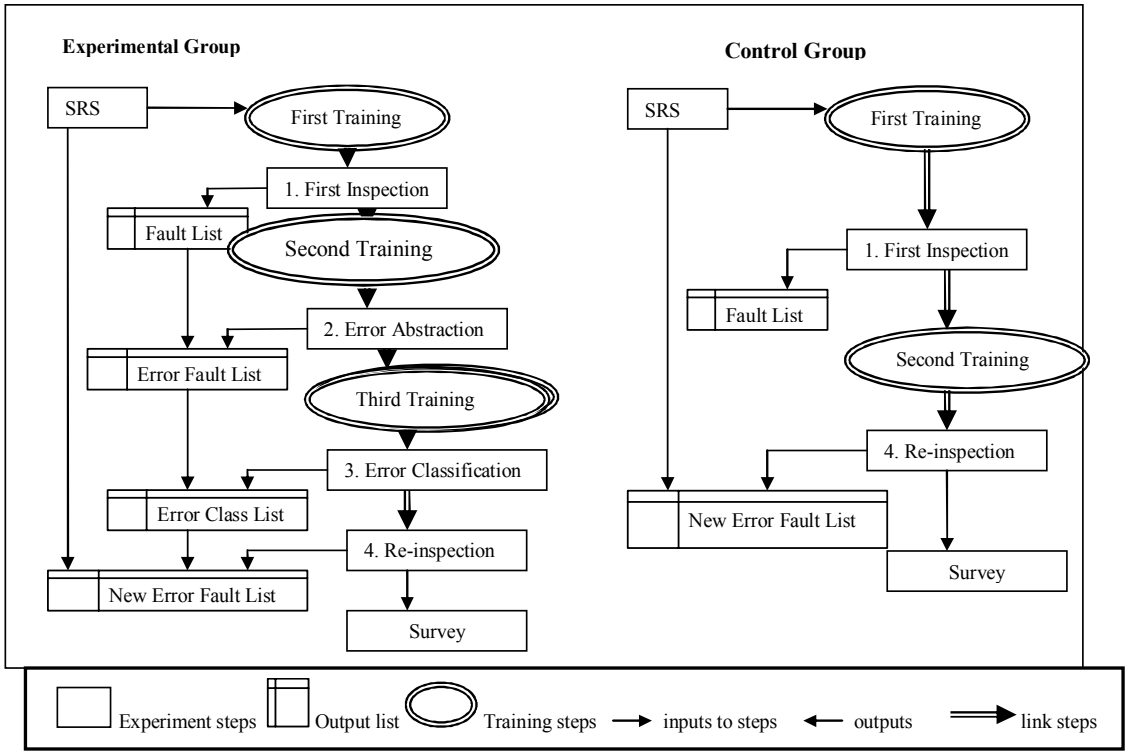

Figure 2. Experimental Procedure

classification, the subjects also recorded any additional errors they discovered while using the error taxonomy. The output of this step was 8 individual error-class lists (one per subject).

- *Step 4* – Locate additional faults: The subjects used the information about errors gathered during Step 3 to reinspect the requirements document to locate additional faults. This step is similar to Step 2 for the Control Group except that the experimental group had performed the error abstraction step. The output of this step was 8 individual new error-fault lists (one per subject).
- *Post-study Follow-up*: The subjects were given a questionnaire to provide feedback about the error abstraction process and the requirement error taxonomy. At the conclusion of the study, an in-class discussion was held with students from both groups to help the researchers (and subjects) better understand the results.

Table 1. Timeline

Experimental	Control	Time spent
Training 1	Training1	One week
First Inspection	First Inspection	
Training2	Training2	One week
Error Abstraction	Second Inspection	
Training3		
Second Inspection		

5. Analysis and Results

This section provides an analysis of the data collected during the study. This section is organized around the hypotheses presented in Section 4.1.1. An alpha value of 0.05 was selected for judging the significance of results.

5.1 Fault Detection Effectiveness and Efficiency (H1)

The effectiveness of the experimental and control groups were compared on the first inspection, the second inspection, and overall using an independent samples t-test. During the first inspection, the average effectiveness for the subjects was similar in both groups (experimental group – 22 faults; control group - 18 faults). Conversely, during the second inspection, the experimental group was significantly more effective than the control group finding an average of 17 faults compared to an average of 3 faults (p= .002). Considering the overall effectiveness (inspection 1 plus inspection 2), the experimental group was also significantly more effective than the control group finding an average of 39 faults compared to an average of 21 faults found for the control group (p= 0.044). These results are shown in Figure 3. In addition, the percentage increase in the number of faults was significantly higher for experimental group [76%] than the control group [17%] (p= 0.016). To reduce the error of getting a significant result by chance while doing the multiple tests, the Bonferroni correction was applied to get an alpha value of 0.013 (0.05/4) [9]. Analysis of the results with respect to the corrected alpha value indicates that the effectiveness when using the error abstraction and classification process is significantly higher than when simply performing two inspections of same document using fault checklist.

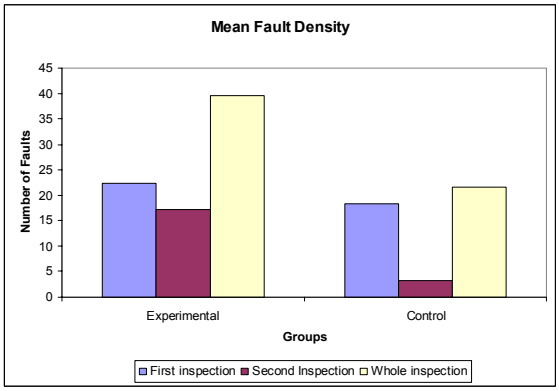

Figure 3. Comparison of Average Number of Faults for Groups

When examining the results for efficiency, there was less difference between the experimental and control groups. Three efficiency values for inspectors in both groups were computed: 1) during the first inspection alone (number of faults found at first inspection divided by time); 2) during the second inspection alone (number of new faults found divided by time); and 3) overall (total faults found at first and second inspection divided by the time spent during both inspections). There was not a significant difference in the efficiency for either the first inspection alone, the second inspection alone or overall. Even though the results were not statistically significant, the experimental group was more efficient on the second inspection and overall.

The lack of significant difference in efficiency seems to indicate that increased effort by the experimental group could be the cause of the increase in effectiveness. To analyze this potential factor, an analysis of covariance was performed. The results

showed that the time spent did not have a significant effect on the effectiveness of the two groups.

5.2 Usefulness of the Requirement Error Taxonomy (H2)

The requirement error taxonomy was evaluated using feedback from the subjects in experiment group on ten essential attributes: simplicity, understandability, usability, intuitiveness, orthogonality, usefulness, comprehensiveness, uniformity across products, adequacy of error classes, and ease of classifying errors. Each subject rated the attributes on a 5-point Likert scale (0 – Very Low, 1 – Low, 2- Medium, 3- High, or 4 – Very High).

A non-parametric binomial test was performed for each attribute to test whether its mean was significantly greater than 2 (the mid-point of the scale). The result was significant (p= 0.08) for the following attributes: usefulness, understandability, comprehensiveness, uniformity across products, adequacy of error classes, and ease of classifying errors. For these attributes, all subjects provided a rating of high or very high. For the intuitiveness attribute, seven subjects rated it as high or very high and one subject rated as moderate (p= 0.070). For the usability and orthogonality attributes, six subjects rated them as high or very high and two subjects rated them as moderate (p= 0.289). Therefore, the requirement error taxonomy was viewed favorably overall.

5.3 Insight Provided (H3)

The three high-level error types and fourteen detailed error classes (Figure 1) were analyzed to determine whether any of them were a major source of the errors and faults. Also, the major causes of redundant, time consuming, multiple, important and severe faults were analyzed. In each of these analyses, errors and faults were analyzed separately to determine if the effects were different.

5.3.1 Error Types vs. Error & Fault Density. Table 2 shows the distribution of errors among the three types. This information was gathered from the characterization of the errors on the error-class list provided by the subjects in the experimental group. People errors were the most common type of error. The p value from a chi-square test confirms that the three error types made significantly different contributions to the overall total (p = 0.052).

Table 2 also shows the distribution of the faults based on the type of error that caused the fault. This information was gathered from the error-class lists, error- fault list, and new error-fault list for the subjects in the experimental group. While people errors caused the highest percentage of errors, documentation errors actually led to more defects. Again, the results from a chi-square test confirms that different error types made significantly different contributions to the number of faults (p = 0.002).

Table 2. Contribution of Error Types

Variable	People	Process	Documentation
% Faults	41%	25%	34%
% Errors	35%	19%	46%

5.3.2 Major Causes of Redundant, Multiple, Time-Consuming, severe and important Faults. To better understand the impacts of the errors, they were analyzed in terms of the number of redundant, multiple, time consuming, severe, and important faults they caused. Because the subjects from the experiment group were the only ones who used the Requirement Error Taxonomy, on their data was used for this analysis. For each of these variables, a chi-square test was conducted to determine if the distribution of errors was significantly different from uniform. Table 3 shows the percentage contribution of error types to each variable along with the p-value from chi square test. The remainder of this section explains each row.

Table 3. Insights Provided by Requirement Error Taxonomy

Error Types	People	Process	Documentation	p-value
Redundant faults	52%	18%	30%	.004
Time-consuming faults	60%	13%	27%	.004
Multiple faults	44%	30%	26%	.002
Important faults	62%	22%	16%	< .001
Severe faults	53%	28%	19%	< .001

Redundant faults are faults that appear on the fault list of more than one subject. The data for this analysis was obtained at Steps 1 and 4 to identify redundant faults and the responsible error type. *Time-consuming faults* are faults that took longer than the average time (12 minutes) to be located. *Errors that cause multiple faults* are errors that have more than one fault abstracted to them. In addition, each subject classified the severity and importance of the faults found during first and second inspection. The *importance* attribute had five levels from zero (not important) to four (highly important). The *severity* attribute had four levels from zero (not severe) to three (will cause failure). For each of these variables, People errors

caused significantly more faults than the other error types (the p-values appear in Table 3).

Another important insight was whether the error types were responsible for particular classes) of faults or whether they led to all types of faults. A fault was classified into one of the following fault classes: General (G), Missing functionality (MF), Missing performance (MP), Missing interface (MI), Missing environment (ME), Ambiguous information (AI), Inconsistent information (II), Incorrect or extra functionality (IF), Wrong section (WS), and Other faults (O). Figure 4 shows the different types of faults that could be traced back to errors in each type. Each error class led to all types of faults. In addition, it appears that People errors most commonly led to faults of ambiguity while Process errors led to faults of ambiguity and missing functionality.

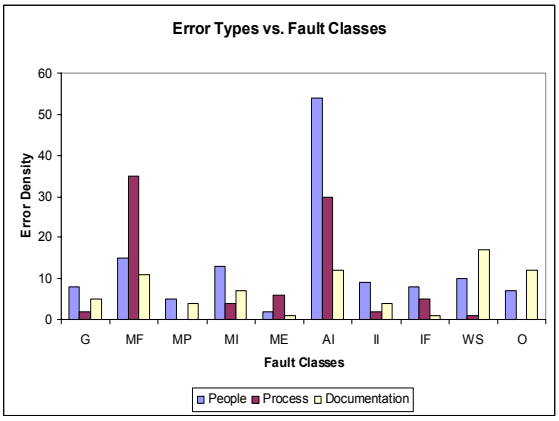

Figure 5. Cause of Fault Classes

5.4 Contribution of Human Cognition to Fault Density (H4)

One of the major contributions of the requirement error taxonomy was the integration of research from human cognition with research from software engineering. In order to understand whether the human cognition research made a meaningful contribution; the percentage of errors (from experiment group) classified into error classes related to human cognition research were analyzed. The types of errors that were related to human cognition were discussed in the previous study [12]. The results shown in Figure 5 indicate that each subject found errors related to human cognition. This result implies that the use of research from human cognition made an important contribution to the requirement error taxonomy.

5.5 Effects of Other Variables (H5)

This section provides analysis of the impact of the remaining independent variables on the dependent variables. The independent variables include: process conformance, pre-test performance, usefulness of training, amount of effort, and difficulty level. The dependent variables include: effectiveness and efficiency. Linear regression tests were performed to analyze the significance of the correlations between the independent and dependent variable as reflected in the p-values.

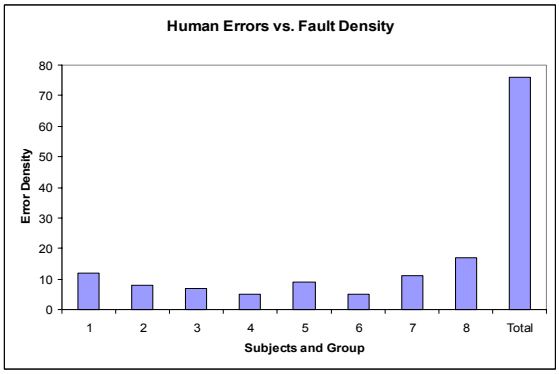

Figure 4. Contribution of Human Research

5.5.1 Process Conformance vs. Effectiveness. Each subject in the experimental group rated their process conformance from one to five on a 5-point Likert scale at four different activities (Step 1, 2, 3 and 4 in Figure 2). The mean value of these ratings was used as the overall process conformance value. The results in Figure 6 show that increased process conformance led to a higher number of defects found. This relationship was significant ($p = 0.041$).

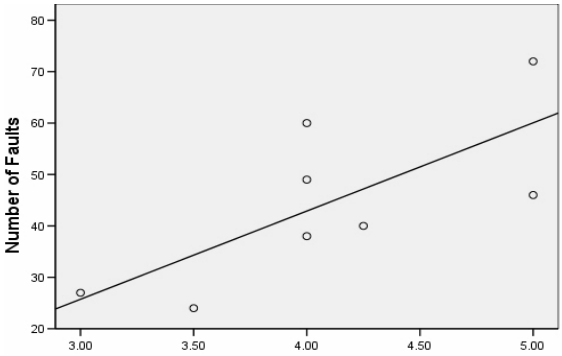

Figure 6. Process Conformance vs. Effectiveness

5.5.2 Effort vs. Effectiveness. Figure 7 shows that the overall effort expended (in hours) was correlated with the number of faults found. This result was significant ($p=0.044$).

5.5.3 Performance on Pre-test vs. Effectiveness. Finally, the number of errors correctly classified during

the pre-test was analyzed to determine its affect on the number of faults detected. The goal of this analysis was to understand whether performance on a pre-test could be an accurate predictor of performance on a real project. Figure 8 shows that there was a significant positive relationship between these two variables (p = .032). This result indicates that the practice run is a good predictor of performance.

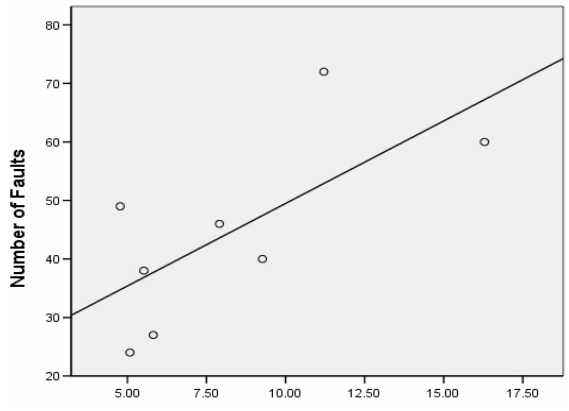

Figure 7. Effort Spent vs. Effectiveness

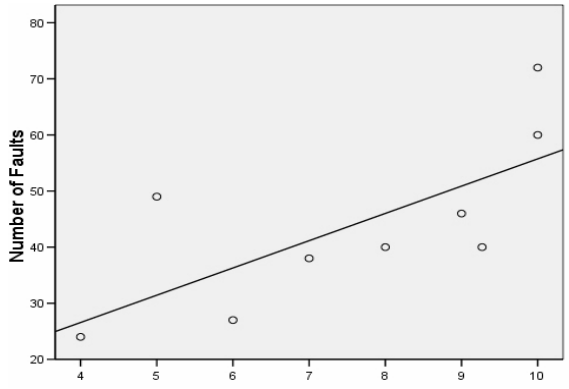

Figure 8. Performance vs. Effectiveness

6. Threats to Validity

In this study, there were some threats to validity that were addressed. In order to avoid a learning effect during the pre-test, the order of the errors being classified was randomized. In addition, to reduce the threat to external validity of using a toy requirements document, the subjects used a real requirements specification document. Finally, the lack of a control group in previous study was addressed by modifying design to include a control group.

However, there were some threats to validity that were not addressed by the experimental design. While the students were given a real requirements document, there were still some other threats to external validity. The study focused on students in a classroom setting who are likely to have different experience and time pressures than would be true of professionals in a real environment. A selection threat exists because the subjects were allocated to groups (experimental and control) based on the course they were enrolled in rather than randomly. Realizing that this threat was present, the course which was most closely related to the subject and therefore likely to perform better (Verification and Validation) was chosen to be the control group. This selection was made so that if any bias was present, it would in favor of the control group and not the proposed approach. The last threat that was unaddressed was maturation. The subjects in control group performed two inspections on the same requirements document using the same technique. While this arrangement provided the fairest comparison to the experimental group, it is likely that subjects were less motivated during the second inspection because they believed they had already found all of the problems.

7. Discussion of Results

In this section, the implications of the results from Section 5 are discussed for each of the original hypotheses posed in Section 4.1.1.

7.1 Hypothesis 1

> *The error abstraction process improves the effectiveness and efficiency for teams and for individuals.*

In terms of effectiveness, the results in Section 5.1 reveal that subjects in both the control group and the experimental group performed similarly during the first inspection. However, during the second inspection, the subjects using the error abstraction process were significantly more effective than the subjects who did not use the error abstraction process. In addition, the subjects that used the error abstraction process were significantly more effective overall than the subjects who did not use the error abstraction process.

In terms of efficiency, the subjects using the error abstraction process were more efficient (more faults per hour) than the subjects who did not use the error abstraction process, but this result was not statistically significantly different. Combining the results for effectiveness and efficiency shows that the use of the error abstraction process led subjects to find more defects and did not hurt their efficiency. Overall, it is a worthwhile use of effort considering the significant increase in the number of faults.

7.2 Hypothesis 2

The requirement error taxonomy is useful for improving software quality.

The results from Section 5.2 showed that in general the subjects rated the requirement error taxonomy favorably on some but not all of the attributes. Post-experiment, interviews, and discussion with the subjects from the experimental group revealed some insight into this result. The subjects indicated that abstracting and classifying errors was difficult because they were not involved in the development of the requirements document nor did they have access to anyone who was involved in the development. This is an area that must be addressed in future studies.

7.3 Hypothesis 3

The requirement error taxonomy provides important insights into the requirement phase of software development process.

The results from Section 5.3 showed that while the people errors were the largest source of errors, the documentation errors caused more faults. In addition, people errors are the major source of redundant faults, multiple faults, time-consuming faults, important faults, and severe faults. Therefore, people errors are an important and major cause of faults. The errors in the requirements error taxonomy led to many types of faults. Also, people errors were the primary cause of ambiguous information faults and process errors were primary cause of missing functionality faults, whereas the documentation errors were more evenly distributed among the fault types. These results provide confidence that the error classes in the requirement error taxonomy are valid and provide a good coverage of the requirements error space.

7.4 Hypothesis 4

The contribution of research from human cognition, psychology, and other fields helps locate more faults.

The results from Section 5.4 indicated that all the subjects in the experiment group found faults related to errors derived from human cognition research. The result supports the hypothesis and provides confidence that using research from fields like human cognition made a beneficial contribution to requirement error taxonomy. This result also establishes the validity of focusing on human errors in requirement phase of software process.

7.5 Hypothesis 5

Individual performance during the error abstraction process depends on various independent variables including: process conformance, performance on pre-test, usefulness of training procedure, effort applied, and difficulty level.

The results from Section 5.5 showed that process conformance, pre-test performance, and overall effort affect the number of faults detected by subjects. These results support the hypothesis and allow the following conclusions to be drawn: 1) to increase the number of faults detected, subjects must follow the process during error abstraction, classification and reinspection; 2) an increase in effort spent is likely to lead to an increase in the number of faults detected; 3) a subject's performance on a pre-test can be used to predict their effectiveness using the error abstraction process.

8. Comparison of Results

This section revisits the original hypotheses from Section 4.1.1 to compare the results from this study with those obtained during the previous study [12]. The results are compared around the standard set of hypotheses for both experiments. Results from the original study (study 1) and the new study (study 2) are summarized for each hypothesis in Table 4. In general, the results from the replication supported the results from the original study.

The error abstraction and classification process improved effectiveness in both cases, whereas the efficiency was improved in the replication only. Results from both studies indicate that it is easy to classify errors using the requirement error taxonomy. In both studies, People errors were an important source of problems. Furthermore, human cognition research could make a meaningful contribution to the analysis of requirement errors. Unlike the previous study, there was no significant relationship between effort and efficiency in the replication. Finally, in both studies, effectiveness depends on performance on the pre-test, process conformance and training.

9. Conclusion

Based on the results of the study, both the error abstraction process and the requirements error taxonomy are beneficial to developers who use them. The feedback provided by the subjects will be used to further refine and improve both processes to make them better for future studies. We plan to conduct more studies to empirically evaluate the error abstraction process and continue to refine both the error abstraction process and the requirement error

taxonomy based on the feedback from subjects. In future, we plan to replicate this study and other studies in different settings including an industrial setting. Our future work also includes creating a Design Error Classification Taxonomy using the same approach. Fault detection techniques will then be constructed based on the error taxonomies.

Table 4. Comparison of Results from Two Experiments

H#	Study 1: Factorial Design experiment	Study 2: Control Group Study
1	75% and 154% increase in faults when using the error abstraction and classification process but lowered the efficiency.	The error abstraction and classification process significantly improved the effectiveness, and improved efficiency (not significant)
2	The requirement error taxonomy was viewed favorably: it was easy to use and classify errors, understandable and well modularized.	The requirement error taxonomy was viewed favorably: it was well understood, useful, and uniform across products.
3	People errors are the major source of errors and faults.	People errors are major source of errors and faults.
4	There was a significant contribution from human cognition	There was a significant contribution from human cognition
5	Performance on pre-test, process conformance, and training affects effectiveness; and overall effort affect the efficiency.	Performance on pre-test, process conformance and overall effort affect the effectiveness.

10. Acknowledgements

We thank the students in Software Verification & Validation and Empirical Software Engineering courses at Mississippi State University for participating in this experiment. We acknowledge the Empirical Software Engineering group at Mississippi State University for providing useful feedback on the research and this paper. This work was supported in part by the Office of Research at Mississippi State University.

11. References

[1] Boehm, B., and Basili, V.R. "Software Defect Reduction Top 10 List," IEEE Computer, 34 (1):135-137, January 2001.

[2] Chaar, J., Halliday, M., Bhandari, I., and Chillarege, R., "In- Process Evaluation for Software Inspection and Test," IEEE Transactions on Software Engineering, 19(11):1055-1070, November 1993.

[3] Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., Ray, B., and Wong, M.Y., "Orthogonal Defect Classification - A Concept for In-Process Measurements," IEEE Transactions on Software Engineering, 18(11):943-956, November 1992.

[4] Damele, G., Bazzana, G., Andreis, F., and Aquilio, S., "Process Improvement through Root Cause Analysis," In Proceedings of the 3rd International Conference on Achieving Quality in Software, Florence, 1996, p.35-47.

[5] Florac, W. A., "Software Quality Measurement: A Framework for Counting Problems and Defect," Technical Report, Carnegie Mellon Software Engineering Institute, Pittsburg, PA., CMU/SEI-92-TR-22, 1992.

[6] Lanubile, F., Shull, F., Basili, V.R., "Experimenting with Error Abstraction in Requirements Documents," In Proceedings of 5th International symposium on software metrics, 1998, Bethesda, MD, USA:IEEE Computer Society.

[7] Lezak, M., Perry, E.D., Stoll, D., "A Case Study in Root Cause Defect Analysis," In Proceedings of the 22nd International Conference on Software Engineering, 2000, Limerick, Ireland.

[8] Software Engineering Laboratory: Software Measurement Guidebook, 1994, NASA/GSFC Software Engineering Laboratory, Technical Report SEL-94-002, 1994.

[9] Miller, R.G., "Simultaneous Statistical Inference," 2nd Edition, Springer Verlag, New York, pp. 6-8, 1981.

[10] Porter, A.A., Votta, L.G., Basili, V.R., "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," IEEE Transactions on Software Engineering, 1995, 21(6): 563-575.

[11] Walia, G.S., "Empirical Validation of Requirement Error Abstraction and Classification: A Multidisciplinary Approach," M.S Thesis, Mississippi State University, MS, 2006.

[12] Walia, G.S., Carver, J., Philip, T., "Requirement Error Abstraction and Classification: An Empirical Study," In Proceedings of the 5th International Symposium on Empirical Software Engineering, Rio de Janeiro, 2006, pp. 336-345.

[13] Walia, G.S., Carver, J., "Development of Requirement Error Taxonomy as a Quality Improvement Approach: A Systematic Literature Review," Department of Computer Science and Engineering, Technical Report, MSU-070404, Mississippi State University, 2007.