

# Requirement Error Abstraction and Classification: An Empirical Study

Gursimran S. Walia, Jeffrey Carver and Thomas Philip

Mississippi State University  
300 Butler Hall, Box 9637  
Mississippi State, MS 39762  
+1 662-325-8798

{gw86, carver, philip}@cse.msstate.edu

## ABSTRACT

Software quality and reliability is a primary concern for successful development organizations. Monitoring and controlling quality by helping developers detect as many faults as possible is a subjective and intricate approach. Due to the inherent difficulties and limitations, additional methods are required to obtain a more complete solution to the software quality problem. This paper analyzes the software quality problem from a different perspective involving a step back from faults to focus on the fundamental causes of faults. The first step in this direction is the application of the Error Abstraction Process (EAP) to the requirements phase of the software lifecycle to develop a Requirement Error Taxonomy (RET). This paper presents an empirical study on the application of the EAP and RET to requirement documents in a controlled classroom setting. The results show that the EAP significantly improves the productivity of subjects, that the RET is useful for improving software quality, that it provides useful insights into the requirements document, and that various context variables also impact the results. These results are promising and are important to motivate further investigation, to refine the RET, and to derive more formalized tools and methods for assisting developers. The result of this investigation will be a sound verification process for requirements phase.

## Categories and Subject Descriptors

D.2.1 [Software Engineering]: Requirements/Specifications – tools.

## General Terms

Documentation, Experimentation, Human Factors, Verification

**Keywords:** Error, Error Abstraction, Inspection, Empirical Study, Root Cause Analysis

## 1. INTRODUCTION

Over the years, researchers have investigated the software quality problem [1, 5]. Different software quality improvement approaches have been developed and evaluated through

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISESE'06, September 21–22, 2006, Rio de Janeiro, Brazil.

Copyright 2006 ACM 1-59593-218-6/06/0009...\$5.00.

experiments in controlled and real settings [3, 4, 7, 9, 11]. The focus of this research has been on extracting semantic information from faults to allow developers to locate defects early and improve software quality. Researchers have examined the cause-effect relationship of faults and their impact at different phases of the software lifecycle. In addition, different fault classification taxonomies have been developed and evaluated [2, 4, 10].

While these fault classifications have proven beneficial, there is still room for additional methods to fill in some of the remaining gaps for more complete and sound verification process. Quantifying, classifying and locating an individual fault is a subjective and intricate notion, especially during the requirements phase [8]. In addition, some of the existing fault classification taxonomies have been criticized for their inability to satisfy essential attributes [4, 9]. These fault classes have also been used in quality measurement and improvement frameworks [7, 11]. However, because the fault information fails to target the underlying cause, these frameworks were not as effective as they could have been. Focusing only on faults, that is, mistakes that are recorded in a document will not be sufficient to eliminate the underlying mistakes that can lead to multiple faults and failures.

A similar notion is the quality improvement approach called Root Cause Analysis (RCA), which focuses on analyzing the causes of faults [6]. However, due to the expenses incurred, the substantial investment of resources and the large number of actions that result, RCA has not found widespread success. Building on this idea, the Orthogonal Defect Classification (ODC) was used to enable in-process feedback to developers. However, the ODC also focuses on using semantic information from only the faults to extract a cause-effect relationship in the development process [4].

The goal of this research is to push RCA further by extracting error signatures in a similar fashion as done by ODC. This process should help developers get to the root of a fault more quickly and understand the real problems in software development. Stated more formally, the goal is:

*To analyze the underlying causes of faults and to provide developers with methods for identifying, and classifying the errors to improve software quality.*

The underlying concept of this work is that addressing the quality problem using error information will provide better results than using only fault information. To that end, previous research has provided some empirical evidence that using an error abstraction process in analyzing a requirements document may be a useful approach [8]. In this original work, the process of error abstraction

was not guided by an error classification taxonomy. This current work builds on that work by quantifying the process of error abstraction using an error classification taxonomy that contains information about the cause- effect relationship between errors and software quality.

Previous research has shown that the software engineering literature alone is not sufficient to address all the errors that can occur throughout software development. Case studies report that human reasons, e.g. reasons not directly related to software engineering, also contribute to fault injection [10, 14]. Since software development is a human-based activity, it is useful to investigate various phenomenon associated with the human mental process and its fallibilities. Thus, one challenge is integrating the relevant contribution from human cognition into the error abstraction process. To accomplish this integration, the potential contribution of human cognition errors must be analyzed.

Section 2 describes the proposed approach, followed by a description of an experiment conducted to evaluate the approach in Section 3. Section 4 describes the analysis and results of study. Section 5 describes the threats to validity followed by Section 6 and 7 that talk about the relevance of the results and conclusions respectively.

## 2. PROPOSED APPROACH

The proposed approach consists of using *error abstraction* to help developers determine the underlying errors from faults a *requirement error classification taxonomy* to locate more errors and faults.

### 2.1 Error Abstraction Process (EAP)

The process of error abstraction consists of analyzing the nature of faults and determining the underlying mistakes, or errors, responsible for those faults. Inspectors are trained on how to abstract errors from the faults to help them understand their actual mistakes, following a process as used in the earlier study [8]. Multiple faults may be abstracted to different underlying mistakes or to the same mistake depending on the nature of the information involved in those faults. In addition, a single error can lead to multiple faults. So, understanding the errors that occurred when creating the requirements document can lead the inspector to locate additional faults, related to those errors

However, during the process of abstracting errors, there is the potential that all errors may not have been located. So, to augment the earlier error abstraction work a Requirement Error Taxonomy (RET) has been developed to help inspectors do a more thorough job of locating errors and their corresponding faults that may have been overlooked before.

### 2.2 Requirement Error Classification Taxonomy (RET)

The RET was developed using the EAP to analyze and abstract as many kinds of requirement errors that could lead to faults and failures as possible. Relevant information in different sources including software engineering literature, fault classification taxonomies, and historical data from previous projects was surveyed to develop the abstraction. An additional important source that was used to incorporate the need to understand the contribution of human cognition errors was research from the

fields of human cognition, psychology and failure management. Human error taxonomies and other relevant human cognition related fallibilities were analyzed to make the list of abstracted requirement errors as comprehensive as possible. A complete description and list of the references examined can be found in Walia's thesis [15].

The next step was to group together similar types of errors from this list of abstracted errors to develop a *Requirement Error Classification Taxonomy* (RET). The errors were grouped into three high-level categories: People Errors, Process Errors and Documentation Errors. *People Errors* include mistakes that originate with the fallibilities of the individuals involved in the project development, *Process Errors* are caused by mistakes in selecting the means of achieving goals and objectives and focus mainly on the inadequacy of requirements engineering processes, and *Documentation Errors* occurs due to mistakes in organizing and specifying the requirements irrespective of whether the developers understood the requirements.

Then, each high-level error type was refined into some more detailed error classes, as shown in Figure 1. Each error class explains similar kinds of errors grouped together to help developers understand the basic symptoms of that error class. In addition, the RET also explains the faults likely to be caused by the errors in each error class. The RET was developed to maintain orthogonality and simplicity while still being comprehensive and intuitive.

The final step was to map the RET back to the relevant human error taxonomies to ensure completeness and provide additional confidence. The RET describes each type of error in detail and traces forward the impact, the error is likely to have on software quality in terms of the faults that can result from it.

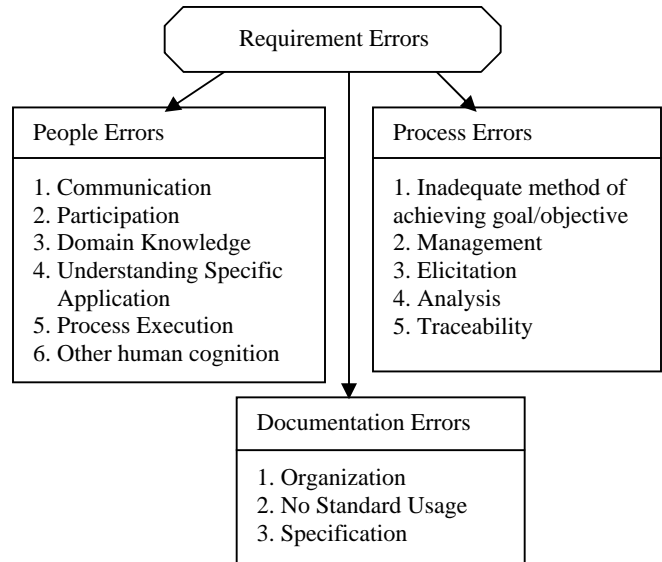


Figure 1: Requirement error classification

The RET aids developers by providing a comprehensive list of errors classified in such a way that helps them check different errors and their corresponding faults at their source thereby reducing the rework effort. It is likely that with an understanding of errors, developers may see additional faults they overlooked

during the first inspection. Thus, it was important to determine whether using the RET in EAP helped developers to reach to actual causes of faults efficiently.

This paper describes the evaluation of the RET in terms of its usefulness in improving the productivity of developers and the amount of insight it provided to the developers. The RET was characterized in terms of eight essential attributes: simplicity, understandability, applicability, intuitiveness, orthogonality, comprehensiveness, usefulness and uniformity across products. A more complete description of the RET and the process for developing it can be found in Walia's thesis [15].

### 3. EXPERIMENT DESIGN

The goal of the experiment was to investigate the usefulness of the EAP and RET when analyzing a real software requirements specification. Primarily, the focus was on whether the EAP and RET improve productivity of individuals and teams.

#### 3.1 Methodology

##### 3.1.1 Hypotheses:

The following set of hypotheses was posed for this study:

**Hypothesis 1:** The EAP improves the effectiveness and efficiency for teams and for individuals.

**Hypothesis 2:** The RET is useful for improving software quality.

**Hypothesis 3:** The RET provides significant insights into the requirement phase of software development process.

**Hypothesis 4:** The contribution of the research from human cognition, psychology and other fields helps locate more faults.

**Hypothesis 5:** Individual performance during the EAP depends on various independent variables including: process conformance, performance on practice run, usefulness of training procedures, and effort applied.

##### 3.1.2 Subjects

The subjects of this study consisted of sixteen (16) senior-level undergraduate students, majoring in either computer science or software engineering enrolled in the Software Engineering Senior Project course at Mississippi State University in the Fall 2005 semester. The course required students to interact with real customers, elicit and document requirements that they would later implement. The subjects were divided into two teams of eight subjects each (by the course instructor, outside the scope of this study). Each team developed a different system.

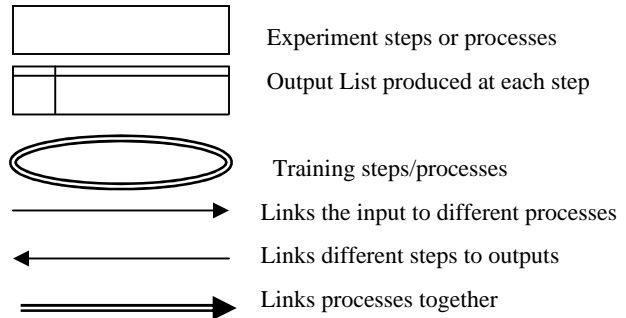
##### 3.1.3 Artifacts

The two teams were developing real systems. Each team interacted with their customers to develop the software requirement specification for their system. Team 1 (T1) worked on a system for managing ticket sales and seat assignments for the Starkville Community Theatre. Team 2 (T2) worked on a system

for managing apartments and town homes properties, including assignment of tenants, rent collection and location of properties by potential renters.

### 3.2 Experimental Procedure

The study consisted of 5 steps and 2 training lectures. The remainder of this section describes each of those steps in detail. Figure 2 provides an overview of the experimental procedure, using the following notation:



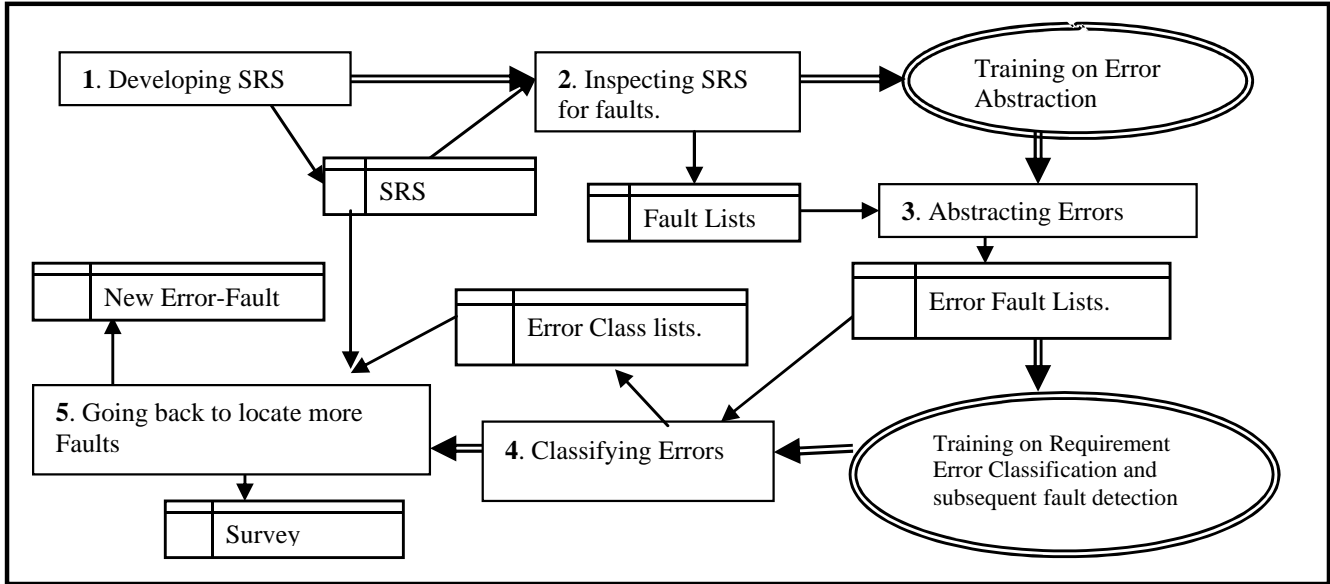
*Step-1 Developing SRS:* In this step, each group interacted with its customer to develop a requirement specification document.

*Step-2 Inspecting SRS for faults:* When the SRS was complete, each student inspected it individually recording any faults detected. To conduct this inspection, each subject used a simple fault checklist. After all team members had conducted their individual inspections, they met together to consolidate their individual fault lists and agree on a team fault list. This step produced 16 **individual fault lists** (1 per subject) and 2 **group fault lists** (1 per team).

*Training 1- Error Abstraction:* During this 40 minute session, the subjects were trained on the EAP and how to use it on their individual fault lists to abstract the underlying errors.

*Step-3 Abstraction of Errors:* After the training, the subjects returned to their individual fault lists to abstract the underlying errors. These errors were documented in an Error-Fault List. The output of this step was 16 **individual error-fault lists** (1 per subject).

*Training 2- Requirement Error Classification:* This 120 minute session focused on the RET and its use. During the training, the RET was explained in detail. The students were then given a description of some fictitious systems along with a list of 12 errors to classify using the RET. The goal of this exercise was to ensure that the students understood the error classification process by using it to classify these errors before using it on their own SRS documents. 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 different lists (list A and list B) were prepared containing the same set of errors but in different orders and gave half of the subjects each list.



**Figure 2: Description of the experimental procedure**

*Step-4 Classification of Errors:* After the second training, the subjects returned to their individual error-fault lists from Step 3 to classify those errors into the RET. While doing this classification, the subjects were to record any additional errors they discovered as a result of using the RET. The output of this step was 16 **individual error-class lists** (1 per subject).

*Step-5 Going back to locate more faults:* Finally, the students used the information in the RET about each error on their individual error-class lists to re-inspect the SRS to locate any additional faults that may be related to that error. Each student developed a new **individual new error-fault list** (1 per student). Also each team developed its **team new error-fault list** (1 per team) and mapped it with their team **Error-Class list** (1 per team).

After completing all of these steps, each student completed a questionnaire to provide feedback regarding the EAP and RET.

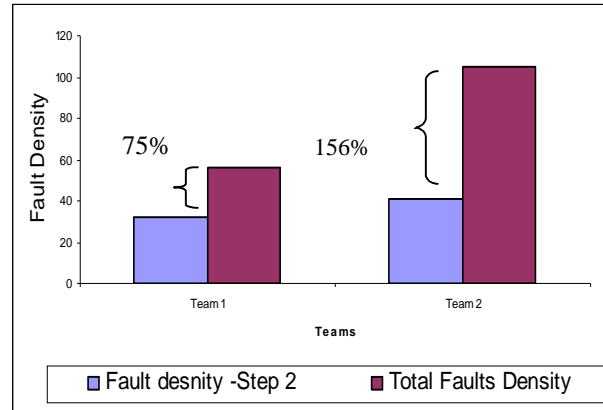
#### 4. ANALYSIS AND RESULTS:

This section provides an analysis of the data collected during the study. The data is organized based on the hypotheses defined in Section 3.1.1. Section 6 provides an interpretation of the results in light of the overall study goals. This analysis uses an alpha value of 0.05 for significance tests.

##### 4.1 Fault Detection Effectiveness and Efficiency (H1):

It was important to first determine whether the EAP provided any increase in effectiveness or efficiency over the standard fault checklist process. To address effectiveness, the number of faults detected during the first inspection (prior to learning about the EAP) was compared with to the number of faults detected during the re-inspection (after learning EAP). Figure 3 compares the number of faults detected during Step-2 of experiment process (before EAP) and the total faults found (including the faults found after using EAP). These results show that there was a 75% and 154% increase in the fault detection density provided by the EAP for T1 and T2 respectively.

To further investigate the large increase in the team fault detection density, the performance of the individual subjects was analyzed to determine if the increase was consistent throughout the sample or concentrated in only a few subjects.



**Figure 3: Comparison of effectiveness of both teams.**

Figure 4 shows the percentage increase for by each subject (the 8 subjects from T1 are on the left and the 8 from T2 on the right). As the figure shows, all subjects displayed some increase when using the EAP, while those on T2 seem to be more evenly distributed.

To verify this observation a one-sample t-test was run separately for each team. The goal of this analysis was to determine whether the number of faults found during the re-inspection was significantly different from zero (0). The results of this test show that the fault detection for both teams was significantly higher than zero ( $p= 0.02$  [T1];  $p= 0.00$  [T2]). To address efficiency, the impact of the EAP on the fault detection rates of T1 and T2 was analyzed (as shown in Figure 5). Because of the multiple steps involved in the process, it was not clear how to arrive at an effort figure to use for comparison.

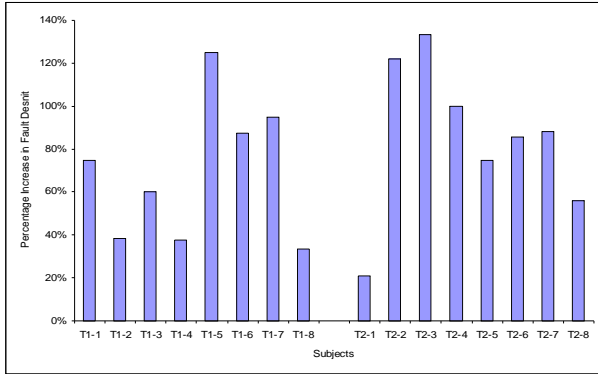


Figure 4: Increase in effectiveness for all subjects

Therefore, Figure 5 compares 3 fault rates for each team. The value for A is computed by dividing the number of faults found during the pre-EAP inspection by the number of hours that inspection took (Step 2). The value for B is computed by dividing the number of faults found during the re-inspection after using EAP (Step 5) divided by the time taken only for the re-inspection (Step 5). The value for C is computed by dividing the number of faults found during the re-inspection after using EAP (Step 5) divided by the total time for error abstraction, error classification and re-inspection (Steps 3, 4 and 5).

As Figure 5 shows, there was a large increase in detection rate when only considering the effort for re-inspection. Conversely, when taking into account all of the effort associated with the EAP, the fault detection rate is slightly lower than the fault detection rate using the standard fault checklist method. However, this small decrease is made up by the benefit of the additional faults detected and is therefore a worthwhile use of effort.

We also used a one-sample t-test for comparing the difference in fault rates of A and C separately for each team. Results of this test show that the difference in the fault rates of A and C is not significant for either team. Thus, there is no significant difference in the fault rates of fault checklist technique and EAP.

#### 4.2 Usefulness of RET (H2)

The RET was evaluated using feedback from subjects on eight essential attributes: simplicity, understandability, applicability, intuitiveness, orthogonality, comprehensiveness, usefulness and uniformity across products as shown in Figure 6. For each attribute, the subjects rated the RET as 1 – Low, 2 – Medium, or 3 – High. Figure 6 shows the distribution of those rankings for the 16 subjects. The results show that, in general, the RET was viewed favorably for all attributes. There are no cases where an attribute received more low ratings than high ratings.

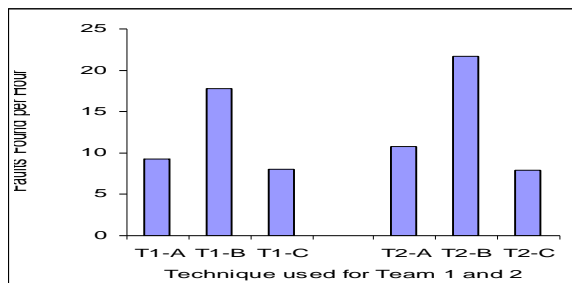


Figure 5: Comparison of efficiency of both teams.

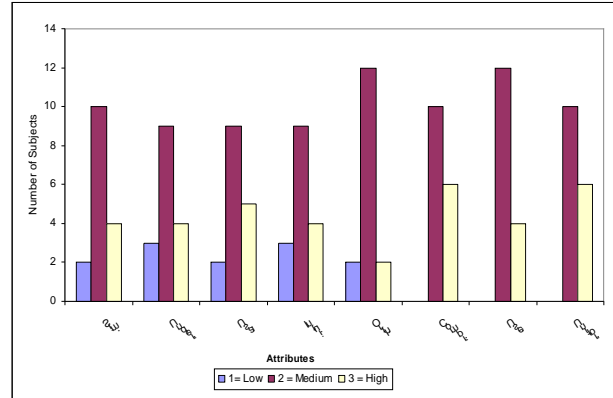


Figure 6: Frequency distribution of weights.

Furthermore, the subjects were asked to report on the adequacy of the error classes for the given task and the ease of placing errors in the appropriate error class. Figure 7 summarizes the responses of the subjects using the same scale as in Figure 6. Only one subject gave the adequacy of error class descriptions a low weight.

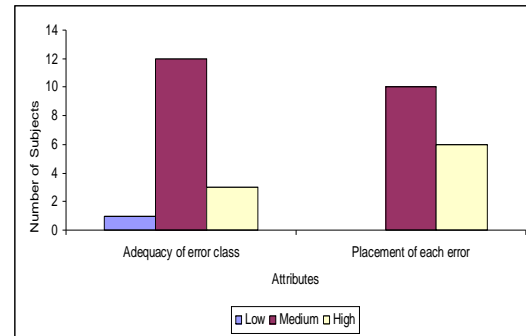


Figure 7: Adequacy of error classes.

In addition to the subjects' qualitative evaluation, their error-class lists were analyzed to determine whether there were any errors that could not be classified in the RET. This analysis showed that all errors could be classified. The qualitative and quantitative data together provide sufficient evidence to believe that the RET was complete and easy to use.

#### 4.3 Insight Provided (H3)

The 3 high-level error types and 14 detailed error classes (Figure 1) were analyzed to determine their contribution towards improving the quality of the software. Two important issues were whether the errors and faults were evenly distributed among the error types and error classes as well as whether any particular type was a major cause of redundant, time consuming or multiple faults. For these analyses the errors and the resulting faults were analyzed separately to determine if the effects of the error types and classes were any different.

##### 4.3.1 Error Types vs. Error & Fault Density

Figure 8 and Figure 9 compare the percentage contributions of the three high-level error types, People (Peo), Process (Pro) and Documentation (Doc) to the total error density for Team 1 (T1) and Team 2 (T2) respectively. These percentages were computed using each team's error-class list to count the number of errors of each type. Graphically the distribution shows that people error

contributed to most of the errors for both teams. However, an ANOVA test did not show a significant difference indicating that the errors were evenly distributed across the three error types.

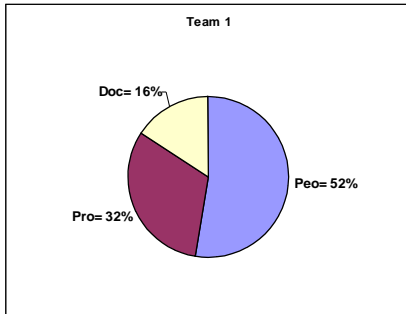


Figure 8: Contribution of error types to error density –T1

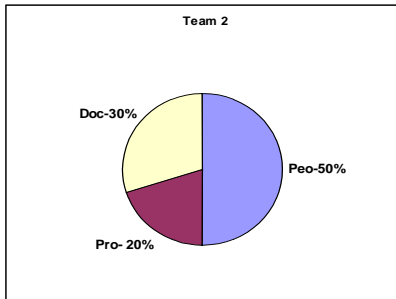


Figure 9: Contribution of error types to error density-T2

Similarly, the contributions of error types to fault injection density were analyzed for Team 1 and Team 2 in Figure 10 and 11 respectively. To perform this analysis, each team’s error-class lists and error- fault list from Step 3 and error-fault list from Step 5 were used to determine all the faults found.

Again, the distributions in Figure 10 and 11 reveal that people errors cause the highest number of faults for both teams. However, the results of the ANOVA test support are similar to the results for error density in that no significant difference was found.

#### 4.3.2 Error Classes vs. Error & Fault Density

In addition to 3 high-level error types, the contribution of 14 more detailed error classes was examined. Figure 12 shows the percent of errors contributed by each of 14 error classes (grouped by their high-level error type). Results of an ANOVA test shows that while the contribution to error density among the 14 error types varies, the difference is not significant. Figure 12 show that there was at least one error from each error class in the taxonomy indicating that all of the error classes are necessary. Therefore a conclusion can be drawn that all 14 detailed error classes are important.

Similarly, for fault detection, Figure 13 shows the percent of faults that were caused by each of the 14 error classes. The results of the ANOVA test again show that there is no significant difference among the contribution of the 14 error types to the overall fault density. Similar to errors, Figure 13 shows that there is at least one fault caused by an error from each class. This result gives more evidence to the validity of the error classes.

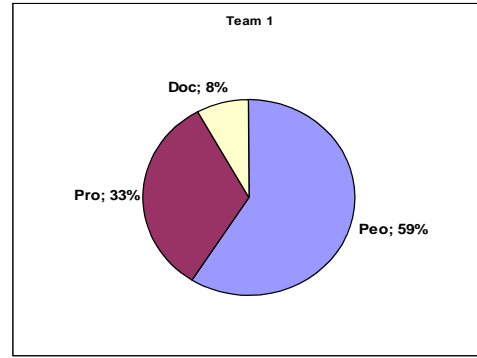


Figure 10: Contribution of error types to fault density-T1

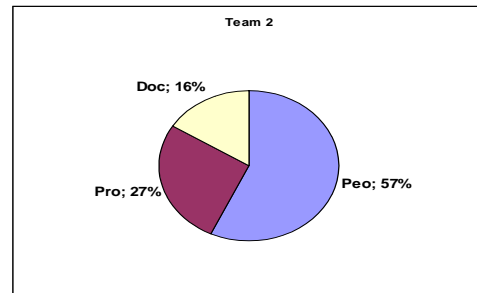


Figure 11: Contribution of error types to fault density-T2

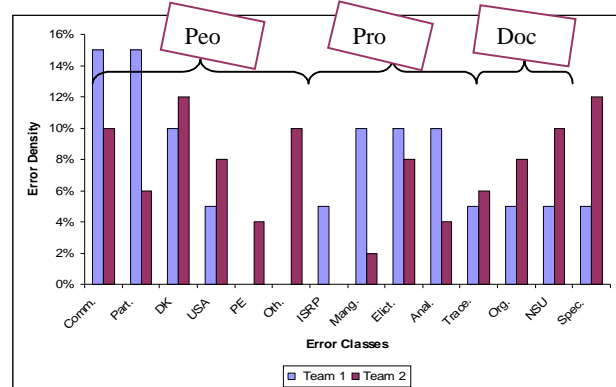


Figure 12: Contribution of error class to error density

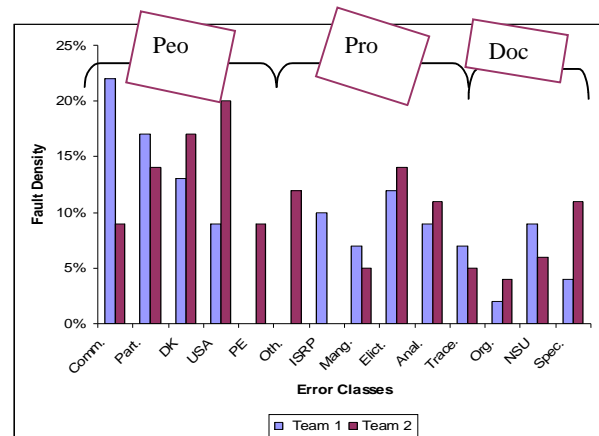


Figure 13: Contribution of error class to fault density

### 4.3.3 Major Causes of Redundant, Multiple and Time Consuming Faults

Another important insight into the RET was whether any of the three high-level error types was responsible for a large number of redundant faults (i.e. faults that appeared on more than one team member's fault list during Step 2 and Step 5). This information helps determine where improvements could be made to reduce the overlap.

Figure 14 compares the contribution of each error type for both the teams. The data shows that people errors have a higher contribution than other error types for both teams. This result was statistically significant using an ANOVA ( $F_{2,5} = 9.700, p = 0.049$ ). Thus, the difference between mean contributions of error types is significant. An additional post-hoc test (the Tukey test) showed that the mean difference between People and Documentation errors is also significant ( $p = 0.048$ ).

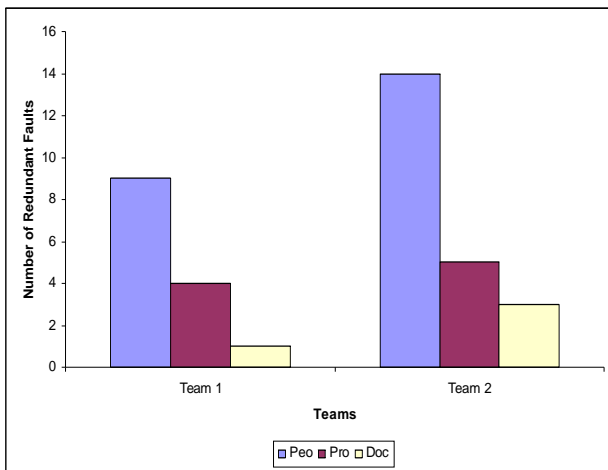


Figure 14: Comparing causes of redundant faults

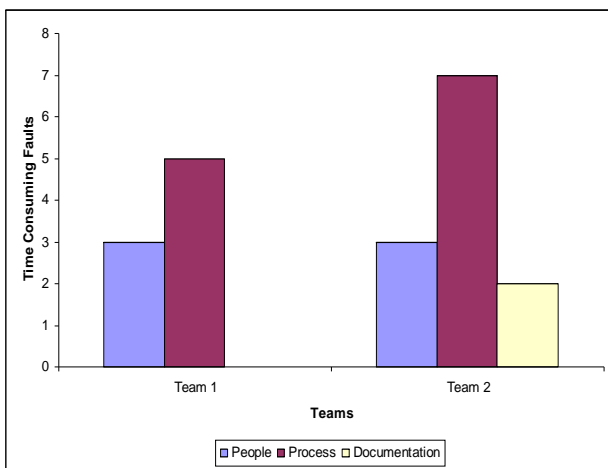


Figure 15: Comparing causes of time consuming faults

Figure 15 shows the contribution of error types to most time consuming faults for both teams. The Team "Fault lists" produced during Step 2 were used to determine the time consuming faults. Also, the consolidated "Error-Fault" lists for each team produced at Step 5 were used to find additional time consuming faults. The time consuming faults were then analyzed to determine their major cause

for both teams. A time consuming fault was defined as a fault that took greater than or equal to 15 minutes to find, which was greater than the average time taken for find most of other faults. The Process errors produced the most time consuming faults compared with the other error types for both teams.

Results from the ANOVA test show that the difference between mean contributions of error types is significant ( $F_{2,5} = 9.500, p = 0.05$ ). Also, the post-hoc Tukey test showed that the mean difference between the Process and Documentation error types was significant ( $p = 0.046$ ).

Another important aspect of the errors is whether an error is responsible for multiple faults. We wanted to determine if errors from any of the three high-level error types were more likely to result in multiple faults. To perform this analysis, we examined the team fault lists and the team error-fault lists, produced in Steps 2 and 5 respectively. Figure 16 shows the results of this analysis. The results showed that People errors were more likely than the other two types to cause multiple faults. This difference is not significant with our alpha value of .05 when evaluated with an ANOVA ( $F_{2,5} = 8.167, p = .061$ ).

### 4.4 Contribution of Human Cognition to Fault Density (H4)

One of the contributions of the RET was the use of research from human cognition to augment the error classes developed from software engineering resources alone.

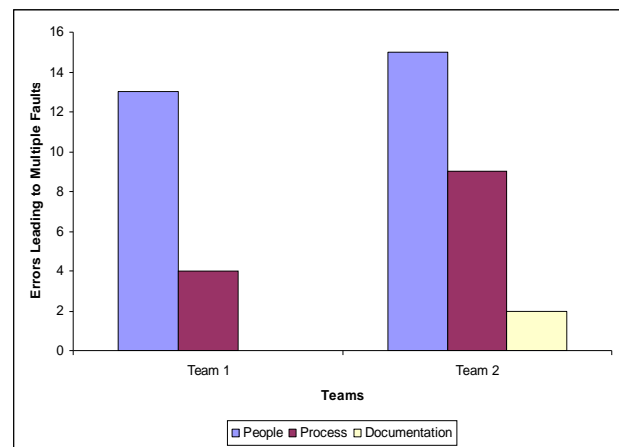


Figure 16: Comparison of causes of multiple faults

In order to understand whether the human cognition research made a meaningful contribution to the RET; the errors were analyzed to determine the percentage of errors from each team that came from detailed classes that were created based on the human cognition research.

The results, shown in Figure 17, indicate that a meaningful contribution was made by human cognition errors for both teams (20% and 25%).

This result indicates that the use of research from these fields was effective in bolstering the RET. Furthermore, Figure 18 breaks this result down by individual subject to show that in most cases each subject found errors that were related to human cognition, with one subject finding only human cognition-related errors.

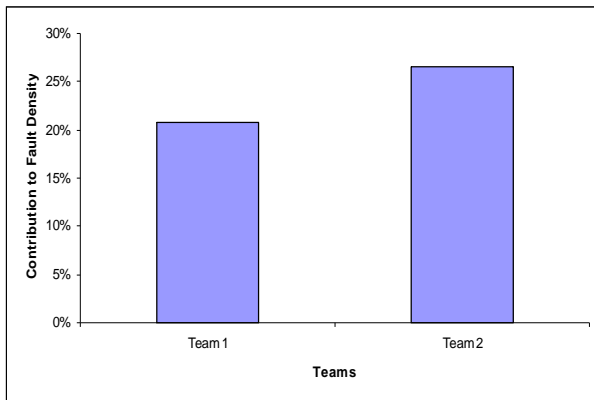


Figure 17: Contribution of human research

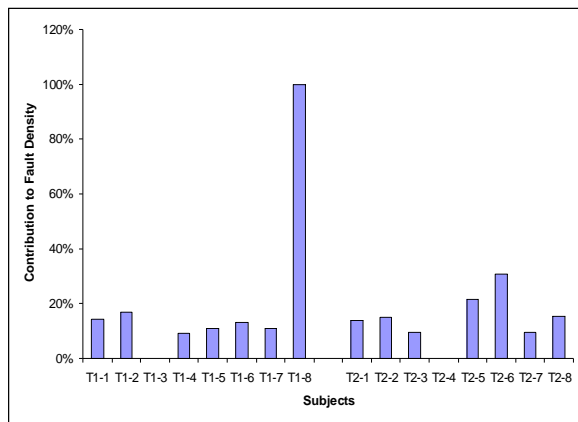


Figure 18: Contribution of human reasons for subjects.

## 4.5 Effects of Other Variables

Finally, the effects of different independent variables on the productivity of subjects were analyzed. The independent variables analyzed were: degree of process conformance, performance on practice run, usefulness of training procedure, and effort applied.

### 4.5.1 Process Conformance vs. Fault Detection Density

The effect that degree of process conformance had on individual fault density was analyzed first. The process conformance data came from qualitative data collected at three different points during the experiment (Step 3, 4 and 5 in Figure 2) as well as their median value. Results show that the increase in fault density only correlates to the process conformance during error abstraction process (Step 3) and median process conformance with the p-values of 0.004 and 0.030 respectively.

### 4.5.2 Effort vs. Fault Detection Rate

Figure 19 shows the relationship between effort expended during the EAP and individual fault detection rate. The effort value was computed using the sum of the hours spent during the three parts of the experiment procedure (abstraction, classification, and re-inspection) and plotted against fault rate for each subject.

The trend line indicates that there is a significant positive relationship between effort and fault rate ( $p = .02$ ). Thus, it shows that effort spent helps increasing fault rate of developers.

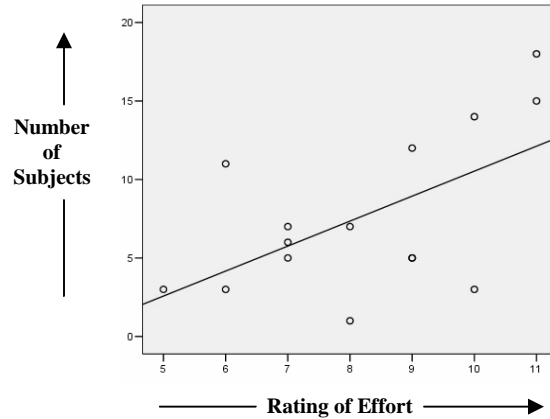


Figure 19: Effort Vs Fault detection rate

### 4.5.3 Usefulness of Training vs. Fault Detection Density

Two training sessions were conducted with the students. The first one focused on abstracting errors from faults (Training 1). The second focused on RET (Training 2). In order to understand the relationship between how useful the students viewed the training and their subsequent fault detection density, each training session was evaluated separately.

The analysis showed a significant relationship for Training session 1 (the EAP) with  $p = .04$ , but not for Training session 2 (the RET). This result indicates that understanding the process of error abstraction was more important than understanding the classification scheme. This result is shown graphically in Figure 20.

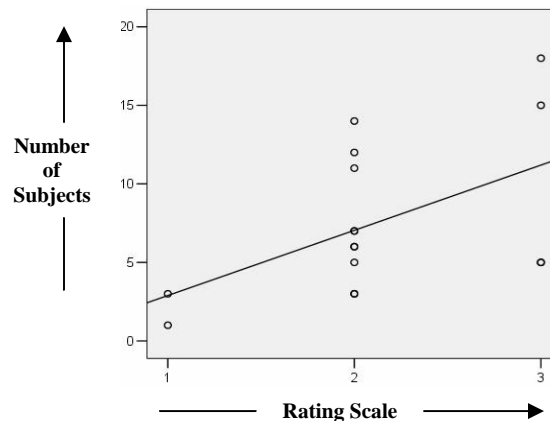


Figure 20: Training 1 Vs Fault density

### 4.5.4 Performance on Practice Run vs. Fault Detection Density

Finally, the students' performance on a classroom exercise of classifying example errors during Training session 2 was compared with their subsequent fault detection density. The goal of this analysis was to understand whether performance during a practice run could be an accurate predictor of performance on a real project. To conduct this analysis, the number of errors each student correctly classified was compared with their fault detection density. Figure 21 shows that there was a significant positive relationship between

these two variables ( $p = .024$ ). This result indicates that the practice run is a good predictor of performance.

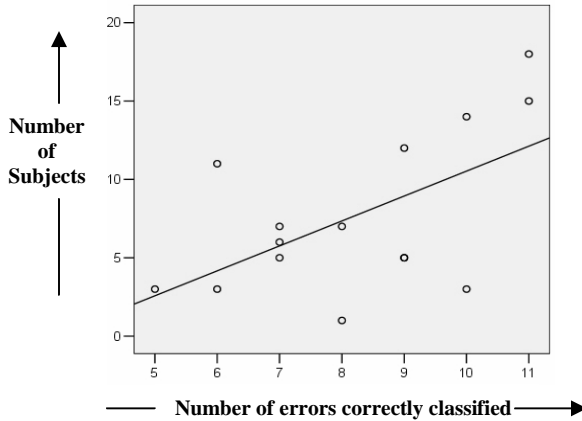


Figure 21: Performance Vs Fault density

#### 4.6 Analysis of Qualitative Data

In addition to the quantitative data, the students provided qualitative feedback about the usefulness of error abstraction process. Figure 22 shows the students rating of the EAP on five different attributes: CIU - Confidence in using error abstraction process, MOR - Meaningfulness of results, UIURP - Usage in understanding real problems, WOES - Worthiness of effort spent, and CERM - Confidence of errors being real misunderstandings. For these attributes, the students had moderate to high opinion of the EAP. This graph shows that in general the students rated the EAP positively for each of these attributes.

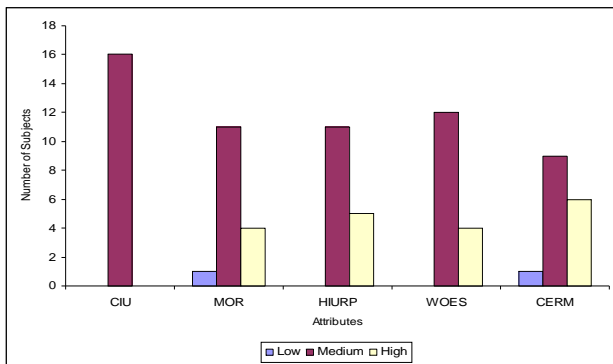


Figure 22: Frequency distribution for EAP

#### 5. Threats to Validity

This section discusses both threats to validity that were addressed and those that were not. First, in order to avoid a learning effect during the classroom practice session, the order of the items being classified was randomized. Also, to reduce the external validity threat that would be caused by using a toy example, the students were working on real systems and interacted with real clients to develop the requirements specification document.

However, there were also some threats to validity that were not addressed by the experimental design. While the students did work on a real system, the study was still done in a classroom environment with students rather than professionals, which affects the external validity of the study. Another important threat is the

lack of a control group. During the re-inspection where the students used the EAP, there was not a group of subjects doing a re-inspection using their standard technique to compare against. Therefore, it is possible that a portion of the observed effect may have been caused by the fact that the students were inspecting the artifact for a second time.

### 6. DISCUSSION OF RESULTS

In this section, each original hypothesis from Section 3.1.1 is revisited to examine the level of support provided by the results presented in Section 4.

#### 6.1 Hypothesis 1

*The EAP improves the effectiveness and efficiency for teams and for individuals.*

In terms of effectiveness, results from the analysis in Section 4.1 showed that the EAP improves the effectiveness of each teams by a large margin (75% and 156%). Furthermore, there was a healthy increase in the effectiveness for each of the 16 subjects. In addition, there was a large increase in efficiency when only the inspection effort is considered. There was a small decrease in efficiency when the full EAP effort is considered, however this decrease was not statistically different

Combining the effectiveness and efficiency results allows the conclusion can be drawn that the EAP was both effective and efficient and should be studied further to for continue improvement.

#### 6.2 Hypothesis 2

*The RET is useful for improving software quality.*

The results from Section 4.2 showed that in general the students found the RET useful. In addition, the classes in the RET covered real requirement errors and it was easy to use. Finally, the RET was understandable and well modularized. There are a few areas that were identified for improvement before using the RET in future studies.

#### 6.3 Hypothesis 3

*The RET provides significant insights into the requirement phase of software development process.*

The results from Section 4.3 showed that while people errors made the largest contribution to error and fault injection density, statistically the errors and faults were evenly distributed across the classes. But, people errors did contribute significantly more redundant faults than the other two classes. Similarly process errors contributed significantly more time consuming faults than the other two classes. Finally, people errors contribute significantly more errors causing multiple faults. These results provide confidence that the error types in the RET are valid and provide a good coverage of the overall requirements error space.

#### 6.4 Hypothesis 4

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

The results from Section 4.4 indicated that between 20% and 25% of the errors found could be classified as human cognition errors. This result supports the hypothesis and indicates that using research from other related fields like human cognition is beneficial for the RET.

## 6.5 Hypothesis 5

*Individual performance during the EAP depends on various independent variables including: process conformance, performance on practice run, usefulness of training procedure, and effort applied.*

The results from Section 4.5 showed that process conformance during the error abstraction process and the median of the process conformance values as well as the performance on the practice run all had a significant impact on the fault detection density. In addition, the defect detection density was significantly correlated with the perceived usefulness of the EAP training but not with the RET training. Surprisingly, effort spent during the EAP had a significant effect on the fault detection rate. These results support the hypothesis and allow the following conclusions to be drawn: 1) to increase fault detection density subjects must follow the process during error abstraction; 2) an increase in effort spent is likely to lead to an increase in fault detection rate; 3) a subject's performance on a practice run can be used to predict their fault detection density using the EAP and 4) useful training on how to abstract errors from faults is necessary for improving fault detection effectiveness.

## 7. CONCLUSION

Based on the results of the study, both the EAP and RET provide a benefit to developers who use them. The feedback provided by the students will be used to refine and improve both processes to make them better for future studies.

This was an initial investigation and more experiments need to be conducted to empirically validate the EAP and continue to refine both the EAP and RET with feedback from subjects. These future studies will include experiments that involve a control group and will be conducted in different settings (e.g. industrial).

Other future plans include creating a Design Error Classification Taxonomy using the same approach and building fault detection techniques based on the error taxonomies. To get an initial idea of the value of these two ideas, the students were asked to provide feedback after the study was over. They were asked whether they thought that it would be worthwhile to develop error taxonomy for later lifecycle phases (ECTSP) and to derive tools and methods for improving software quality based on the RET (DTRE). The responses of the students are shown in Figure 23. Based on the positive results of the study and these responses, it will be a worthwhile endeavor to pursue these ideas in the future.

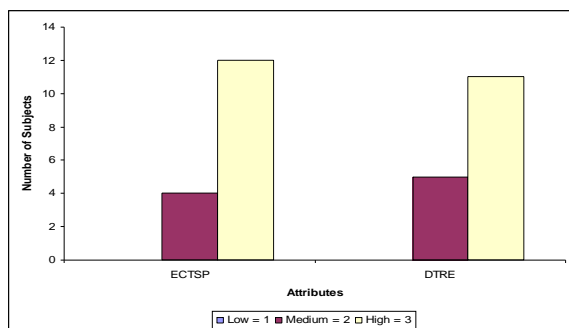


Figure 23: Frequency distribution for motivation

## 8. ACKNOWLEDGEMENTS

We would like to thank the students in Fall 2005 CSE 3213 course at Mississippi State University for participating in this experiment and working hard to provide the data. We also thank Dr. Thomas

Philip for allowing us to conduct the study in his course. Finally we acknowledge the Empirical Software Engineering group at Mississippi State University for providing useful feedback on the research and this paper.

## 9. REFERENCES

- [1] Boehm and V. R. Basili, "Software Defect Reduction Top 10 List," *IEEE Computer*, vol. 34, no. 1, pp. 135-137, January 2001.
- [2] J. Carver, "Impact of Background and Experience on Software Inspections," PhD. Thesis, University of Maryland, College Park. 2003.
- [3] J. Chaar, M. Halliday, I. Bhandari, and R. Chillarege, "In-Process Evaluation for Software Inspection and Test," *IEEE Transactions on Software Engineering*, 19(11):1055-1070, November 1993.
- [4] R. Chillarege, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray, and M. Y. Wong, "Orthogonal Defect Classification - A Concept for In-Process Measurements," *IEEE Transactions on Software Engineering*, 18(11):943-956, November 1992.
- [5] W. J. Christopher, "The Cost of Errors in Software Development: Evidence from Industry," *The Journal of System and Software*, pp. 1-9, 2002.
- [6] G. Damele, G. Bazzana, F. Andreis, S. Aquilio, "Process Improvement through Root Cause Analysis," *In Proceedings of Third International Conference on Achieving Quality in Software*, pp. 35-47, 1996.
- [7] W. Florac, "Software Quality Measurement: A Framework for Counting Problems and Defect," Software Engineering Institute, Technical Report CMU/SEI-92-TR-22, 1992.
- [8] F. Lanubile, F. Shull, V.R. Basili, "Experimenting with Error Abstraction in Requirements Documents," *In Proceedings of 5th International symposium on software metrics*.
- [9] C. P. Lawrence, I. Kosuke, "Design Error Classification and Knowledge Management," *Journal of Knowledge Management Practice*, May 2004.
- [10] M. Lezak, E. D. Perry, D. Stoll, "A Case Study in Root Cause Defect Analysis," ICSE 2000, Limerick, Ireland.
- [11] S. Sakhivel, "A Survey of Requirements Verification Techniques," *Journal of Information Technology*, Vol.6, pp. 68-79, 1991.
- [12] F. Shull, I. Rus, and V. Basili, "How Perspective-Based Reading Can Improve Requirements Inspections," *IEEE Computer*, Vol. 33, No. 7, July 2000.
- [13] "Software Engineering Laboratory: Software Measurement Guidebook," NASA/GSFC Software Engineering Laboratory Technical Report SEL-94-002, 1994.
- [14] C. Trevor, S. Jim, C. Judith, K. Brain, "Human Error in the Software Generation Process," [www.branchlines.org.uk/Research/Tread1.pdf](http://www.branchlines.org.uk/Research/Tread1.pdf)
- [15] G. Walia, "Empirical Validation of Requirement Error Abstraction and Classification: A Multidisciplinary Approach," M.S Thesis, Mississippi State University, MS, 2006.