2016 IEEE 27th International Symposium on Software Reliability Engineering

Using A Cognitive Psychology Perspective on Errors to Improve Requirements Quality: An Empirical Investigation

Vaibhav Anu, Gursimran Walia
Department of Computer Science
North Dakota State University
Fargo, USA
vaibhav.anu@ndsu.edu
gursimran.walia@ndsu.edu

Wenhua Hu, Jeffrey C. Carver
Department of Computer Science
University of Alabama
Tuscaloosa, USA
whu10@crimson.ua.edu,
carver@cs.ua.edu

Gary Bradshaw
Department of Psychology
Mississippi State University
Mississippi State, USA
glb2@psychology.msstate.edu

***Abstract*—Software inspections are an effective method for early detection of faults present in software development artifacts (e.g., requirements and design documents). However, many faults are left undetected due to the lack of focus on the underlying sources of faults (i.e., what caused the injection of the fault?). To address this problem, research work done by Psychologists on analyzing the failures of human cognition (i.e., human errors) is being used in this research to help inspectors detect errors and corresponding faults (manifestations of errors) in requirements documents. We hypothesize that the fault detection performance will demonstrate significant gains when using a formal taxonomy of human errors (the underlying source of faults). This paper describes a newly developed Human Error Taxonomy (HET) and a formal Error-Abstraction and Inspection (EAI) process to improve fault detection performance of inspectors during the requirements inspection. A controlled empirical study evaluated the usefulness of HET and EAI compared to fault based inspection. The results verify our hypothesis and provide useful insights into commonly occurring human errors that contributed to requirement faults along with areas to further refine both the HET and the EAI process.**

Keywords—human error; requirements inspection; taxonomy

I. Introduction

Contemporary requirements inspection techniques utilize information gleaned from historical fault data to guide inspectors when they are seeking to identify faults recorded in requirements documents [1]. Historical data is converted into checklist items and these items help the inspectors in identifying faults. This is especially true for the fault checklist (FC) technique, a widely used requirements inspection technique. Researchers have also developed and evaluated fault classification taxonomies to help support the fault based inspection process with varying degrees of success [2, 3].

While fault based techniques like the fault checklist (FC) technique have proven beneficial, they cannot help inspectors identify all (or even a large percentage) the faults because they focus on the manifestation of the problem (faults) and not the source of the problem (underlying error). We propose that a better approach of finding faults is to analyze and understand the underlying causes of faults (i.e., errors).

The terms *fault* and *error* have competing definitions in the software engineering literature ranging from *human error* (mental event) [4, 5] to *service error* (related to coding or programmatic failures or operator error) [6]). The term *error* in our work refers to *human error*, which is a purely mental event (a failure of human cognition that results in a fault/defect) rather than the s*ervice error*.

To understand the source of the faults in requirements specifications, Lanubile et al introduced the notion of *error abstraction*, which was intended as an addendum to standard Fault Checklist (FC) inspections [7]. They provided evidence for the usefulness of abstracting a fault's underlying cause and then utilizing this abstracted information to locate other related faults (that are overlooked during the original inspection). Building on Lanubile's work, Walia & Carver reviewed published literature to develop an initial classification of requirement errors (as shown in Fig. 1), to make developers more effective during the error abstraction process by providing inspectors with structured error information [8]. Based on a comprehensive dataset of empirical studies, Walia & Carver also provided strong empirical evidence that the use of structured error information helped inspectors find a significantly large percentage of faults that were originally not found during the fault-checklist inspection [9–12]. The results also highlighted that a significant proportion of requirements faults were traced back to the *People Errors category* of RET [10]. The *People Errors* category relates to the failures of human cognition (i.e., *human errors*). Other researchers who used the requirement error taxonomy (RET) to analyze errors in space software requirements also reported people errors (*human errors*) as most frequent type of errors [13].

These findings that *human errors* are the underlying cause of requirements faults are not surprising as requirements engineering is a human centric activity. Although the major *people errors* described in RET (e.g., *process execution* errors, *specific application* or *domain knowledge* errors, *process* errors) represented real problems, RET lacked an evaluation of underlying human cognitive mechanisms (or human information processing models) that caused these problems and consequently, faults. That is, there was not a clear mapping

This study was sponsored by National Science Foundation Awards 1423279 and 1421006

between a fault (manifestation of a problem) and its underlying human error (source of the problem).

To that end, the current research extends RET by: (1) refining the *People Errors* category of RET, and (2) employing distinct error types identified by cognitive *human error researchers* (also employed by researchers in fields such as aviation, oil industry, medicine) [14–16] to produce distinct subcategories of RET's *People Errors*. This produced a taxonomy of human errors committed during the development of software requirements specification (SRS) document. A detailed discussion on the human error taxonomy (HET) appears in Section II.A.

Next, we evaluated the usefulness of HET in helping inspectors find human errors and resulting faults during the requirements inspection. To perform this evaluation, the error abstraction process (originally coined by Lanubile et al., [7] and adapted in RET studies [9, 11, 12]) was further enhanced by including *Human Error Abstraction Assist* (referred to as **HEAA** here onwards). The HEAA assisted inspectors in identifying known error patterns (contained in HET) and interpreting them in light of requirements engineering activities. A description of *HEAA* appears in Section II.C.

The remainder of the current paper reports the results from an empirical study that compared the usefulness of using structured human error information to find faults that are left undetected after the standard fault checklist (FC) inspection. The paper also provides insights into the incidence of *human errors* during the requirements development process. Section II describes HET, HEAA and EAI. Section III describes empirical study design followed by a description of major results in Section IV. Section V describes the validity threats, Section VI and VII discuss the results and future work.

II. Error Absraction and Inspection (EAI)

Proposed Approach – EAI: The proposed approach (EAI) adds an extra step to the fault checklist (FC) inspection process. The extra step consists of assisting inspectors in identifying underlying human errors (abstracted from the faults found during the FC inspection). Inspectors then use the abstracted human error information to re-inspect SRS document for additional faults. Error abstraction step requires inspectors to *retrospectively* analyze each fault (found during the FC inspection) to determine the cognitive process that went awry (or was flawed to begin with), thus causing the fault to be injected. Requirements development involves multiple activities (elicitation, analysis, documentation etc.) and multiple people (client, end-users, requirement analysts, requirement author). So, to assist inspectors during the error abstraction, human errors collected from software engineering and psychology domains were structured into an *error taxonomy* (HET- Section II.A), which in turn was reduced to an Error Abstraction Assist (HEAA – Section II.C).

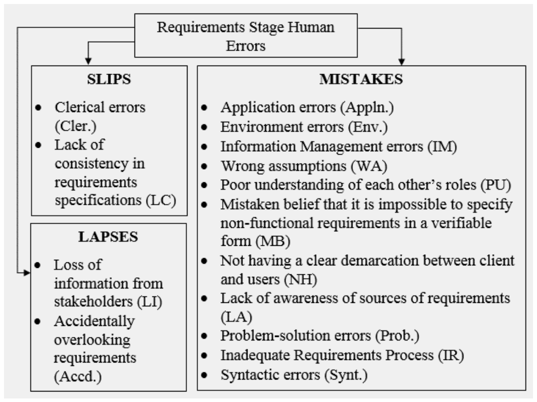
A. Human error Taxonomy

The Human Error Taxonomy (HET) was heavily influenced by human information processing models (human error models) described by Cognitive Psychologists, most notably by James Reason and Rasmussen [17–19]. The HET was developed in consultation with Cognitive Psychologists (via human error workshop – WAHESE 2015; http://humanerrorinse.org); one of the psychology experts is also a co-author on this paper. Additionally, we conducted an extensive systematic literature review (SLR) of both the Software Engineering (SE) literature and the Cognitive Psychology literature [20]. The detailed procedure followed during the creation of HET can be found in [20]. A brief description of the SLR process (as more details are outside the scope of this paper) and resulting HET is provided in the following paragraphs.

SE and psychology literature was surveyed with the focus of identifying the various requirements phase human errors reported in published sources. After identifying the human errors, similar human errors were merged together to create human error classes. Fig. 1 shows the 15 human error classes that emerged as a result of this endeavor. Each of the 15 human error classes represent mental human errors that can occur when performing different tasks during the requirements development process. Next, Cognitive Psychology literature was surveyed for identifying the human error classification system appropriate for classifying requirements phase human errors. A human error theory called the *Swiss Cheese Model*, proposed by James Reason provides a human error classification system that was deemed most appropriate for the SE domain [17, 18]. Reason's theory has been extensively adapted in several domains for the purpose of mapping accident causation to human cognition failures. For example, the *Human Factors Analysis Classification System* (HFACS), based on Reason's human error theory has been used in aviation accident analysis, health care incident analysis, and analysis of military mishaps [15, 16, 21].

Based on Reason's classification, human errors were classified into cognitive failures during *planning* and cognitive failures during *execution* (i.e., execution of a plan). Reason further broke down execution failures into *slips* and *lapses,* whereas planning failures were mapped to *mistakes*. To aid reader's understanding, everyday examples of *Slips, Lapses* and *Mistakes* are provided along with software specific examples in [20, 22].

Slips are understood as execution failures and occur when a planned action is poorly executed due to inattention. "*Fat-fingering*" or typing an incorrect letter is a very common


Requirements Stage Human Errors
SLIPS
Clerical errors (Cler.)
Lack of consistency in requirements specifications (LC)
LAPSES
Loss of information from stakeholders (LI)
Accidentally overlooking requirements (Accd.)
MISTAKES
Application errors (Appln.)
Environment errors (Env.)
Information Management errors (IM)
Wrong assumptions (WA)
Poor understanding of each other's roles (PU)
Mistaken belief that it is impossible to specify non-functional requirements in a verifiable form (MB)
Not having a clear demarcation between client and users (NH)
Lack of awareness of sources of requirements (LA)
Problem-solution errors (Prob.)
Inadequate Requirements Process (IR)
Syntactic errors (Synt.)
Fig. 1. Human Error Taxonomy (HET)

example of slip. Typing is a routine action that is executed (as a part of tasks like writing a paper or an email) by individuals on a daily basis. Typing an incorrect letter due to not paying proper attention is a common *slip*. *Lapses* are also execution failures and occur when an action is forgotten while executing a planned task. Lapses are understood as memory related execution failures. A common example of a lapse is forgetting to save a file (e.g., MS Word or Excel file) when closing the file or when logging off from a computer. *Mistakes* occur due to inadequate planning which is generally the result of an unfamiliar situation. A common medical example is misdiagnosis due to not being familiar with a certain patient's condition, or in most cases due to not studying the symptoms of the patient properly. So, mistakes happen due to making a plan without having proper knowledge about the situation or problem-space.

The 15 requirements phase human error classes found from the software engineering literature were classified into *Slips, Lapses*, or *Mistakes* to create the Human Error Taxonomy (HET) shown in Fig. 1. A complete description of the error classes in HET, and examples of human errors and corresponding faults can be found in [20, 22].

B. Contemporary Human Factors Analysis Frameworks and HET: A Discussion

The primary goal of our research (developing HET) is to improve requirements quality via early fault removal. HET works by helping inspectors investigate the underlying reasons behind requirements faults, which is analogous to accident investigation frameworks like HFACS [15] and SERA [23]. These human error analysis frameworks provide the ability to both: (1) investigate the human factors causes of accidents/incidents, and (2) assess risks associated with human failures. SERA (Systematic Error and Risk Analysis), provides 12 failure-modes that occur during manual or process control [24, 25]. These failure–modes (referred to as active failures) represent breakdowns in human information processing that arise during routine skilled behavior [25]. SERA begins with accident reports (equivalent to a system failure). The goal of our HET is to find and identify faults before failures occur, where they are easier and cheaper to fix. HET provides concrete instances of human cognitive failures that occur in requirements development process. These concrete instances (e.g., *overlooking requirements, lack of awareness of sources of requirements etc.*) have been collected from published resources. Here we demonstrate that awareness of these error types allows software engineers to find faults early in the system, even before the coding process begins.

C. Human Error Abstraction Assist (HEAA)

To help inspectors focus on the human errors across different requirements activities (elicitation, analysis, specification, verification, management), the authors created a Human Error Abstraction Assist (HEAA) in consultation with the cognitive psychology expert. HEAA guides inspectors during the error discovery and is shown in Appendix A.

HEAA was developed after a pilot testing with a different set of subjects (different from this study) that provided us the feedback that instead of starting with the categorizing of the human errors (into *Slips, Lapses* or *Mistakes*); it is more intuitive to focus on the requirement phase scenarios (Question #1 in Appendix A). Creation of HEAA required the researchers to distribute the 15 human error classes of HET across various requirements engineering (RE) activities. This required the researchers to use their knowledge of RE to make judgements about - *if and how* a human error can occur during multiple RE activities. For example, Clerical Errors (which occur due to carelessness while performing mechanical transcriptions from one format or from one medium to another), can happen while recording user needs (elicitation) or while formally documenting the recorded user needs (specification). The researchers note that this is just the preliminary distribution of human errors across RE activities and the distribution will be refined with experience and data garnered from empirical studies and expert opinion.

HEAA provided the inspectors an intuitive framework for abstracting human error/s from a given fault during the current study. As Question #1 (in Appendix A), a checklist of items was provided to guide the selection of RE activity. The next question required the inspector to visualize and provide an account of the scenario where the human error occurred. This helped the inspectors in improving their understanding of the requirement phase activity where the human error might have occurred (so in a sense steps 1 and 2 were iterative in nature). Next, the inspector picked a human error from the options provided to him/her under error boxes labelled with requirement activity names. Finally, the description of how the selected error caused the fault (from which it was being abstracted) helped researchers evaluate the consistency and correctness in an individuals' understanding of human errors and their impact in terms of faults caused by them. While the primary goal of this study was not to evaluate HEAA, authors eventually plan to reduce it to an online tool that can be used by inspectors.

III. Experiment Design

The goal of this study was to evaluate whether the use of structured error information (i.e., HET) to support the fault checklist inspection can help improve requirements quality when compared to using only the fault checklist (FC) inspection. To achieve this goal, a controlled experiment measured effectiveness of subjects during the FC process vs. the EAI process (supported by HET) during inspection of an SRS. The experiment was designed as quasi-experimental repeated measures investigation. The experiment consisted of

TABLE I. Hypotheses

#	Description
H1	An error-abstraction inspection will help inspectors' discover a significantly larger number of faults that are otherwise left undetected when using the standard fault based inspection
H2	HET is useful for helping inspectors understand errors and detect faults present in the SRS documents
H3	HET provides significant insights about the type of human errors that cause most faults in the SRS document
H4	Individual performance during the error-abstraction inspection depends on other independent variables (e.g., initial training, process conformance, effort, training usefulness)

TABLE II. STUDY VARIABLES

Independent Variables	Description
Process Conformance	measures how closely the subjects followed the experimental procedure
Training Usefulness	measures usefulness of training on inspection techniques
Effort Spent	measures the time spent during the inspection
Training Performance	measures performance of subjects during a practice inspection exercise using EAI
Dependent Variables	**Description**
Effectiveness	Number of faults found by each subject
Efficiency	Number of faults found per hour

two sessions: an initial *training session* and a final *transfer session*. Transfer session refers to the leg of the experiment, wherein subjects use (or transfer) the knowledge gained during training to carry out the actual experiment tasks. During the course, subjects had been trained on using FC to inspect multiple requirements documents. During the initial *training session*, subjects abstracted and classified errors from a requirements document, learned about the EAI process, and applied their knowledge to re-inspect the same document. The *transfer session* evaluated subjects' performance of FC and EAI on a new SRS document.

A. *Methodology*

This section describes the hypotheses, variables, subjects and artifacts, experiment steps and data collected during the study. The complete experimental package is available at http://humanerrorinse.org/Studies/2015/Fall_NDSU/

1) Hypotheses and variables:

The study hypotheses are listed in Table I. Hypothesis 4 investigated effect of independent variables on the dependent variables defined in Table II.

2) Subjects:

The participating subjects in this study were 17 graduate students, enrolled in the *Software Requirements Definition and Analysis* course at North Dakota State University in the Fall-2015 semester. The students were a mix of MS and PhD in computer science or software engineering and had prior IT industry experience. The course trained students on identifying, analyzing, documenting and verifying requirements for a software system.

3) Artifacts:

Two different SRS artifacts (PGCS and RIM) were used during the study. The ***Parking Garage Control System (PGCS)*** artifact was used during initial training as subjects performed a practice inspection using EAI on a SRS document. PGCS described requirements for controlling the entries and exits of a parking garage. The PGCS SRS [11, 22, 26] was developed at University of Maryland, was 10 pages long, and seeded by the original developers with 30 realistic faults. PGCS was chosen during the training due to its generic domain and seeded set of faults.

The ***Restaurant Interactive Menu system (RIM)*** artifact was used during the transfer session. The RIM system is responsible for taking customer's orders in a restaurant with the help of an interactive PDA or online system. This system gives customers the ability to make dining choices at their own pace and request assistance at their convenience, and gives the restaurant owner better manageability over the menu, staff and revenue/cost analysis. The RIM SRS [11, 26] document was developed for a real project through interaction with clients, was 21 pages long, and contained real faults. Both PGCS and RIM SRS have been used in prior inspection studies [11, 26], and have been improved (after discovering some inconsistencies in the document) to standardize formatting (e.g., font & font size, line & paragraph spacing, sections), and to remove minor issues (e.g., blank description).

B. *Experimental Procedure*

Fig. 2 shows the experiment procedure including training,

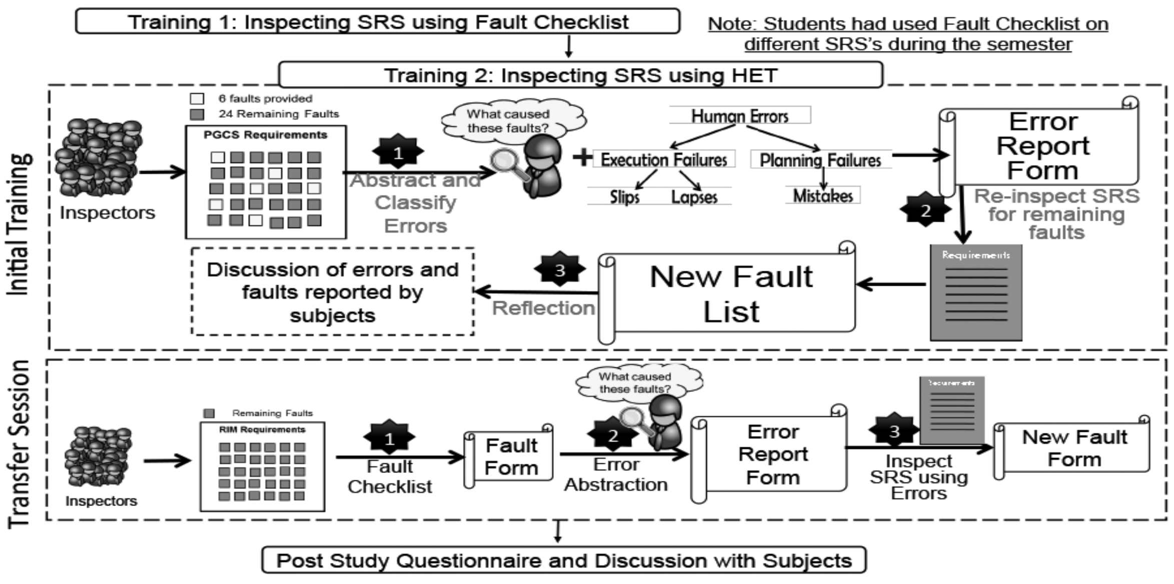

Training 1: Inspecting SRS using Fault Checklist
Note: Students had used Fault Checklist on different SRS's during the semester
Training 2: Inspecting SRS using HET
Initial Training
6 faults provided
24 Remaining Faults
PGCS Requirements
Inspectors
1
Abstract and Classify Errors
What caused these faults?
Human Errors
Execution Failures
Planning Failures
Slips
Lapses
Mistakes
Error Report Form
2
Re-inspect SRS for remaining faults
Requirements
3
Discussion of errors and faults reported by subjects
Reflection
New Fault List
Transfer Session
Inspectors
Remaining Faults
RIM Requirements
1
Fault Checklist
Fault Form
What caused these faults?
2
Error Abstraction
Error Report Form
3
Inspect SRS using Errors
New Fault Form
Post Study Questionnaire and Discussion with Subjects

Fig. 2. Experimental Procedure

experimental tasks, and output for each task. The study was conducted in two phases (initial training and transfer session), included two training sessions (one for FC and one for EAI).

1) Training 1 – Pre-experimental training on FC:

Over the course of the semester, subjects had been trained on fault checklist (FC) based inspections and had applied FC to detect faults on various SRS documents. Therefore, a quick recap of FC training session was done because subjects were experienced at applying the FC inspection technique.

2) Training 2 –Initial training on EAI:

During this 90-minute training session, subjects were trained on the human error taxonomy (HET), error abstraction and classification using HET, using the Human Error Abstraction Assist (HEAA) to record and classify human errors, and using abstracted human errors to locate additional faults in the SRS document. After their introduction to the HET, subjects were provided with the PGCS SRS document along with a sample of 6 (out of 30 seeded) randomly chosen faults. Subjects were asked to perform: (1) Error Abstraction and Classification on the 6 faults; and (2) Re-inspect SRS for remaining faults.

- *Step 1 – Abstraction and classification of human errors in PGCS SRS*: The subjects used information from initial training to abstract and classify human errors from the faults using Human Error Abstraction Assist (HEAA). The output of this step was 17 individual Error Report Forms (one per subject) containing human errors present in PGCS SRS.
- *Step 2 – Re-inspecting PGCS SRS to detect faults*: The subjects re-inspected the PGCS SRS using the human error information contained in error report form (from Step 1). The output of this step was 17 individual New-Fault lists (one per subject) containing new faults found during the re-inspection.

Following the completion of Step 2, researchers discussed the issues faced by individual subjects when performing error abstraction, and re-inspection using the EAI process. Most of the discussion was focused around improving their understanding of EAI and reporting of errors the Error Report form and new faults in the New-Fault form.

3) Transfer Session:

During the transfer session, subjects were provided the RIM SRS document and were asked to perform: 1) Fault Checklist inspection; 2) Error Abstraction and Classification; and 3) Re-inspection of the SRS for remaining faults using error information.

- *Step 1 – FC Based Inspection*: The Subjects used the FC inspection technique to inspect the RIM SRS. The result of this step was 17 individual Fault Forms (one per subject) containing faults present in RIM SRS.
- *Step 2 – Human Error Abstraction and classification*: Subjects used HEAA to abstract and classify human errors for each fault they found during Step 1. The result of this step was 17 individual Error Report Forms (one per subject) containing human errors committed during the development of RIM SRS.
- *Step 3 – Re-inspecting RIM SRS for remaining faults*: The subjects re-inspected the RIM SRS using the human error information from error-report form (Step 2). The output of this step was 17 individual New-Fault Lists (one per subject) containing new faults present in RIM SRS.

4) Post-study Questionnaire:

After completing of the above steps, subjects provided feedback regarding the usefulness of the EAI, HET (on attributes listed in Table III) and training procedures to better understand the results and make further improvements.

C. Data collection

Qualitative and quantitative data was collected from individual subjects during both phases of the study (i.e., training and transfer sessions). The *quantitative* data collected during training included the number of abstracted errors (from faults provided to them) and additional faults found by each subject during the re-inspection of PGCS SRS. Subjects also reported the fault type for any additional faults they identified. The quantitative data collected during transfer session included: (1) fault data – fault description, type of fault, during the first inspection; (2) error data – error description and classification, which requirements phase the error occurred?, who committed the error?, during the error abstraction step; and (3) new faults found by subjects during the re-inspection. The error and fault reporting forms also included timing data, i.e., start and end time for each task, the exact time when they found a human error or a fault, and breaks they took.

The faults found by each subject (reported in *fault forms*) were validated for true-positive by comparing it against list of true-positive faults known to be present in PGCS (seeded with 30 faults) and the RIM document (via data collected from a series of past studies). We removed the false-positives prior to analysis. Similarly, the error data (reported in *error-report forms*), was evaluated based on three parameters: (1) subject's error description represented a flaw in human thought process, (2) subject's error description displayed a sound understanding of the activities involved in requirements engineering, and (3) subject selected an error class from HET which consistent with their error description.

Other *quantitative* data that was collected included feedback provided by subjects on a 5-point scale (ranging from "1- very low" to "5-very high") on various aspects of HET, HEAA, and training usefulness. This was done via a survey that also collected feedback from subjects regarding the "*worthiness of effort*", and "*process conformance*" when using the EAI process to re-inspect RIM SRS. The attribute, *worthiness of effort*, provided insights about whether the subjects thought that the effort they spent in using HET to re-inspect the SRS was valuable in locating additional faults.

IV. Analysis and Results

This section provides the analysis of the *quantitative* data collected during the inspection of RIM document (transfer session in Fig. 2) and *qualitative* data collected post inspection. The results are organized around the four study hypotheses (Table I). An alpha value of 0.05 was used.

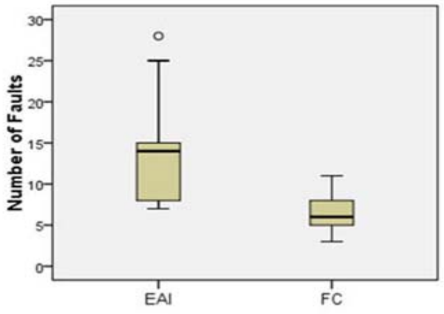


Fig. 3. Effectiveness of FC and EAI – Box Plots

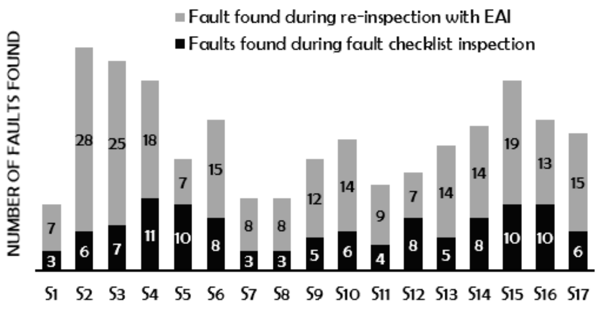


Fig. 4. Effectiveness of EAI vs. FC inspection

A. *Fault Detection Effectiveness and Efficiency*

Fault Detection Effectiveness: We measured the effect of the EAI (and HET) on fault detection effectiveness of individual inspectors by comparing the number of faults detected during the FC inspection of RIM SRS against the number of faults detected during the EAI-based re-inspection of RIM SRS. To provide an overview of the results, Fig. 3 compares the effectiveness (# of faults found) during the EAI vs. FC. The results showed that, subjects were able to detect a large number of faults when performing error-based inspection (guided by HET) as compared to the fault-based inspection (using FC). Fig. 3 also showed that, error inspection performance is skewed (to the right) and contained one outlier (subject#2). To better understand the individual performance, Fig. 4 compares the fault count of each subject during FC inspection (bottom portion of each column) vs. the new fault count during re-inspection using EAI (top portion of the same column).

For example, S1 (inspector# 1) found 3 faults during the first inspection (using FC), and found 7 new faults during the re-inspection (using EAI), which computes to a percentage increase of 233% in fault detection effectiveness. Overall, subjects found an average of 6 faults during the first inspection (FC) and an average of 14 new faults during the second inspection (EAI), with an average increase in effectiveness of 225%. A one-sample t-test was used to evaluate whether the average number of faults found using EAI were significantly greater than the average number of faults found using FC. Subjects found an average of six (6) faults during FC inspection, so a count of 6 faults was used as a comparison number (as opposed to evaluating contribution significantly greater than 0). The result of the one-sample test ($p<0.001$) showed that the average number of faults found using EAI was significantly higher than six (6).

Results shown in Fig. 4 and results from the one-sample test verify our hypothesis that an error-abstraction inspection (EAI), supported by HET, can help inspectors discover a significantly more number of faults in an SRS that are otherwise left undetected during the fault based inspection. This is an important result, even when considering that some of the new faults could be attributable to the fact that the document was being reviewed the second time (further discussion on this aspect is provided in Section VI).

Fault Detection Efficiency: We also evaluated the effect of EAI on fault detection efficiency (*faults/time*). To calculate efficiency during the first inspection, time spent when using FC to record faults was used. To calculate efficiency during the re-inspection, time spent during error-based inspection was used. The results showed that, although subjects found faults faster during re-inspection (an average of 5.5 fault/hour) as compared to the fault rate during the first inspection (an average of 4.2 faults/hour); the improvement was not statistically significant ($p=0.117$ based on a paired t-test).

The lack of a significant effect notwithstanding, the higher sample rate of fault detection using the HET than using the FC implies that software developers do not have to sacrifice efficiency in order to employ the HET for fault-finding. This was expected because of the fact that HEAA checklist was being used the first time in our research and required students to write a detailed scenario of which human error (in HET) occurred during the SRS development (for us to be able to evaluate their understanding). We expect that, in real settings, there will be a single re-inspection step. Therefore, based on the *effectiveness* and *efficiency* results, subjects found a significantly larger number of faults when using human errors to guide the re-inspection but not at a significantly faster rate.

B. *Usefulness of HET (H2)*

As mentioned earlier, we measured subjects' opinion of HET on seven essential attributes: *simplicity, ease of understanding, ease of using, intuitiveness, orthogonality (error classes don't overlap), comprehensiveness,* and *usefulness*, on a scale of one to five (1-very poor; 2-poor; 3-neither good nor poor; 4-good; 5-very good). The average ratings of each characteristic across all the subjects is shown in Table III. While the average rating for each characteristic was

TABLE III. AVERAGE RATING OF HET ATTRIBUTES

HET Attribute	Mean Rating	Median	p-value
Simplicity	3.8	4.0	< 0.001
Ease of Understanding	4.0	4.0	< 0.001
Ease of Using	3.87	4.0	< 0.001
Intuitiveness	3.73	4.0	< 0.001
Orthogonality	3.73	4.0	< 0.001
Comprehensiveness	4.2	4.0	< 0.001
Usefulness	4.2	4.0	< 0.001

greater than 3 (mid-point of the 5-point scale); we performed one-sample t-tests to determine if HET was rated significantly better than 3 (mid-point of 5-point scale) on each attribute. Based on the results, HET was positively rated ($p<0.001$) on all attributes. Also, all the attributes received a median rating of 4 (median value shown in Table III).

Additionally, subjects evaluated HET using 5-point scale on the *worthiness* of effort spent using the HET to locate faults in SRS, and the *confidence* that human errors in HET represent real mistakes made during the SRS development. Subjects rated the *worthiness* of HET and their *confidence* in error classes of HET significantly positive (p-value <0.001 for both). Overall, subjects perceived HET positively (consistent with their improved performance during the re-inspection).

C. *Major Insights Provided by HET (H3)*

Error data (error-report forms) and fault data (fault lists) was analyzed to gain insights into the relative contributions of three high-level human error types (*Slips* vs. *Lapses* vs. *Mistakes*) and the 15 detailed human error classes (*Clerical Errors, Application Errors etc.* in Fig. 1) to total errors and total faults found during the RIM inspection (i.e., transfer session). This was done to understand the most frequently occurring human errors, and the usefulness of human error classes.

1) Contribution of human error type and error class vs total errors and total faults:

We analyzed the reports of human errors and their classifications (during error abstraction) to determine the percentage contribution of 3 high-level human error types (*Slips, Lapses*, and *Mistakes*) and 15 error classes to the total number of errors reported by subjects. Similarly, we analyzed human errors in terms of the faults caused by those errors (during fault inspection and re-inspection) to determine the relative contributions of 3 high-level human error types (*Slips, Lapses*, and *Mistakes*) and 15 error classes towards the total fault count. These analyses evaluate: *whether 3 error types provide uniform distribution or whether more faults stem from a single error type? Are all 15 error classes relevant to the requirements engineering process?* Fig. 5 shows the percentage contributions of Slips, Lapses and Mistakes to total errors and total faults found by subjects during the RIM inspection. Also, further analysis of contribution of 15 error classes within these 3 error types towards total faults and errors is shown in Table IV.

Major Insights: Major insights gained from Fig. 5 and Table IV are discussed follows: Regarding the ***error-types*** (Fig. 5); *Slips* were reported as most commonly occurring human

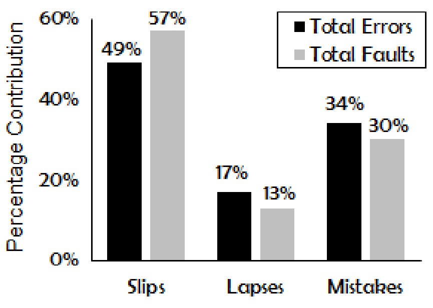


Fig. 5. Percentage Contribution of Human Errors

error (49% of total errors) during the development of RIM SRS. Consequently, *slips* led to the detection of largest number of faults (57% of total faults). The results from Chi-Square test showed that the observed contribution of error types was significantly different from uniform distribution (33.33%) for total errors and total faults ($p<0.001$ for both cases). The high contribution of *Slips* (vs. *Mistakes*) was unexpected considering that HET included only 2 error classes belonging to Slips, but 11 error classes belonging to Mistakes.

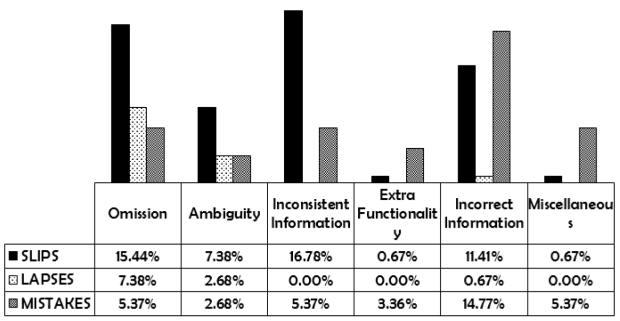
Regarding the ***error-classes*** (Table IV); *Clerical Errors* (a class of *Slip*) caused the largest percentage of total errors (37%) and total faults (43%). Clerical errors occur due to carelessness while performing mechanical transcriptions from one format or from one medium (e.g., requirements engineer notes while eliciting user needs) to another (writing specifications). Next, *Application Errors* (a class of *Mistake*) were second largest contributor to errors (16%) and faults (12%). Application errors arise due to misunderstanding of problem domain (e.g., incorrect belief that the information regarding all the stakeholders have been identified) or a misunderstanding of an aspect of overall functionality.

Regarding the ***validity of error classes***, subjects classified errors into 14 out of the 15 human error classes, which indicates that the human error classes in the HET are relevant to the requirements phase of the software development process. No errors were classified into an error class NH (*not having a clear demarcation between clients and users*). The researchers intend to improve the details of this error class (with better examples) in order to help inspectors get a better understanding.

Compared to Psychology findings: The results (i.e., significantly higher contributions of Slips) are consistent with the human error frequencies reported in the cognitive science literature. In his seminal book on Human Error, James Reason provides reports of errors where in general, *Slips* and *Lapses* together contributed to around 61% of the human errors [14,

TABLE IV. PERCENTAGE CONTRIBUTION OF ERROR CLASSES

	Slips		Lapses		----------------------------Mistakes---------------------------										
	Clerical	LC	LI	Accd.	Appl.	Env.	IM	WA	PU	MB	NH	LA	Prob.	IR	Synt.
Total Errors	37%	12%	7%	10%	16%	2%	2%	1%	3%	1%		1%	3%	1%	5%
Total Faults	43%	14%	8%	5%	12%	1%		5%	2%			1%	1%	2%	5%



	Omission	Ambiguity	Inconsistent Information	Extra Functionality	Incorrect Information	Miscellaneous
■ SLIPS	15.44%	7.38%	16.78%	0.67%	11.41%	0.67%
▨ LAPSES	7.38%	2.68%	0.00%	0.00%	0.67%	0.00%
▩ MISTAKES	5.37%	2.68%	5.37%	3.36%	14.77%	5.37%

Fig. 6. Human Errors vs Document Fault Types

17, 18]. The results of the current analysis show that Slips and Lapses together contributed to 65% to the total errors and 70% percent to the total faults. This result is also consistent with results from Cognitive Psychology research that prove slips and lapses can be more easily and readily detected when compared to mistakes [14, 17, 18]. As Table IV and Fig. 5 show, subjects were able to trace a relatively high number of faults to Slips (57%).

2) Human errors vs fault types:

We also wanted to gain insights into the type of faults (e.g., Omitted information, Ambiguous or incorrect fact) that were caused by different type of human errors (Slips, Lapses and Mistakes) described in HET. The distribution of human errors (slips, lapses, mistakes) in RIM SRS by fault types (i.e., omission, ambiguity, inconsistent information, extra functionality, and incorrect information) was analyzed. Major insights gained from Fig. 6:

- *Slips* (carelessness) resulted in a majority of *Omission* fault types, which is consistent with human error theories that slips happen while executing routine tasks. It is common to omit some information about a requirement (for e.g., value of a variable) while eliciting or specifying requirements. This happens due to typing incorrectly or taking incorrect notes due to inattention or carelessness. *Inconsistent Information* fault types were also attributed to slips, mostly during specification activity (i.e., errors committed when transforming elicited and analyzed requirements into formalized requirements documents). As Table V shows, *slips* contributed to 85% of total specification activity faults.
- Our analysis also showed that *Mistakes* and *Slips* contributed significantly to the *Incorrect Information* fault type. The contribution from mistakes can be attributed to the fact that elicited user needs are broken down and converted into requirement information during the analysis phase, which is more prone to *Application Errors*, a sub-class of *Mistakes*.

Overall, different human errors led to the detection of different type of faults and *Slips* and *Mistakes* found some faults for each fault type (however same was not true for *Lapses*).

3) Human errors vs specific requirements engineering activities:

Error-fault data specific to various requirement phase activities was examined. The goal of this analysis was to focus software developers' attention on human errors that lead to most faults when specific requirement phase activities are being performed. Table V shows the resulting distribution of errors across requirements' elicitation, analysis, specification, and management activities. Based on Table V, following observations were made:

- *Elicitation*: Most faults in requirements' elicitation were traced back to *Clerical Errors* (a Slip), and *Information Loss* (a Lapse). Furthermore, subjects associated clerical errors and information loss errors to requirement gathering person's carelessness in not asking follow-up questions or just forgetting to note down stakeholder needs.
- *Analysis*: Requirement analysis activity was found most prone to *Application errors* (Mistakes). Subjects reported that the lack of domain knowledge or lack of understanding of certain system functionalities on the part of the requirement analyst resulted in incorrect or incomplete analysis of the stakeholders needs (i.e. requirements), which in turn resulted in injection of faults.
- *Specification*: Most (85%) of the human errors that occurred during specification were due to the two sub-classes of *Slips* (Clerical errors and Lack of consistency in requirement specification). Subjects reported that carelessness on requirement author's part when transforming analyzed requirements into formal specifications resulted in fault-injection during the specification activity.
- *Management*: Very few human errors (five) were found to be associated with the management activity. The reason behind this might be that the HET focusses primarily on human errors and the organizational factors are currently not of significant consideration when human error abstraction is performed. Errors in this activity were due to the *mistakes* involved in selection of an inadequate

TABLE V. CONTRIBUTION OF ERRORS TO TOTAL FAULTS IN DIFFERENT REQUIREMENTS ENGINEERING ACTIVITIES

number (percentage)

	Slips		Lapses		-----------------------------------Mistakes------------------------------------										
	Cler.	LC	LI	Accd.	Appl.	Env.	IM	WA	PU	MB	NH	LA	Prob.	IR	Synt.
Elicitation	37 (15%)	3 (1%)	19 (8%)	10 (4%)	7 (3%)			9 (4%)		1 (0.4%)		1 (0.4%)			
Analysis					23 (9.5%)	1 (0.4%)		3 (1%)	4 (1.7%)			1 (0.4%)	3 (1%)		
Specification	67 (28%)	31 (13%)	1 (0.4%)	2 (0.8%)		2 (0.8%)									12 (5%)
Management							1 (0.4%)							4 (1.7%)	

requirements process or mishandling of certain requirements engineering procedures.

The goal of this analysis was to determine the likelihood of the occurrence of an error in specific requirement activity so that preventive mechanisms (e.g., training, tools and other assists) can be developed. This is very similar to the interventions used in other disciplines for reducing errors (to err is human).

D. *Effect of Independent Variables on Inspection Performance using EAI (H4)*

We also evaluated if independent variables can help us predict or improve the fault detection performance of the subjects when using the EAI process using regression. The subjective self-reported variables included: *Degree of process conformance*, and *Usefulness of training procedures*. Other two variables included objective data: *Effort applied* (time spent) and *Initial training* performance (number of new faults found during the initial training).

Process Conformance vs. Effectiveness: Process Conformance measured using a 5-point scale (1-didn't follow at all to 5-very closely) how closely the subjects followed the steps in HEAA to identify human errors. The results showed that, process conformance during the error abstraction had a strong positive correlation ($r^2 = 0.657$; $p= 0.05$) with the new fault detection effectiveness of the subjects during the RIM inspection. **Training Usefulness vs. Effectiveness:** The results showed a strong positive correlation ($r^2 = 0.766$; $p=0.02$) between the perceived usefulness of EAI training and fault detection effectiveness. **Effort vs. Effectiveness:** This analysis was performed to evaluate whether the amount of time spent during error abstraction (step 2) and re-inspection (step 3) had an impact on the fault detection effectiveness of the subjects. The results showed a significant, but weak correlation ($r^2 = 0.058$; $p=0.476$) between these variables. This is an area of improvement, i.e., help inspectors find faults faster during EAI. **Training vs Transfer Session performance:** The performance of subjects during training (number of faults found during the re-inspection of PGCS document) had a strong positive correlation ($r^2 = 0.974$; $p=0.013$) with the fault detection effectiveness during transfer session (i.e., number of faults found during the re-inspection of RIM document). Therefore, a subjects' practice performance of EAI process can predict his/her performance when using EAI on a real project.

In conclusion, if subjects follow the error-abstraction process closely, find training useful, and perform well during the practice run, they are likely to do well during the EAI inspection on a real project.

V. Threats to Validity

We were able to address the external threat to validity to a certain extent because the subjects, though students, were pursuing MS or PhD and had some experience in the IT industry. However, a major threat unaddressed was a lack of control group. We intend to address this threat by designing and executing control group experiments in future where HET can be compared against Requirement Error Taxonomy (RET) proposed by Walia & Carver [8].

Another validity threat comes from re-inspection. The subjects are re-inspecting the same document using EAI (with a time gap between inspections). However, previous results have shown that, re-inspecting the SRS using proven fault based techniques do not result in detection of a large number of new faults [10]. Therefore, the faults found during the EAI inspection are attributable to the understanding of underlying source of the problem (errors).

Although the RIM specification document was chosen based on its successful usage in past studies, it may not be representative of a real industrial requirements document. This threat to external validity is slightly reduced as the document is written in a formal requirements specification notation, and it has been continuously improved over multiple studies [11, 26]. Another validity threat that comes from using a document such as RIM SRS is that we do not know all the faults and errors present in the document. The validity of faults as true-positives is based on one of the researcher's judgment and a list of faults created by researchers who have previously used RIM SRS in their studies [11, 26]. The subjects were not made aware of the hypotheses of our study, which can create a climate where subjects give favorable feedback instead of real feedback. The conclusion validity threat that comes with a small subject population could not be addressed in this study. We intend to replicate the study with the involvement of a larger number of subjects who will produce larger datasets for analysis purpose.

VI. Discussion of Results

This section provides a discussion of implications of the results of from Section IV. We compare the HET results against the results from RET studies that had similar designs.

A. *Summary of Findings*

This subsection presents the major results on each of the four hypothesis as follows:

H1- Fault Detection Effectiveness: EAI, supported by HET, on an average, increased the *fault detection effectiveness* by 225% as compared to conducting only FC inspection on the SRS (Fig. 4). The results from a one-sample t-test ($p<0.001$) showed that a significantly large number of faults were found during re-inspection that were left undetected during the first inspection (which was conducted using FC technique). These results show that adding EAI (supported by HET) to FC can provide significant gains in inspectors' fault detection effectiveness. Furthermore, even though the subjects were re-inspecting the same document during EAI inspection, the significantly large number of additional faults found shows that EAI is a very useful addition to FC for improving requirements quality. Regarding the *efficiency*, there was a small increase in efficiency when re-inspection effort was compared against the FC inspection effort. The efficiency of EAI decreased (was lower than FC) when the error abstraction and classification effort was included in the calculations. This decrease, however, is made up for by the high number of new faults found when using EAI. Also, in future we expect to combine error abstraction and re-inspection steps to further improve the efficiency of EAI process.

H2- Usefulness and Validity of HET: Subjects perceived HET useful in finding errors and faults during the SRS inspections. HET was rated rating significantly positive (larger than mid-point of a 5-point scale) on the following attributes: *simplicity (3.8)*, *ease of understanding* (4.0), *ease of use* (3.87), *intuitiveness* (3.73), *orthogonality* (3.73), *comprehensiveness* (4.2), *usefulness* (4.2), *worthiness* of HET in finding faults (4.2), and *confidence* that error classes represented real problems (4.2). Of all the attributes, *intuitiveness* and *orthogonality* of error classes received lowest scores (3.73). The "orthogonality" ratings were expected because HET has a three-dimensional structure wherein, the same human error (e.g., *Application Error* – a Mistake) can occur during different requirements engineering activities (e.g., during the requirements' elicitation, analysis and verification). During the post-study survey and discussion, the subjects' provided feedback that more examples of human errors and corresponding faults (in HET training document) should be added to help inspectors during the error abstraction step. The authors will improve the training materials for future studies.

H3- Major Insights: The results in Section IV.C revealed that *Clerical Errors* (*Slips*) were the source of majority of faults in RIM SRS. In terms of requirements activities, requirements *elicitation* and *specification* activities were most prone to *Clerical Errors* when RIM SRS was being created. This observation is consistent with the clerical error definition (*carelessness while writing specifications from elicited user need*s). More data needs to be collected before developing some intervention mechanisms (e.g., tools, training, changes in mechanical transcription tasks) that can be used to reduce the frequency of most commonly occurring human errors during the requirements development. Finally, one error class -NH (e.g., *not able to distinguish between client and users*) did not capture any faults in this study. More studies are needed before this error class can be removed from the taxonomy.

H4- Effect of Independent Variables: Individual performance during the EAI was positively correlated with several independent variables (not strongly correlated in all cases). Results showed that, *Process Conformance* (i.e., closely following the steps in error abstraction and re-inspection); and *Training Usefulness* (training and HEAA instrument) were strongly correlated with higher fault detection effectiveness during the EAI. *Effort* (time spent) applied, had a positive but not a strong impact on the fault detection effectiveness. Since we included time spent writing report of human errors in our measurement, we hope that inspectors using HET will be able to find faults faster. An interesting result showed that, an inspectors' performance using EAI during the *initial training* can be used to predict their performance during *transfer task*. This is a significant result with practical implications. ***First***, it was found that, number of new faults found in initial training was significantly correlated with the number of new faults found in transfer-session task during the error inspection. Therefore, project managers can staff the inspectors during an inspection of real project based on their training performance. ***Second***, it was also found that, an individual subjects' performance during the error abstraction/classification in initial training (i.e., error classification accuracy) was positively and strongly correlated ($p<0.001$) with their ability to find faults in transfer-session task. *Error classification accuracy* was calculated by dividing the number of errors that were correctly classified by the total number of reported errors. That is, subjects' who understand the error abstraction (and HET better) during the practice run are likely to perform better during the inspection on a real project. That is, training can improve subjects' understanding of EAI process and their eventual inspection effectiveness.

B. HET vs RET Results

The development of HET was motivated by the initial promising results of RET [9-12], so we compared current results to the results of RET studies to understand whether HET led to an improved inspection performance (vs. RET). RET study 1 [9] and study 2 [12] had a similar design to the current study (but in a different setting) wherein subjects performed an individual inspection using FC followed by using RET to guide the re-inspection of the same document. Based on the results; effectiveness of HET has been further improved (225% for HET vs. 75% - 156% for RET studies). Also, unlike RET*;* HET was rated favorably on all attributes.

VII. Conclusions and Future Works

In summary, EAI and HET can help provide significant gains in fault detection effectiveness. This was an initial investigation of HET and we intend to refine both HET and EAI using feedback provided by the subjects. The supplementary material like the Human Error Abstraction Assist (HEAA) will also be refined in order to make the error abstraction process less subjective and more systematic. In future, we plan to conduct more empirical evaluations based on the validity threats that were not addressed in this study. We also intend to evaluate the performance of EAI and HET in helping inspectors locate severe or critical faults. This evaluation will help software projects decide whether adding EAI to FC can help avoid the economic penalty that projects incur due to unfound faults of high severity. We also intend to evaluate HET and EAI in industrial settings.

A formal coding scheme to analyze the error abstraction data is being developed under the guidance of the Cognitive Psychology expert. This coding scheme will be developed based on the Cognitive Psychology principles of protocol analysis [27]. We plan to utilize protocol analysis to break down subjective abstraction data into small, meaningful bits of information. The coding scheme will help provide a sound evaluation technique for the human error abstraction data provided by subjects. This in turn will help HET and EAI in providing accurate insights about the requirements creation process. The coding scheme will also provide valuable information to the researchers in order to further refine both the HET and the EAI process.

Acknowledgment

This study was supported by National Science Foundation Awards 1423279 and 1421006. The authors would like to thank the students of the *Software Requirements Definition and Analysis* course at North Dakota State University for participating in this study.

References

[1] A. A. Porter, L. G. Votta, and V. R. Basili, "Comparing detection methods for software requirements inspections: a replicated experiment," IEEE Trans. Softw. Eng., vol. 21, no. 6, pp. 563–575, 1995.

[2] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, B. K. Ray, and D. S. Moebus, "Orthogonal Defect ClassificationA Concept for In-Process Measurements," IEEE Trans. Softw. Eng., vol. 18, no. 11, pp. 943–956, 1992.

[3] M. Leszak, D. E. Perry, and D. Stoll, "A case study in root cause defect analysis," in Proceedings of the 22nd international conference on Software engineering, ICSE 2000, 2000, pp. 428–437.

[4] IEEE std, "IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). Los Alamitos," CA IEEE Comput. Soc., vol. 610.12–199, 1990.

[5] REQB, "Standard glossary of terms used in Requirements Engineering," Requir. Eng. Qualif. Board, vol. 1.0, p. 24, 2011.

[6] A. Avižienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Trans. Dependable Secur. Comput., vol. 1, no. 1, pp. 11–33, 2004.

[7] F. Lanubile, F. Shull, and V. R. Basili, "Experimenting with Error Abstraction in Requirements Documents," in Proceedings of the 5th International symposium on software metrics, 1998.

[8] G. S. Walia and J. C. Carver, "A systematic literature review to identify and classify software requirement errors," Inf. Softw. Technol., vol. 51, no. 7, pp. 1087–1109, 2009.

[9] G. S. Walia, J. C. Carver, and P. Thomas, "Requirement Error Abstraction and Classification: An Empirical Study," in Proceedings of the 5th International Symposium on Empirical Software Engineering, 2006, pp. 336–345.

[10] G. S. Walia and J. C. Carver, "Using error abstraction and classification to improve requirement quality: Conclusions from a family of four empirical studies," Empir. Softw. Eng., vol. 18, no. 4, pp. 625–658, 2013.

[11] G. S. Walia and J. C. Carver, "Evaluating the use of requirement error abstraction and classification method for preventing errors during artifact creation: A feasibility study," in Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering, ISSRE, 2010, pp. 81–90.

[12] G. S. Walia, J. C. Carver, and T. Philip, "Requirement error abstraction and classification: A control group replicated study," in Proceedings of the 18th IEEE International Symposium on Software Reliability Engineering, ISSRE, 2007, pp. 71–80.

[13] P. C. Véras, E. Villani, A. M. Ambrosio, N. Silva, M. Vieira, and H. Madeira, "Errors on space software requirements: A field study and application scenarios," in Proceedings - International Symposium on Software Reliability Engineering, ISSRE, 2010, pp. 61–70.

[14] A. Esgate, D. Groome, and K. Baker, An Introduction to Applied Cognitive Psychology. Psychology Press, 2005.

[15] S. A. Shappell and D. A. Wiegmann, "Applying Reason: the human factors analysis and classification system (HFACS)," Hum. Factors Aerosp. Saf., vol. 1, no. 1, pp. 59–86, 2001.

[16] D. Wiegmann and C. Detwiler, "Human Error and General Aviation Accidents : A Comprehensive , Fine-Grained Analysis Using HFACS," Security, no. December, pp. 1–5, 2005.

[17] J. Reason, Human error. New York, NY: Cambridge University Press, 1990.

[18] J. Reason, Managing the risks of organizational accidents. Burlington, VT: Ashgate, 1997.

[19] J. Rasmussen and K. J. Vicente, "Coping with human errors through system design: implications for ecological interface design," Int. J. Man. Mach. Stud., vol. 31, no. 5, pp. 517–534, 1989.

[20] V. Anu, G. S. Walia, W. Hu, J. C. Carver, and G. Bradshaw, "Development of A Taxonomy of Requirements Phase Human Errors Using a Systematic Literature Review: A Technical Report," 2016, Technical Report. http://www.vaibhavanu.com/techrep/techrep2016.pdf

[21] J. Jennings, "Human factors analysis and classification: applying the Department of Defense system during combat operations in Iraq," Prof. Saf., vol. 53, no. 06, pp. 44–51, 2008.

[22] V. K. Anu, G. S. Walia, W. Hu, J. C. Carver, and G. Bradshaw, "Using cognitive psychology perspective on errors to improve requirements quality," 2015. Experimental Package. http://humanerrorinse.org/Studies/2015/Fall_NDSU/

[23] K. C. Hendy, "A tool for Human Factors Accident Investigation , Classification and Risk Management," Toronto, Canada, 2003.

[24] B. Gallina, E. Sefer, and A. Refsdal, "Towards Safety Risk Assessment of Socio-technical Systems via Failure Logic Analysis," in 2014 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2014, pp. 287–292.

[25] E. Sefer, "A model-based safety analysis approach for high-integrity socio-technical component-based systems," Mälardalen University Sweden, 2015.

[26] A. Goswami and G. Walia, "An empirical study of the effect of learning styles on the faults found during the software requirements inspection," in Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering (ISSRE), 2013, pp. 330–339.

[27] B. Robbins and J. Carver, "Cognitive factors in Perspective-Based Reading (PBR): A protocol analysis study," in Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM, 2009, pp. 145–155.

Appendix A: Human Error Abstraction Assist (HEAA)

1. Choose one of the following options to decide where the fault originated:
   (a) Did the fault occur
   - While the system was being analyzed?
   - While a large system was being divided into smaller parts?
   - While system functionalities (functional requirements) and system behavior (performance and other non-functional requirements) were being determined?

   (b) Did the fault occur during interviews or discussions with the stakeholders (end users, project sponsors, etc.)? This is where the user needs are gathered.
   (c) Did the fault occur when the system information/requirements were being documented to create a formal software requirements document?
   (d) Did the fault occur
   - During the *management* of the activities in a), b), or c) above?
   - As requirements evolved or changed (i.e., traceability, version control, etc.)

RE activity associated to each option:

<u>Option (a)</u> – Requirement Analysis.	**<u>Option (b)</u>** – Requirement Elicitation.
<u>Option (c)</u> – Requirement Specification.	**<u>Option (d)</u>** – Requirement Management

2. Please consider the task which was being performed when the fault was injected (i.e., when the human error occurred) and form a task/problem statement. For example,

"Analyzing the number of available parking spaces in the parking garage to arrive at a generic formula for calculating the number of available parking spaces (a = k-r)."

<u>Note that you will be asked to provide this task/problem statement in the 'Error-Report Form'</u>

The boxes below provide the human errors that are relevant to various RE activities. Based on your answer to Question# 1, go to the appropriate box and pick the human error you think caused the fault.
Refer to the **HET details document to get a detailed description of the human error and an example fault.
<u>Note that you will be asked to provide a brief description of why you picked a specific human error in the 'Error-Report Form'.</u>

<u>Requirement Analysis</u>

- **Application error:** requirement analyst's misunderstanding or lack of knowledge of a part of (or the whole) system or problem
- **Environment error:** misunderstanding or misuse of the requirement analysis tools available for use in the project
- **Wrong assumptions** made by requirement analyst about user/stakeholder needs or opinions or any incorrect assumptions by RE analysts
- **Low understanding of each other's roles**: RE analyst does not understand the roles of all end users, stakeholders and other RE analysts.
- **Mistaken belief** of RE analysts that it is impossible to specify non-functional requirements in a verifiable form
- **Problem-Solution errors:** Lack of knowledge of the requirement analysis process and general requirement engineering know-how

<u>Requirement Elicitation</u>

- **Clerical Error:** Carelessness while recording user needs
- **Loss of information from stakeholders:** Forgetting, discarding or failing to store information or documents provided by stakeholders.
- **Accidentally overlooking requirements**
- **Application error:** stakeholder's or requirement gathering person's misunderstanding of a part of (or the whole) system or problem
- **Environment error:** misunderstanding or misuse of the requirement gathering tools available for use in the project
- **Wrong assumptions** made by requirement gathering person about user/stakeholder needs or opinions or any incorrect assumptions made by requirement gathering person.
- **Low understanding of each other's roles:** Requirement gathering person does not understand the roles of all end users and stakeholders.
- **Mistaken belief** of requirement gathering person that it is impossible to specify non-functional requirements in a verifiable form
- **Not having a clear demarcation between client and users**: Requirement gathering person's misunderstanding of the difference between clients and users
- **Lack of awareness of sources of requirements**

<u>Requirement Specification</u>

- **Clerical Error:** Carelessness while documenting specifications from elicited requirements.
- **Lack of consistency In Requirement Specifications:** Lack of logical coherence in the requirement specification documentation, which makes it difficult to be interpreted correctly
- **Environment error:** misunderstanding or misuse of the requirement specification tools available for use in the project
- **Syntactic error:** Misunderstanding of grammatical rules of natural language (English) or grammatical rules of a formal requirement specification language.

<u>Requirement Management</u>

- **Inadequate Requirements Process:** All steps required to ensure a robust requirement engineering process are not followed
- **Information Management error**: lack of knowledge about standard procedures and practices defined by the organization