Empir Software Eng (2013) 18:625–658
DOI 10.1007/s10664-012-9202-3

Using error abstraction and classification to improve requirement quality: conclusions from a family of four empirical studies

Gursimran S. Walia · Jeffrey C. Carver

Published online: 28 March 2012

Editor: Murray Wood

Abstract Achieving high software quality is a primary concern for software development organizations. Researchers have developed many quality improvement methods that help developers detect faults early in the lifecycle. To address some of the limitations of fault-based quality improvement approaches, this paper describes an approach based on errors (i.e. the sources of the faults). This research extends Lanubile et al.'s, error abstraction process by providing a formal requirement error taxonomy to help developers identify both faults and errors. The taxonomy was derived from the software engineering and psychology literature. The error abstraction and classification process and the requirement error taxonomy are validated using a family of four empirical studies. The main conclusions derived from the four studies are: (1) the error abstraction and classification process is an effective approach for identifying faults; (2) the requirement error taxonomy is useful addition to the error abstraction process; and (3) deriving requirement errors from cognitive psychology research is useful.

Keywords Software inspections · Error abstraction · Software engineering · Software quality · Empirical studies

1 Introduction

Ensuring software quality is a major software engineering goal. Researchers have developed many quality improvement approaches and evaluated them using controlled experiments and case studies in both laboratory and realistic settings (e.g., (Sakthivel 1991; Chillarege et al. 1992; Florac 1992; Chaar et al. 1993; Lawrence and Kosuke 2004)). Researchers have

G. S. Walia
Department of Computer Science, North Dakota State University, Fargo, ND, USA
e-mail: gursimran.walia@ndsu.edu

J. C. Carver (✉)
Department of Computer Science, University of Alabama, Tuscaloosa, AL, USA
e-mail: carver@cs.ua.edu

devoted considerable effort to creating methods that help developers find and repair problems early in the lifecycle. Estimates are that 40–50% of development effort is spent on avoidable rework, i.e. fixing problems that should have been fixed earlier or should have been prevented (Boehm and Basili 2001). To eliminate unnecessary rework, the effectiveness of early-lifecycle fault detection and removal techniques must be improved.

Before proceeding, we need to clearly define two important terms: *error* and *fault*. Unfortunately, the software engineering literature contains competing, and often contradictory definitions of these two terms. In fact, IEEE Standard 610.12–1990 provides four definitions for the term error ranging from "incorrect program condition" (referred to as a program error) to "mistake in the human thought process" (referred to as a human error) (IEEE Std 610.12 1990). To allay confusion, we use the following definitions consistently throughout this paper. These definitions were originally given by Lanubile et al. (1998), and are consistent with software engineering textbooks (Endres and Rombach 2003; Pfleeger and Atlee 2006; Sommerville 2007) and IEEE Standard 610.12-1990 (IEEE Std 610.12 1990).

> **Error**—*mistake in the human thought process while trying to understand given information, solve problems, or use methods and tools. For example, in the context of software requirements, an error is a misconception of actual needs of a user or customer.*
> **Fault**—*concrete manifestation of an error within the software, one error may cause several faults and various errors may cause identical faults.*

The definition of an error used in this paper more closely correlates to the *human error* definition rather than the *program error* definition in IEEE Standard 610.12-1990. Using these two definitions, an error (mistake in the thought process) leads to one or more faults (mistakes recorded in a software artifact).

Most early-lifecycle quality improvement methods focus on the identification and removal of faults. To support these methods, researchers typically group related faults together based on their similar characteristics to form a taxonomy. Previous research shows that the similar faults occur in similar ways and require similar type of effort to repair (Carver 2003). Such a formal taxonomy of faults provides developers with a framework within which they can recognize the categories of problems and help them identify faults as they review a software artifact. Empirical evidence shows that various types of fault taxonomies are valuable for improving software quality (e.g., (Chillarege et al. 1992; Lezak et al. 2000; Carver 2003)). While they are useful, approaches based on fault taxonomies are not 100% effective in helping developers identify faults. They do not help developers identify high-level misunderstandings that may have led to fault insertion, causing some faults to be overlooked. Therefore, there is a need for additional approaches that can help overcome the deficiencies in the existing approaches and further improve software quality.

The main shortcoming of fault-based approaches is the absence of a direct mapping between errors (the source of the problem) and faults (the manifestation of the problem). For example, a common requirement fault, omission of important functionality, has at least two possible sources: 1) the requirement engineer may have lacked the requisite domain knowledge to realize that the functionality was needed, or 2) the missing functionality was needed by a stakeholder who was not included in the elicitation process. In order to understand and fix the problem, the development team must identify the omission and, more importantly, determine its cause. Understanding why the fault occurred should lead to identification of additional related faults.

We can make an analogy to medicine here. When a patient visits a doctor with symptoms of a disease, the doctor's goal is to identify the underlying disease causing the visible symptoms. If a doctor treats only the visible symptoms rather than the source of the symptoms, the patient will likely manifest other, possibly more serious, symptoms in the future. Therefore, a doctor will use the symptoms to diagnose and identify the underlying disease. With this knowledge the doctor can then more effectively treat the patient, including treating additional symptoms that may not have presented themselves yet.

Similarly, an error-based approach should be more effective in improving software quality than a fault-based approach because it treats the disease rather than the symptoms. Previously, other researchers have developed techniques that focus on the sources of faults, e.g., Root Cause Analysis and the Orthogonal Defect Classification (see Section 2.1). A taxonomy created by grouping related errors will help developers detect errors in the same way that fault taxonomies help developers identify faults. This paper investigates the use of an error taxonomy to help developers *detect errors* and the resulting faults during an inspection. If this approach is successful, it will help developers identify and eliminate errors, which will reduce the number of faults and increase software quality.

Because errors occur in the human thought process, it is reasonable to investigate the fallibilities of the human thought process in relation to software development. Therefore, we have exploited *human error* research advances made by cognitive psychologists. Human error research draws upon models of how human reasoning, planning, and problem solving processes can go awry. Exploiting a connection with cognitive research is particularly useful for understanding the causes of software faults. Case studies have shown that *human errors*, i.e., errors related to the general human cognitive process rather than specifically related to software engineering, occur during software development (e.g., (Lezak et al. 2000)).

It is challenging to integrate research findings from cognitive psychology with knowledge about software quality improvement. This integration is facilitated by an in-depth understanding of how the human cognitive process can fail during artifact creation. These cognitive failures (errors) result in software faults. The main goal of this paper is to present the results from four studies used to validate the usefulness of an error abstraction process supported by a requirement error taxonomy. We first briefly describe the requirement error taxonomy created based on information from software engineering research and cognitive psychology research.

The remainder of this paper is organized as follows: Section 2 discusses existing error-based quality improvement approaches, along with their limitations, to provide context for the research approach described in Section 3. Section 3 describes the error taxonomy and the framework used to evaluate it. Sections 4 and 5 describe the four empirical studies used to evaluate the requirement error taxonomy. Section 6 discusses the major findings and implications of the studies. Finally, the conclusions and future work appear in Section 7.

2 Related Work

Previously, other researchers have developed quality improvement approaches based on the source of faults. While these approaches have been effective, they have two shortcomings: 1) they typically do not define a formal error identification and repair process; and 2) they were developed based on a sample of observed faults rather than on a strong cognitive theory that provides comprehensive insight into human mistakes. Section 2.1 discusses three approaches that were developed based on the sources of faults. Section 2.2 provides an overview of how cognitive psychology research can help identify the sources of faults.

2.1 Approaches Based on the Sources of Faults

First, there are approaches that use the source of faults found late in the lifecycle to identify systematic problems to motivate process improvement (e.g. *Defect Causal Analysis, Root Cause Analysis*, *Software Failure Analysis*, and *Software Bug Analysis* (e.g., (Mays et al. 1990; Kan et al. 1994; Grady 1996; Card 1998; Nakashima et al. 1999; Lezak et al. 2000; Jacobs et al. 2005; Masuck 2005))). In a systematic literature review, we identified nine methods that fall into this class (Walia and Carver 2009). This section provides a brief summary of five of these methods.

In *Defect Causal Analysis*, an organization stores faults in a database and analyzes them to identify their cause and how they can be prevented (Card 1998). This process requires the software team to analyze a sample of the faults (as opposed to all of them) in order to understand the cause-effect relationships (i.e., a systematic investigation to identify the actions that produced observed consequences). The *Root Cause Analysis* approach provides a set of multi-dimensional *triggers* to help developers characterize the fault source (Lawrence and Kosuke 2004). It then suggests improvement actions to help developers detect faults earlier and significantly reduce the total number of faults. In *Software Failure Analysis*, developers analyze a sample of faults to identify common sources (e.g. User Interface faults). They then use a fishbone diagram as a brainstorming tool to combine and organize their thoughts about fault sources and provide recommendations for improvements (Grady 1996). The *Software Bug Analysis* process uses accumulated expert knowledge to identify the sources of faults rather than performing a detailed analysis of each fault (Jacobs et al. 2005). Our work builds upon the ideas embodied in these approaches and shifts the emphasis to faults that can be found early in the lifecycle (i.e., during the requirements phase) rather than late.

Second, the *Orthogonal Defect Classification (ODC)* is an in-process method for classifying faults into a predefined taxonomy. Developers identify the *trigger* that revealed the failure (not necessarily the source of the fault). Because the trigger is a concrete series of actions at the code level, identifying the trigger is more objective than identifying the source of the fault, which may be less clear. The ODC has provided useful feedback to developers (Chillarege et al. 1992). Our work builds on the concepts of the ODC by helping developers identify not only what caused a fault to be revealed, but more importantly what caused the fault to be inserted in the first place.

Third, in *Error Abstraction*, developers determine the source of faults detected during an inspection, i.e. the errors. After inspecting an artifact to identify faults, the inspectors perform the error abstraction step, i.e. determining the underlying errors that led to the faults. Inspectors then reinspect the artifact to identify additional faults related to the errors. This method showed promising initial results, but was not pursued further (Lanubile et al. 1998). One of the shortcomings of this method is that the error abstraction process is subjective and relies heavily on the expertise of the inspector. Our work extends the Error Abstraction method by providing concrete guidance in the form of an error taxonomy, which includes research from cognitive psychology (Section 2.2), to support inspectors during the error abstraction and reinspection process. Classifying errors into the error taxonomy (Section 3.1) helps developers gain a better understanding of the errors and guides their reinspection.

2.2 A Cognitive Psychology Perspective on Errors

While the psychological study of human errors began during the 1920's (Reason 1990), it received increased attention beginning in the early 1980's (e.g. (Norman 1981; Card et al.

1983)). This research was further motivated by tragic large-scale accidents, such as the Bhopal pesticide accident in 1984 and the Chernobyl nuclear power plant accident in 1986. Systematic human error models were built upon basic, theoretical human cognition research, especially from an information processing approach. It quickly became apparent that errors were not the result of irrational behavior, but rather resulted from normal psychological processes gone awry. Two competing schools of thought have contributed important perspectives and methods for studying and reducing human error. The first school of thought focuses on an individual's actions and psychological processes that result in errors. The second school of thought focuses on organization-level activities (e.g. problems in communication or training within an organization) that result in errors.

To explain errors made by individuals, the Generic Error-Modeling System (GEMS) identified three types of errors: 1) *Skill-based error*—routine action erroneously carried out in a familiar environment; 2) *Rule-based error*—a familiar if-then rule erroneously applied in an inappropriate situation; and 3) *Knowledge-based error*—mistake when reasoning about and planning solutions to novel problems or situations (Reason 1990). In software engineering, skill-based errors may appear during coding (e.g., a programmer forgot to include a buffer-overflow check even though he intended to) or during requirements and design (e.g., by omitting important details when documenting a familiar environment). Rule-based errors are more likely to occur during design with a designer selecting a familiar design pattern even though it is not appropriate for the current system. Finally, a knowledge-based error may occur when a software engineer produces an incomplete or incorrect set of software requirements because he failed to understand the unique aspects of a new domain.

A key assumption of the GEMS approach is that, when confronted with a problem-solving task, people tend to find a prepackaged rule-based solution before resorting to a far more effort-intensive knowledge-based solution, even when the latter is demanded at the outset (Reason 1990). The implication of this tendency is that software engineers are likely to employ familiar requirements engineering approaches even when these approaches are inappropriate for the current system and will lead to faults. The GEMS approach defines several skill-based, rule-based, and knowledge-based errors related to factors such as attention, information overload, and problems with human reasoning about if-then rules in familiar and novel situations.

At the organizational level, Rasmussen et al. (1982; Rasmussen et al. 1983) employed a similar skill-rule-knowledge framework. Rasmussen defines *skill* as actions which come naturally as a result of practice and require no conscious checking, *rule* as the type of behavior which follows a standard sequence developed through experience, and *knowledge* as the behavior exhibited when reacting to novel situations for which no standard working method is available. Rasmussen focused on the human information processing and corresponding knowledge states during the decision-making process. Accidents are caused by flaws in the organizational architecture that supports the individual human roles as an operator or initiator. By focusing at the level of the organization as the cause of accidents, Rasmussen's approach has helped foster error-tolerant systems by helping identify design flaws. This approach has been particularly valuable in industrial situations where accidents may have tragically large costs.

Even with these successes, the understanding of how to apply human error research to software engineering is not obvious. Errors in software engineering processes are somewhat different than the examples previously discussed. Therefore, the models of human error have to be adapted to software engineering. This adaptation will provide a better understanding of the means to reduce software engineering errors and the corresponding faults and should have a positive impact on software quality.

3 Research Approach

Our research combines information from software engineering and cognitive psychology to improve the requirement inspection process by focusing on errors. The two major goals of this research are: 1) to develop a *requirement error taxonomy,* and 2) to evaluate the use of the error taxonomy to support the error abstraction process. Figure 1 gives an overview of the process for developing and evaluating the requirement error taxonomy.

We developed the requirement error taxonomy by combining specific types of errors identified within the software engineering research together with cognitive psychology research about human errors. First, we developed an initial taxonomy (1. *Ad-hoc Review*) and empirically evaluated it (2. *Feasibility Study at Mississippi State University [MSU]*). Then, we refined the taxonomy using a more systematic approach (3. *Systematic Review*), and re-evaluated it using three additional empirical studies (4. *Control Group Study* and 5. *Observational Study* at *MSU;* and 6. *Control Group Replicated Study at North Dakota State University [NDSU]*).

Because the systematic literature review and the Requirement Error Taxonomy (V2.0) have already been published (Walia and Carver 2009), Section 3.1 just provides a summary of the taxonomy development. Section 3.2 provides an overview of the four empirical studies (with the details from each study described Sections 4 and 5).

3.1 Development of Requirement Error Taxonomy

The requirement error taxonomy helps inspectors identify and understand errors and provides guidance for identifying additional errors and faults. It has evolved through two versions. We used an ad-hoc review of the software engineering and psychology literature to develop the initial requirement error taxonomy (V1.0) (Walia 2006a). After the feasibility study (Walia et al. 2006b), we performed a *systematic literature review* to more completely identify and document the types of requirement errors (Walia and Carver 2009). This review included 149 papers from software engineering, human cognition and psychology. The systematic approach is commonly used in other fields (e.g. medicine) to draw high-level conclusions across a series of detailed studies (Kitchenham 2004).

To generate the taxonomy we grouped the errors identified in the software engineering, human cognition, and psychology literature into 14 detailed error classes which we then grouped into three high-level error types (Table 1). The three error types are *People Errors*, *Process Errors*, and *Documentation Errors*.

People Errors include mistakes caused by fallibilities of the individuals involved in project development. *Process Errors* are caused by mistakes in selecting the means of achieving goals/objectives and focus mostly on the inadequacy of the requirements

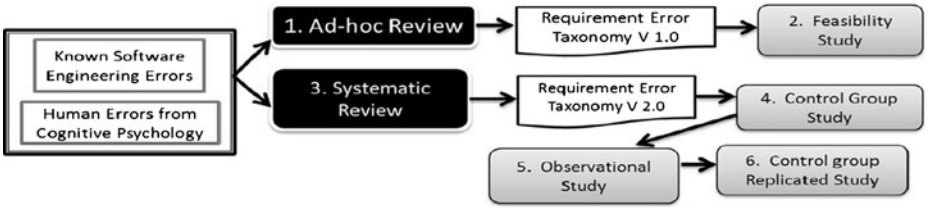

Fig. 1 Research approach

Table 1 Description of requirement error classes (Walia and Carver 2009)

Error Type	Error Class	Description
People Errors	Communication	Poor or missing communication among the various stakeholders
	Participation	Inadequate or missing participation of important stakeholders
	Domain Knowledge	Requirement authors lack knowledge or experience with problem domain
	Specific Application Knowledge	Requirement authors lack knowledge about specific aspects of the application
	Process Execution	Requirement authors make mistakes while executing requirement elicitation and development, regardless of the adequacy of the chosen process
	Other Cognition	Other errors resulting from the constraints on the cognitive abilities of the requirement authors
Process Errors	Inadequate Method of Achieving Goals and Objectives	Selecting inadequate or incorrect methods, techniques, or approaches to achieve a given goal or objective
	Management	Inadequate or poor management processes
	Elicitation	Inadequate requirements elicitation process
	Analysis	Inadequate requirements analysis process
	Traceability	Inadequate or incomplete requirements traceability
Documentation Errors	Organization	Problems while organizing requirements during documentation
	No Use of Standard	Problems resulting from the lack of using a documentation standard
	Specification	General documentation errors, regardless of whether the requirement author correctly understood the requirements

engineering process. *Documentation Errors* are caused by mistakes in organizing and specifying the requirements regardless of whether the developer correctly understood the requirement.

Each error class was derived from specific errors identified in the software engineering and human cognition literature. Specifically, errors related to human cognition span all three error types and fall into seven of the fourteen error classes: *Domain Knowledge*, *Specific Application Knowledge*, *Process Execution*, *Inadequate Method of Achieving Objectives*, *Organization*, *Specification*, and *Other Cognition.*

The complete description of the systematic review process, details of the requirement error taxonomy and examples of errors and related faults for all 14 error classes can be found in the systematic literature review (Walia and Carver 2009) and in Appendix A. To illustrate the error taxonomy, we provide an example *participation error* along with a related fault taken from a generic ATM system:

> *Error*: An important stakeholder (e.g., a bank manager in an ATM system) was not involved in the requirement gathering process;
> *Fault*: Some functionality (e.g., handling multiple ATM cards simultaneously at different machines) was omitted.

3.2 Evaluation of Requirement Error Taxonomy

To evaluate the usefulness of the requirement error taxonomy and the error abstraction process, we conducted a family of four controlled experiments (studies one, two and three at MSU and study four at NDSU). A family of studies is a set of related studies that focus on the same research question or hypotheses (Basili et al. 1999). While each of our studies tested the same basic hypotheses, the designs of later studies were slightly modified based on

the lessons learned during the earlier studies. As shown in Fig. 2, the results from Study 1 motivated the design of Studies 2 and 3 (by adding a control group variable and an observational variable respectively). Similarly, the results from Study 2 motivated the design of Study 4 (by replicating Study 2 in a different university setting).

Section 3.2.1 discusses the hypotheses and defines the independent and dependent variables common to all studies. Sections 3.2.2 through 3.2.5 provide a high-level overview of each study along with its major results to illustrate the evolution of studies and provide context for the more detailed descriptions of the studies that appear in Sections 4 and 5. The goal of this paper is to highlight important results from each study and to draw conclusions across all four studies, rather than report all the details of each study. Studies 1 and 2 have each been reported in detail (Walia et al. 2006b; Walia et al. 2007).

3.2.1 Study Hypotheses and Variables

Table 2 defines the independent and dependent variables common to all the studies. Table 3 contains the five hypotheses tested in each study. Hypothesis 5 investigates the effect that the five independent variables defined in Table 2 have on the two dependent variables.

3.2.2 Study 1: The Feasibility Study

Study 1 evaluated the initial version (V 1.0) of the requirement error taxonomy. This study was a repeated-measures quasi-experiment (without a control group) conducted in a senior-level full-year capstone course in which students developed a real system for a real client. Students first developed a requirements document and then inspected it using a fault checklist. They then used the error abstraction and classification process along with the requirement error taxonomy to guide a reinspection of the requirements. The study results indicated that the error taxonomy was easy to use and led to detection of a significant number of faults during the reinspection. The main weaknesses in this study were 1) the lack of a control group and 2) the lack of detailed insight into why the observed results occurred.

The promising results from Study 1 motivated the systematic literature review and the design of Study 2 (adding a control group) and Study 3 (including a participant observer) to address the weaknesses in Study 1.

3.2.3 Study 2: A Control Group Replicated Study

It was not clear from Study 1 whether the faults found during the reinspection would have been found by anyone performing a reinspection or whether they were found because of the

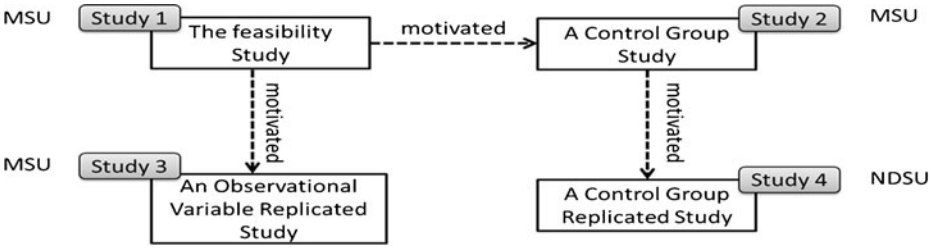

Fig. 2 The family of experiments

Table 2 Study variables

Independent Variable	Definition
Process conformance	Measures how closely participants follow the error abstraction, classification and re-inspection steps
Pre-test	Measures the performance of participants on using the requirement error taxonomy during an in-class exercise
Training usefulness	Measures the perceived usefulness of training procedure for each participant
Effort spent	Amount of time spent during each step of error abstraction and classification process (i.e., error abstraction, error classification, and re-inspection)
Difficulty level	Measures the degree of difficulty the students perceived while performing the experimental tasks

Dependent Variable	Definition
Effectiveness	The number of faults found
Efficiency	The number of faults found per hour

error abstraction process and taxonomy (V2.0). Therefore, the goal of Study 2 was to replicate Study 1 with the addition of a control group to address this uncertainty.

The experimental group followed the same process as in Study 1. They performed an inspection with a fault checklist, followed by a reinspection using the error abstraction and classification process. In contrast, the control group performed two inspections (without using the error abstraction and classification process during the second inspection). We compared the number of faults found by the control group during the reinspection with the number found by the experimental group during reinspection to determine whether there was a difference.

The major result of this study was that the experimental group found significantly more faults than the control group during the reinspection. This result indicates that those using the error abstraction and classification process were more effective than those simply performing two inspections. The main weakness in Study 2 was that the control group used the same inspection method for the initial inspection and for the reinspection, possibly limiting the effectiveness of the reinspection. Study 4 addresses this weakness.

3.2.4 Study 3: An Observational Variable Replicated Study

The goal of Study 3 was to provide further insight into the results of Study 1 through qualitative observations about the participants' thought process while using the error abstraction and classification process. We achieved this goal by replicating Study 1 with the addition of a participant observer who observed the interactions among the participants and the client during the requirement development meetings and among the participants during

Table 3 Study hypotheses

Hypothesis #1	The *error abstraction and classification* approach improves the effectiveness and efficiency of inspection teams and individual inspectors
Hypothesis #2	The *requirement error taxonomy* is useful for helping inspectors find errors and faults
Hypothesis #3	The *requirement error taxonomy* provides important insights into the major sources(s) of requirement faults
Hypothesis #4	The contributions from the *cognitive psychology* field help inspectors locate more faults
Hypothesis #5	Other *independent variables* (process conformance, performance on a pre-test, usefulness of training, effort and perceived difficulty) affect the individual performance during the error abstraction and classification process

inspection team meetings. The observer noted any errors made during development to see whether the participants identified those errors during the inspection.

The results showed that developers found the error taxonomy (V2.0) easy to use and effective. The observational data also provided useful insights to augment the findings of Study 1 (e.g., the inspection team meetings that were informed by the error abstraction and classification process were more engaging and effective at finding new errors and faults than the team meetings informed only by the fault checklist technique).

3.2.5 Study 4: A Control Group Replicated Study

During Study 2, there was a possibility that the control group participants were less motivated during the second inspection because they were using the same fault checklist as they used in the first inspection. To address this maturation threat, Study 4 replicated Study 2 with the addition of the *Perspective Based Reading* (PBR) technique during the second inspection. PBR is a mature inspection technique in which inspectors 1) read the artifact from a specific stakeholder's point of view (e.g., user, designer, or tester), 2) follow a series of steps to create an abstraction of the artifact based on their perspective (e.g., a PBR designer produces a system design), and 3) use this abstraction to answer questions (derived from a fault taxonomy) to find faults. PBR has been empirically validated as an effective fault detection technique (Basili et al. 1996).

In Study 4, all participants performed the first inspection using the fault checklist technique, as in Study 2. However, during the second inspection, the participants in the control group used the PBR technique (as opposed to reusing the fault checklist as in Study 2) and the participants in the experimental group used the error abstraction and classification process (the same as in Study 2).

The results showed that the experimental group was significantly more effective during the second inspection, providing further evidence that using the error abstraction and classification process yields betters results than using a fault-based requirements inspection technique.

3.2.6 Organization of the Discussion of the Four Studies

Rather than discussing the four studies in chronological order, to reduce duplication we have grouped the studies based on study type. Section 4 discusses Studies 1 and 3 which evaluate the use of the requirement error taxonomy in the context of a real world project developed by university students without a control group. Similarly, Section 5 discusses Studies 2 and 4 which are classic control-group studies. All the experiment materials (including the training slides, training documents, fault and error forms and the requirement document) are available online at https://cs.ndsu.edu/research/experiments/walia/usingerror.htm

4 Study 1 and Study 3: Factorial Design Replicated Studies

The goal of Study 1, a repeated measures quasi-experiment, was to understand whether V1.0 of the requirement error taxonomy could be effectively used in a traditional inspection process (Walia et al. 2006b). Study 3 was a replication of Study 1 with two changes. First, it evaluated V2.0 of the requirement error taxonomy. Second, it placed additional emphasis on qualitative data by using a *participant observer* to capture firsthand observations about the behavior and interactions among the participants and the client. The main goal of this change was to gain insight into the human thought process related to requirement error and faults.

4.1 Study Design

The participants in both studies were senior-level computer science or software engineering students enrolled in the capstone course at MSU during two different years. The capstone course at MSU is a two–semester sequence. During the Fall semester, the students work with their clients (identified and assigned by the instructor) to elicit and document the requirements for a system that they will implement during the Spring semester. Study 1 and Study 3 were both conducted during the Fall semester of their respective capstone projects. In each study, the participants were divided into two independent teams that created their own requirements document. Table 4 provides information about the teams.

Note that while both teams in Study 3 developed a conference management system, they worked independently and created separate requirement documents and separate systems.

The studies began after each team developed their initial requirement specification. Figure 3 illustrates the study design, with details provided in Table 5. Because Study 3 is a replication of Study 1, the only procedural difference between the two studies was the presence of the participant observer.

4.2 Data Collection

This section provides a brief description of qualitative and quantitative data collected during Study 1 and Study 3. The quantitative data includes the faults found by participants using the fault checklist technique (i.e., during the first inspection) and the faults and errors found by the participants using the error abstraction and classification process (i.e., during the second inspection). The fault reporting forms also provide the participants with space to indicate timing information, including the start and end times of the inspection, the time they found each fault, and any breaks they took. In addition, the fault reporting forms used during the Study 3 required the participants to rate the *importance level* and the *severity* of the faults identified during the first and second inspection cycle. The importance level, which is the potential that a particular requirement fault found during inspection can cause a redesign of the software, was classified using the following scale (0- not important, designer should easily see the problem; 1—problem, if a failure occurs it should be easy to find and fix; 2—important, if a failure occurs it could be hard to find and fix, 3—very important, if a failure occurs it could be very hard to find and fix, 4—if a failure occurs it could cause a redesign). The severity, which is the probability that a particular fault will cause a system failure, was classified using the following scale (0—will not cause failure, regardless whether it is caught by the designer; 1—will not cause failure, because it will be caught by the designer; 2—could cause a failure, but most likely be caught by the designer, 3—would cause a failure, will most likely not be caught by the designer).

One of the researchers validated that the faults reported by each participant were true-positives. The researcher, who had knowledge of the system for which the requirements were developed, read through the faults reported by each participant to remove any false-

Table 4 Study 1 and Study 3: Teams, participants and systems

Study #	Semester	Team#	Number of Participants	System
1	Fall 2005	1-A	8	Starkville theatre system
		1-B	8	Apartment Management
3	Fall 2006	3-A	6	Conference Management
		3-B	6	

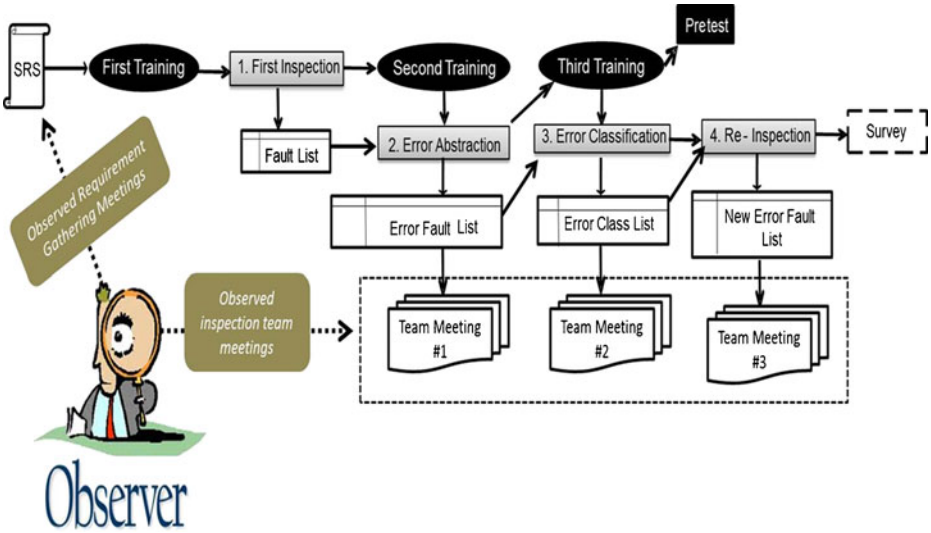

Fig. 3 Experiment design for Studies 1 and 3 (observational aspect in dotted lines)

positives before analyzing the data. If any faults were unclear, the researcher clarified them with the participant to accurately determine the validity of the fault. Regarding the evaluation of the errors reported by participants, the researcher read through the errors to ensure that the description of each error represented a real mistake or misunderstanding that could have happened during the development. Also, the researcher evaluated the errors for correct classification by making sure that the description of the error was consistent with the actual description of that error class.

For qualitative data, we gathered both subjective self-report data and data from the participant observer. Using a three-point scale in Study 1 (ranging from "low" to "high) and a five-point scale in Study 3 (ranging from "very low" to "very high", the participants rated the requirement error taxonomy (V1.0 in Study 1 and V2.0 in Study 3) on ten

Table 5 Study 1 and Study 3: Experiment steps, training, and outputs produced

	Experiment Steps Performed by Participants in Study 1 and Study 3
Training 1 – Fault Checklist	Trained on how to use fault checklist to find faults
Step 1 – Requirements Inspection	Each participant inspected the requirements to identify faults. ***Output – Individual Fault Lists***
Training 2 – Error Abstraction	Error Abstraction – participants were trained on how to abstract errors from faults
Step 2 – Abstraction of Errors	Each participant abstracted errors from their own fault lists from Step 1. ***Output – Individual Error-Fault Lists***
Training 3 – Requirement Error Taxonomy	The taxonomy was explained in detail. Participants were taught how to classify errors. They were given an in-class error classification exercise. The exercise was debriefed to ensure the students understood the classification process. Finally, the participants were taught how to use the classified errors to guide reinspection
Step 3 – Classify Errors	Participants classified errors they identified during Step 2. ***Output – Individual Error-Class Lists***
Step 4 – Reinspection	Each participant used their classified errors from Step 3 to reinspect the requirements. ***Output – Individual New Fault Lists***
Post-study Survey	Focused on gathering feedback about error abstraction and classification process.

characteristics, including: simplicity (sim), usability (usa), orthogonality (orth), usefulness (use), understandability (und), intuitiveness (int), comprehensiveness (comp), applicability of errors across different software products (aesp), adequacy of error classes (aec), and ease of classifying errors (ece). The participants also rated their level of *process conformance* during the first inspection and during the second inspection on a 3-point (Study 1) or 5-point (Study 3) scale.

During Study 3, the first author acted as a participant observer during portions of the requirements development and review process. Seaman et al., noted that observation of communication among software developers during team meetings is an easy and important way to understand their thought processes (Seaman and Basili 1997; Seaman 1999). In this study the participant observer used the requirement error taxonomy during the requirements gathering meetings to document the observed errors and took *field notes* during the inspection team meetings to collect data on the length and nature of discussion and interactions between team members. The participant observer did not play any direct role during the requirement gathering or inspection team meetings.

Observations During the Requirements Gathering Meetings During these meetings, the client explained the requirements to members of both teams together, who were given an opportunity to ask questions for clarification. Later, each team met separately with the client to clarify any questions and gather additional requirements. By observing these meetings and documenting any observed errors, we could later check whether the participants identified those errors during the error inspection. The data collected included notes about the communication between the participants and the client.

Observations During the Inspection Team Meetings Prior to any team meetings, each participant independently performed the inspection, error abstraction and error classification tasks. There were then three team meetings in which the team members discussed their individual lists to agree on a final team list of errors or faults (depending on the stage of the process). These meetings occurred after the first inspection, after the error classification, and after the reinspection. The observer was present at each meeting to observe the communication and interactions among participants and to record any additional faults or errors identified during the team meetings.

4.3 Data Analysis and Results

This section describes the results from Studies 1 and 3 relative to the five hypotheses presented in Section 3. It also provides qualitative insights from the participant observer from Study 3. In all statistical analyses, an alpha value of 0.05 is used to judge the statistical significance of the results.

4.3.1 Hypothesis 1: Improved Inspection Effectiveness and Efficiency

We analyzed effectiveness by comparing the total number of new, unique faults found by each team during the second inspection (using the error abstraction and classification process) to the number of faults found during the first inspection (using the fault checklist). The goal of this analysis was to determine whether a reinspection guided by the error taxonomy would be effective at detecting the faults that were missed when using the fault checklist during the first inspection.

As Fig. 4 shows, the teams showed an increase (*faults found during reinspection/faults found during first inspection*) of between 75% and 156% when using the error abstraction and classification process. Therefore, using error abstraction and classification to inform the reinspection was highly effective at helping inspectors detect faults that were missed during the first inspection. Further analysis of the data from each participant showed that, while some had a larger increase than others, all participants found new faults during the second inspection.

Efficiency is calculated as *faults/time*. For the first inspection, *time* includes the time taken during first inspection (i.e., Step 1). For the second inspection, *time* includes the time spent during error abstraction, error classification, and reinspection (i.e., Steps 2, 3, and 4). The average efficiency value during the first inspection for the participants on team 1-A (9 faults/hour) and team 1-B (11 faults/hour) was greater (though non-significant) than the efficiency values during the second inspection for the participants on team 1-A (8 faults/hour) and team 1-B (8 faults/hour). The efficiency values for the participants in Study 3 followed a similar trend. The efficiency values of the participants on team 3-A (13 faults per hour) and 3-B (10 faults per hour) during the first inspection was also not significantly different than the efficiency of the participants on teams 3-A (11 faults per hour) and 3-B (12 faults per hour) during the second inspection. To provide more detail, Table 6 shows the min, max and average number of minutes spent by the participants in the 1st Inspection, the Error Abstraction/Error Classification step and the 2nd inspection.

The results showed that the error abstraction and classification process did not significantly affect efficiency in either study. There was actually a slight decrease in efficiency during the second inspection for teams in Study 1.

4.3.2 Hypothesis 2: Usefulness of the Requirement Error Taxonomy

The participants evaluated the requirement error taxonomy on the ten characteristics listed in Section 4.2. Table 7 shows the mean value of the ratings for each characteristic in both studies. For each characteristic, we conducted a non-parametric one-sample Wilcoxon Signed Rank test to determine whether the mean response was significantly greater than the midpoint of the scale. The shaded cells in Table 7 represent significantly positive characteristics (i.e., $p<0.05$). In Study 1, only half of the characteristics were rated significantly positive whereas in Study 3, all characteristics were rated significantly positive. None of the attributes in either of the studies was rated negative (i.e., the mean value was significantly less than the mid-point of the scale). There are two potential reasons that error taxonomy was rated more positively in Study 3: 1) the use of 5-point scale rather than a 3-

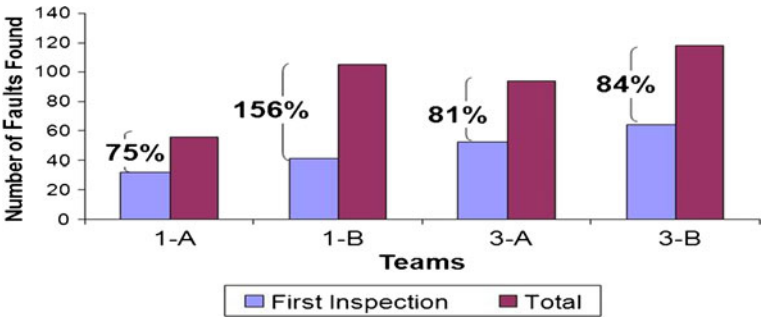

Fig. 4 Increase in the number of faults found during second inspection

Table 6 Time spent in 1st inspection, error abstraction/classification and 2nd inspection

	1st Inspection Time Min/Avg/Max (in minutes)	Error Abst/Class Time Min/Avg/Max (in minutes)	2nd Inspection Time Min/Avg/Max (in minutes)
S1 – Team A	30/61/81	27/79/117	18/54/145
S1 – Team B	21/54/86	44/56/79	40/72/100
S3 – Team A	25/63/92	59/82/98	58/73/96
S3 – Team B	42/100/142	62/87/102	65/89/110

point scale, and 2) use of a revised version (V2.0) of the taxonomy. These results show that, overall, the participants had a positive impression of the requirement error taxonomy.

4.3.3 Hypothesis 3: Insights into the Major Source(s) of Requirement Faults

This section investigates three major questions. Question 1: *Do the three error types make significantly different contributions to the overall fault rate?* The first eight rows in Table 8 (i.e. those related to *Total Errors* and *Total Faults*) show the number of total errors and faults (under the column labeled "N") and the percentage contribution of errors and faults across the three error types. In both studies, the *People Error* type led to the most errors and faults.

Table 8 also shows the p-values from chi-square tests to evaluate whether the observed distribution was significantly different from a uniform distribution, that is, to determine if one error type was disproportionately represented. In seven of the eight cases, the distribution was significantly different than uniform.

Question 2: *Is each of the 14 error classes valid and necessary?* The results showed that in each study there was at least one fault caused by an error from each error class, indicating that each error class is valid and necessary. Further analysis of the distribution of the total errors and faults across the 14 error classes showed that there was no single error class that was responsible for the majority of faults. For example, amongst *People Errors*, Team 1-A found a larger number of *Communication* errors and *Participation* errors, whereas Team 1-B found a larger number of faults caused by *Domain Knowledge* errors and *Specific Application Knowledge* errors. Similar trends were observed for the *Process Errors* and the *Documentation Errors* in both studies.

Question 3: *Which error type(s) are the major source of redundant faults, time consuming faults, multiple faults, important faults and severe faults?* Redundant faults are faults found by more than one participant. Time consuming faults are faults that took longer than the average time to locate (12 min in Study 1 and 10 min in Study 3). Errors that cause multiple faults are errors that are the source of more than one fault. The importance attribute has five levels (0-not important to 4-highly important). The severity attribute has four levels (0-not

Table 7 Study 1 and Study 3: Mean value of the ratings of the requirement error taxonomy on different characteristics

	Middle point	1. simp	2. usa	3. orth	4. use	5. und	6. int	7. comp	8. aesp	9. aec	10. ece
Study 1	Medium-"2"	2.13	2.31	2	2.3	2.1	2.21	2.37	2.3	2.6	2.12
Study 3	Medium-"3"	4.08	4.1	4	4.08	4.16	3.67	4.08	4.25	4	4.1

Table 8 Study 1 and Study 3: Insights provided by the requirement error taxonomy

Variable	Team	N	People Errors	Process Errors	Documentation Errors	p-value
Total Errors	1-A	19	52%	32%	16%	**< .001**
	1-B	50	59%	24%	17%	**< .0001**
	3-A	59	50%	30%	30%	**.026**
	3-B	59	38%	35%	27%	.379
Total Faults	1-A	55	50%	33%	8%	**< .001**
	1-B	105	78%	9%	13%	**< .0001**
	3-A	94	57%	27%	16%	**< .001**
	3-B	118	47%	19%	34%	**.003**
Redundant Faults	1-A	44	64%	29%	7%	**< .001**
	1-B	37	64%	23%	14%	**< .001**
	3-A	53	63%	21%	16%	**< .0001**
	3-B	37	49%	17%	34%	**< .0001**
Time-Consuming Faults	1-A	8	38%	62%	0%	**.016**
	1-B	12	25%	58%	17%	**< .001**
	3-A	7	24%	68%	8%	**< .0001**
	3-B	13	31%	54%	15%	**< .0001**
Multiple Faults	1-A	7	80%	20%	0%	**< .001**
	1-B	9	60%	36%	4%	**< .001**
	3-A	14	47%	15%	38%	**< .0001**
	3-B	11	38%	11%	51%	**< .0001**
Important Faults	3-A	35	65%	21%	14%	**< .0001**
	3-B	26	52%	39%	9%	**< .0001**
Severe Faults	3-A	24	71%	29%	0%	**< .0001**
	3-B	19	58%	37%	5%	**< .0001**

severe to 3-will cause failure). Important faults are those with values of 2, 3, or 4 (ranging from important to highly important). Severe faults are those with values of 2 or 3. Importance and severity were only collected during Study 3.

Again a chi-square test is used to evaluate whether the distribution is significantly different from uniform. Table 8 shows the p-value from the chi-square test regarding the contribution of the *People*, *Process*, and *Documentation* errors to the redundant faults, time-consuming faults, multiple faults, important faults, and severe faults. In both studies, *People Errors* were significantly more likely to cause redundant, multiple, important and severe faults in all but one case (for team 3-B *Documentation Errors* were more likely to cause multiple faults). However, *Process Errors* actually led to more time consuming faults.

4.3.4 Hypothesis 4: Usefulness of Cognitive Psychology

Some of the error types were derived from cognitive psychology research (as explained in Section 3.1). Errors related to cognitive psychology accounted for at least 1/5 of the errors found by each team (1-A → 21%, 1-B → 26%, 3-A → 32%, and 3-B → 24%). This result was fairly consistent across all participants. The results show that all but two of the participants in Study 1 and all but one participant in Study 3 found at least one human cognition error. In fact, one participant found only human cognition errors.

Figure 5 shows the breakdown of the percentage of the faults found during each inspection that were attributed to the errors derived from cognitive psychology research. A larger percentage of the faults found in the second inspection could be attributed to the cognitive psychology errors compared with those found in the first inspection. These results

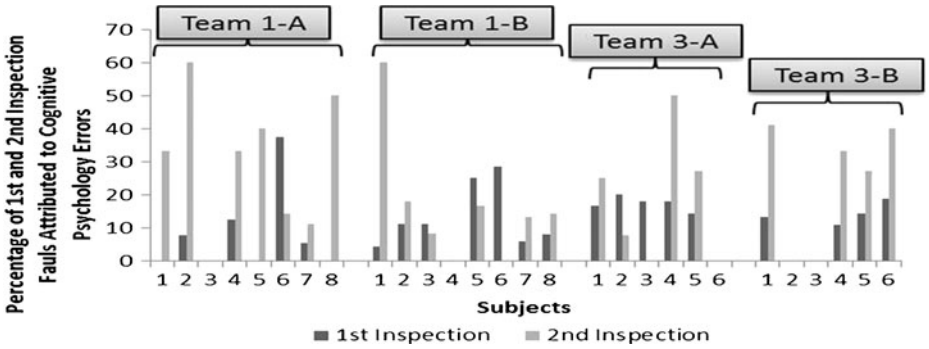

Fig. 5 Percentage contribution of cognitive psychology errors to the faults found during the first inspection and the second inspection by teams 1-A, 1-B, 3-A, and 3-B

indicate that errors related to Cognitive Psychology research are useful and important to include in the requirement error taxonomy.

4.3.5 Hypothesis 5: Effect of Independent Variables

We analyzed the effects of the five independent variables on the dependent variables (Table 2) using multiple regression (Field 2007). The goal was to find any significant correlations between those variables and participant effectiveness during the reinspection. The process conformance, training usefulness and difficulty level variables were measured on a 3-point (Study 1) or 5-point (Study 3) scale. The pre-test variable (the number of errors correctly classified during an in-class exercise) and effort variable (the amount of time spent) were measured with the actual numbers. The results show that:

- *Pre-test* performance had a weak positive correlation to effectiveness during the reinspection for Study 1 (r^2=.366, p=.017). However, for Study 3, the correlation was stronger, but not significant (r^2=.548, p=.065). Better performance on the pre-test indicates a better understanding of the error taxonomy.
- In Study *1, effort* had a weak positive correlation to efficiency (r^2=.331, p=.02). This result is interesting because it shows that the more effort spent, the more faults/hour found (not just more faults found). There was no correlation in Study 3.
- *Usefulness of training* was significantly correlated with effectiveness in reinspection for Study 3 only (p=.020), i.e. participants who found the training useful were more effective.
- *Process conformance* had no correlation with effectiveness or efficiency in either study.
- *Difficulty level* had no correlation with effectiveness or efficiency in either study.

The multiple regression analysis did not show any significant positive correlations among the independent variables.

4.4 Additional Insights from the Participant Observer

Only Study 3 had a participant observer, so this analysis is based only on Study 3. During the *requirement gathering team meetings*, the participant observer recorded any errors he observed during development of the requirement documents. This list of errors did not include errors that might have occurred outside of the team meeting. The observer noted 17 errors made by team 3-A and 22 by team 3-B. When we compared these lists of errors

against the final team error list, team 3-A found 13/17 errors and team 3-B found 20/22 errors. All of these errors were identified by the participants during their individual inspections prior to the team meetings. Therefore, the requirement error taxonomy was useful for the observer to identify errors committed during requirements development and useful for helping inspectors find the errors. (Note: we did inform the teams of the remaining errors after study completion before they implemented their systems).

For the data collected during the *inspection team meetings*, we performed four analyses. First, we wanted to understand whether the meeting led to additional fault or error identification. There were two meetings related to faults (meetings 1 and 3) and one meeting related to errors (meeting 2). During meeting 1, after the first inspection, team 3-A found one new fault and team 3-B zero new faults. During meeting 3, after the reinspection, team 3-A found five new faults and team 3-B three new faults. During meeting 2, after error abstraction and classification, team 3-A found one new error and team 3-B three new errors. The team meetings led both teams to identify problems that had been missed by individuals. Specifically, the participants were more effective at identifying new issues in meetings 2 and 3, which were informed by the error abstraction and classification process than they were during meeting 1.

Second, we analyzed the thought processes of the inspectors as they discussed the inspection results. These discussions led to the identification of new faults not found by any individual. For example, one member of team 3-A indicated that the team leader did not communicate an important requirement properly which lead to incorrect documentation. During this discussion, the team members were able to find two faults introduced in that requirement. Often during such interactions, other team members contributed to the discussion especially in terms of how such errors affected other parts of the requirements document. The results of this analysis indicated that after discussing the errors with each other, the participants better understood the requirement problems and the faults that were likely to result. Therefore, the participants found additional faults.

Third, we analyzed the content of the discussions. The discussions were more engaging and longer during team meetings 2 and 3, which were informed by the error abstraction and classification process, than during team meeting 1. Specifically:

- During team meeting 1, the discussion focused mainly on deciding whether the fault(s) found by individuals made sense to the rest of the team. Team members also eliminated redundant faults and compiled the team fault list. The team members seemed to disagree mostly on the type of fault (e.g., incorrect fact, ambiguous) with some instances where no clear consensus was reached.
- During team meeting 2, often different team members had classified the same error into different error classes. The discussion focused mainly on agreeing on one compiled list of errors and their classification. While there were initial disagreements, the teams always reached consensus.
- During team meeting 3, the team members discussed different faults they found which were caused by the same error. While compiling the individual fault lists into a team list, they identified new faults. The team discussed these faults and included them on the final list. Each team found more faults in this meeting than during team meeting 1.

Finally, we wanted to understand the pattern of discussion about faults and errors that inspectors had identified individually. This analysis was performed to understand whether discussing the root cause of fault (i.e., error) with the other team members could highlight other faults and errors that were not found during the individual inspections. Because the team leader's job was to compile the list, he was always the center of the discussion.

However, we observed a pattern that after a team member finished describing their own errors or faults, they became less interactive. Also, the new faults and errors that arose during the discussions did not originate from a single person or small group of people. Rather, various team members were involved in identifying different faults or errors.

4.5 Validity Threats

We faced the following threats to validity in one or both studies.

Conclusion Validity The threat due to the heterogeneity of participants was controlled because all participants were drawn from the same course and had same level of education. The conclusion validity threat caused by the use of a 3-point rating scale in Study 1 was addressed by increasing the scale to a 5-point scale in Study 3. However, there remains a threat that we treated these scales as an interval scale rather than ordinal scale, following the standard practice in the social sciences.

External Validity To increase external validity, Studies 1 and 3 were conducted in a capstone course where the participants worked with a real client to develop requirements for a system that they later implemented. However, there remains a threat because the participants were all undergraduate students in an educational setting and likely do not represent professional developers.

Construct Validity A threat exists due to the participants' self-evaluation of the "importance" and "severity" of the faults. Similarly, the participants self-classified other independent variables (e.g., process conformance, training usefulness). This threat is somewhat reduced because the participants were actually planning to implement the requirements and should have been objective in judging the importance and severity of faults.

Internal Validity The most important internal validity threat in both studies was the lack of a control group. We cannot determine the proportion of the faults found during reinspection that were due to the use of error abstraction and classification and the proportion that would have been found without the error abstraction and classification process. To address this threat, Studies 2 and 4 (discussed in next section) added a control group. Also, there are two threats related to the errors that were recorded by the participant observer during the team meetings. First, there is a subjectivity bias because the participant observer is one of authors and therefore could have subconsciously focused on only observing errors that were included in the error taxonomy. Secondly, the participant observer only collected errors during the requirement gathering meetings and could not observe or note any errors made outside the team meetings. Therefore, it is possible that teams could have made more errors during development that were not detected during the inspection. Also, we did not collect any data regarding faults that might have occurred after the study, i.e. during implementation.

5 Study 2 and Study 4: Control Group Studies

Studies 2 and 4 utilized a pretest-posttest control group quasi-experiment design to understand whether faults found during the reinspection step are due to use of the error abstraction and classification process or due simply to inspecting the artifact for a second time.

5.1 Study Designs

Study 2 The 18 participants were graduate students enrolled in either the Software Verification and Validation (V&V) course or the Empirical Software Engineering (ESE) course at MSU. The V&V course covered quality improvement approaches including software inspections. The ESE course focused on empirical study design and data analysis. During this two-week study, the participants inspected a natural language requirements specification document for a data warehouse system that was developed by the Naval Oceanographic Office. The document was 30 pages long and included the overview (scope and purpose of the system), the functional requirements, and other (e.g., security, performance, interface) requirements. The participants did not develop the requirements document, nor did they have access to any of the developers of the requirement document. The participants were divided into a control group, which included nine students from the V&V course and an experiment group, which included eight students from the ESE course. Four participants were enrolled in both courses. To balance the groups, these participants were randomly assigned to only one group and were not aware of the activities of other group.

Study 4 The 46 participants were graduate students enrolled in the Software Development Processes course at NDSU. The course covered the breadth of development activities included in the software engineering processes. The participants inspected the same requirements document as in Study 2. The participants were randomly assigned into the experiment and control groups. Two participants dropped out of the study leaving 23 in the control group and 21 in the experimental group. During the second inspection, the participants in the control group were further divided into three groups of 8, 8, and 7 that inspected the requirement document using the PBR User, Designer, and Tester perspectives respectively.

The experiment steps and training lectures in Studies 2 and 4 are shown in Fig. 6. The participants in experiment groups followed the error abstraction and classification process in same way as in Studies 1 and 3. The experiment steps, training lectures, and the output from the participants in the experiment group are the same as detailed in Table 5. The control group inspected the requirement documents twice without using the error abstraction and classification process. The main differences between the design of Studies 2 and 4 are highlighted in Fig. 6 and summarized follows:

In study 2, the control group participants performed two inspections on the same document using the fault checklist. It is likely that these participants were less motivated during the second inspection as compared to the experiment group participants. To address this threat related to maturation, the control group participants in Study 4 used the PBR technique during second inspection instead of the fault checklist technique as used in Study 2;

The experimental and control group participants in Study 2 were from different courses, whereas the experimental and control group participants in Study 4 were from the same course;

No in-class discussion was held with the participants in Study 4 after the study. Only the post-study questionnaire was used to gather feedback;

The feedback from the Study 2 participants indicated that the error information in the taxonomy could have helped them during the error abstraction step. So, in Study 4, Training 2 (on error abstraction) and 3 (on the taxonomy) from Study 2 were combined into a single training session (Training 2). Consequently, the "*error-fault list*"

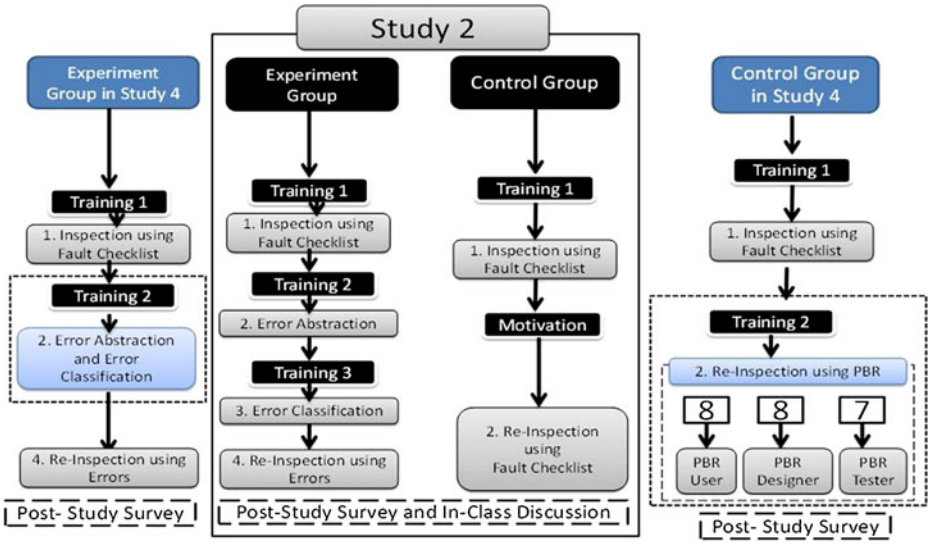

Fig. 6 Experiment design for control-group Studies 2 and 4

and "*error-class list*" (as shown in Table 4) were combined into a single list as an output from Step 2 in Study 4.

5.2 Data Analysis and Results

The data collection process for Studies 2 and 4 was the same as described in Section 4.2 for Studies 1 and 3, without the participant observer. In studies 2 and 4, we used a 5-point scale for qualitative data collection. The results from these studies are organized around the five hypotheses presented in Section 3. An alpha value of .05 was again used for determining statistical significance.

5.2.1 Hypothesis 1: Improved Effectiveness and Efficiency

As opposed to Studies 1 and 3, which were within-group studies, Studies 2 and 4 were between group studies, therefore the analysis is different. We compared the results from the experiment group and control group to determine what proportion of the additional faults found during the reinspection could be attributed to the use of the error abstraction and classification process. Figure 7 shows that there was little difference between the groups during the first inspection. However, during the second inspection, the experimental groups were more effective than the control groups. Using an independent samples *t*-test, this difference was significant ($t_{16}=-4.119$, $p=.002$ for Study 2 and $t_{42}=2.005$, $p=.05$ for Study 4).

Because we conducted four analyses on the same dataset, we used the Bonferroni correction (Bland 2000) to reduce the likelihood of making a Type I error. With the reduced alpha of .013 (.05/4), only the experimental group in Study 2 is significantly more effective than the control group. Also, the experimental group in Study 2 was significantly more effective overall (i.e. first inspection plus second inspection) than the control group

(t_{16}=−2.241, p=0.004). However, the experimental group in Study 4 was not significantly more effective overall than the control group.

We also performed the same analysis relative to efficiency (*faults/time*). For the experimental group, the *time* includes the time spent abstracting and classifying errors and the time spent performing the inspection. For the control group, the time includes only the inspection time. During the reinspection, the average efficiency values for the participants in the experiment group (5.48 faults/hour in Study 2 and 10.28 faults/hour in Study 4) was greater than the average efficiency values for those in the control group (2.27 faults/hour in Study 2 and 6.6 faults/hour in Study 4) in both studies. However, this difference was significant only in Study 4 (t_{44}=-2.96, p<0.015). The higher efficiency rates for the experiment group in Study 4 can be attributable to the fact that participants in Study 2 performed the error abstraction and error classification steps separately whereas participants in Study 4 combined them into a single step. Consequently, the participants in Study 4 combined the error-fault list and "error class list" into a single list.

Similar to Studies 1 and 3, Table 9 provides more details about the min, max and average time spent by the participants in each phase of the study.

Because the experimental group in Study 2 was not significantly more efficient, it is possible that the increased effectiveness by the experimental group was due to extra effort. An ANCOVA test showed that the amount of effort did not have a significant impact on effectiveness.

5.2.2 Hypothesis 2: Usefulness of Requirement Error Taxonomy

As in Study 3, the participants used the same 5-point scale to evaluate V2.0 of the requirement error taxonomy on the same ten characteristics. Table 10 shows the mean of the responses for each characteristic. Similar to Studies 1 and 3, we used a one-sample Wilcoxon Signed Ranks test to see if the ratings were significantly greater than the mean. The shaded cells in Table 10 highlight those results that were significant. The participants rated five characteristics in Study 2 and nine in Study 4 significantly positive. The other characteristics were rated positive but not significantly. Therefore, the results in Table 10 show that the participants viewed the requirement error taxonomy favorably.

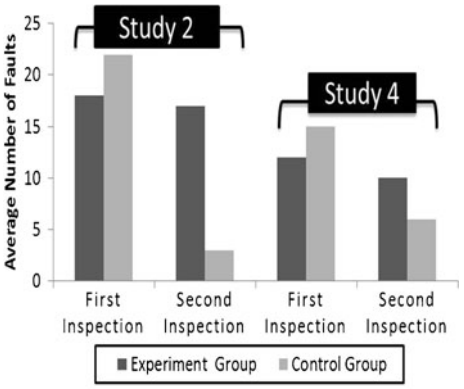

Fig. 7 Comparison of effectiveness

5.2.3 Hypothesis 3: Insights into Major Source(s) of Requirement Faults

We conducted this analysis in the same way as in Studies 1 and 3. The first two rows in Table 11 show the number and the percentage distribution of errors and faults across the error types. *People errors* were the most common in both studies, while *documentation errors* actually led to more faults in Study 2. We used the chi-square test to evaluate whether each distribution is significantly different from uniform. The result shows that the *People* and *Documentation* error types made significantly different contributions to fault injection, as highlighted in Table 11.

Table 11 also shows the p-value from the chi-square test regarding the contribution of the *People*, *Process*, and *Documentation* errors to the redundant, time-consuming, multiple, and redundant faults. Similar to the results from earlier studies, for each variable, *People Errors* caused significantly more faults than *Process* or *Documentation* errors. Even though documentation errors led to more faults in Study 2, those faults tended to be less important and severe, according to the participants' rankings.

5.2.4 Hypothesis 4: Usefulness of Cognitive Psychology Research

In Study 2, 25% of the errors and 21% of the faults were related to cognitive psychology. Similarly, in Study 4, 29% of the errors and 28% of the faults were related to cognitive psychology. Only one participant in Study 4 did not find any cognitive psychology related errors.

Figure 8 shows the breakdown of the faults found during each inspection that were attributed to the errors derived from cognitive psychology research. The result shows that these faults were spread fairly evenly across all participants. Also, the results show that a larger percentage of the faults found during the second inspection were due to cognitive psychology errors compared with those from the first inspection. However, there are certain cases, in which the participants (i.e., in Study 4) did not find any faults during the second inspection that could be attributed to the cognitive psychology errors. Overall, this result indicates that cognitive psychology research made a useful contribution to the error taxonomy and without these error types, the inspectors would have missed some faults.

5.2.5 Hypothesis 5: Effect of Independent Variables

We analyzed the same five independent variables that were analyzed in Studies 1 and 3 using a multiple regression. *Process Conformance* had a significant positive correlation to effectiveness during reinspection in Study 2 only. *Pre-test* and *Effort Expended* had a significant positive correlation to each other and to the participants' effectiveness in both studies. This result means that participants who understood the taxonomy and invested an appropriate amount of effort were more effective than those that did not.

Table 9 Time Spent in 1st Inspection, error abstraction/classification and 2nd inspection

	1st Inspection Time Min/Avg/Max (in minutes)	Error Abst/Class Time Min/Avg/Max (in minutes)	2nd Inspection Time Min/Avg/Max (in minutes)
S2 - Exp	93/157/232	25/37/51	29/82/134
S2 – Control	132/190/265	*Did not perform*	95/210/271
S4 – Exp	35/113/179	17/69/120	12/61/140
S4 – Control	82/169/210	*Did not perform*	27/151/240

Table 10 Study 2 and Study 4: Mean value of the ratings of the requirement error taxonomy on different characteristics

	Middle point	1. simp	2. usa	3. orth	4. use	5. und	6. Int	7. comp	8. aesp	9. aec	10. ece
Study 2	Medium-"3"	3.13	3.25	3.38	4.38	4	3.5	4.28	3.34	4.4	3.13
Study 4	Medium-"3"	3.1	4.16	3.8	4.25	4.38	4.08	4.5	4.25	4.18	4.38

5.2.6 Validity Threats

We faced the following threats to validity in one or both of these studies.

Conclusion Validity Similar to Studies 1 and 3, we have treated 5-point scales as interval data rather than ordinal data. To reduce construct validity problems, we used nonparametric tests, where appropriate, to minimize data assumptions. However, there still remains a threat that we treated the 5-point scale as an interval scale, following the standard practice in the social sciences. This practice means that the p-value needs to be treated with care when interpreting the results.

Internal Validity To increase internal validity, we did not inform the participants of the study goals. Therefore they should not have been biased in the data provided. Participants were graded based on their participation in the study rather than the actual information they provided. Conversely, a selection threat exists in Study 2. We allocated the participants to groups based on the course in which they were enrolled. To reduce this threat, we chose the course (V&V) whose students were likely to

Table 11 Study 2 and Study 4: Insights provided by the requirement error taxonomy

Variable	Study #	N	People Errors	Process Errors	Documentation Errors	p-value
Total Errors	Study 2	114	41%	25%	34%	.145
	Study 4	125	51%	24%	25%	**< .001**
Total Faults	Study 2	168	35%	19%	46%	**.004**
	Study 4	184	55%	25%	20%	**< .001**
Redundant Faults	Study 2	78	52%	18%	30%	**.004**
	Study 4	94	49%	22%	29%	**.005**
Time-consuming Faults	Study 2	29	60%	13%	27%	**.004**
	Study 4	17	51%	17%	32%	**.002**
Multiple Faults	Study 2	45	44%	30%	26%	.068
	Study 4	38	57%	24%	19%	**< .001**
Important Faults	Study 2	87	62%	22%	16%	**< .001**
	Study 4	62	55%	33%	12%	**< .001**
Severe Faults	Study 2	103	53%	28%	19%	**< .001**
	Study 4	54	56%	32%	12%	**< .001**

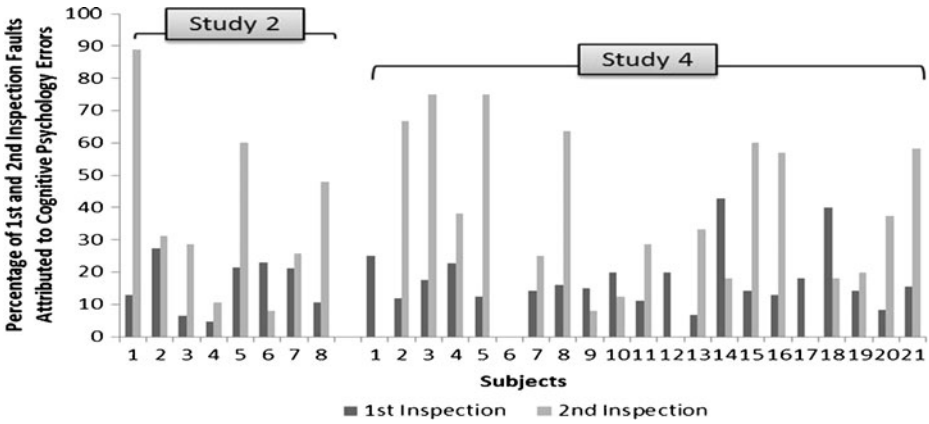

Fig. 8 Contribution of cognitive psychology research errors to the number of faults found during first and second inspections of Study 2 and Study 4

perform better inspections (because of the course material) to be the control group. This choice was made so that if there was any bias, it would be towards the control group and not the experimental group. This bias was eliminated in Study 4 by randomly assigning participants to groups.

External Validity To increase external validity, the participants in both studies inspected a real requirements document developed by professional developers at the National Oceanographic Office. Conversely, there remains an external threat to validity because all of the participants in both studies were graduate students rather than professional developers.

6 Discussion of Results

To bring the results from all four studies together, this section discusses each hypothesis in light of the results of all four studies. We draw conclusions across all four studies rather than from each individual study, (e.g., (Walia 2006a; Walia et al. 2007)). In some cases the general conclusions are the same as those drawn from a single study, but stronger because of common results in all four studies.

6.1 Hypothesis 1

The error abstraction and classification approach improves the effectiveness (number of faults) and efficiency (faults per hour) of inspection teams and individual inspectors

For team effectiveness, all four studies showed that the *error abstraction and classification* process was beneficial. In Studies 1 and 3, each team found a significant number of new faults when using the error abstraction and classification process (75% and 156% in Study 1 and 81% and 84% in Study 3). In Studies 2 and 4, the participants who used the error abstraction and classification process were significantly more effective during reinspection than the control group, even when the control group used a proven fault detection technique (PBR). In terms of individual effectiveness, the participants found a significant number of faults during the reinspection.

The results from all four studies showed that the error abstraction and classification process did not significantly hurt efficiency. This result is positive because there are extra steps required by the error abstraction and classification process. Based on a suggestion from the participants in Study 2, the participants in Study 4 directly used the requirement error taxonomy to abstract errors and reinspect rather than performing them as two separate steps. This change resulted in improved efficiency.

Combining the results of effectiveness and efficiency, we can conclude that the error abstraction and classification process significantly improved the effectiveness while not hurting the efficiency of inspectors compared with the fault checklist and PBR inspection method. Based on these results, future work will investigate ways to improve efficiency.

Also, contrary to the Studies 1 and 3, the participants in Study 2 and Study 4 did not meet with other participants in the experiment group to discuss their individual errors and faults when using the error abstraction and classification process. The results from Studies 1 and 3 showed that the discussion of the individual errors detected by participants helped the teams to identify additional problems that were not detected during the individual inspections. Therefore, having a team meeting fosters the discussion of mistakes that were made during the development of requirements document and helps identify a larger number of requirement problems.

6.2 Hypothesis 2

The requirement error taxonomy is useful for helping inspectors find errors and faults

The data to evaluate this hypothesis came from the subjective ratings by the participants. In Study 1, the participants evaluated V1.0 of the error taxonomy using a 3-point scale whereas in Studies 2, 3 and 4, they evaluated V2.0 of the error taxonomy using a 5-point scale.

Table 12 summarizes the results. The shaded cells represent a significant result. The participants rated seven out of ten characteristics positively in at least three studies and three characteristics positively in all four studies. They did not rate any characteristics negatively in any study.

Post-study interviews in Studies 1, 3, 4 and in-classroom discussion in Study 2 provided some additional insights into these results. First, based on the discussions we can pose a new hypothesis. ***H6: The error abstraction and classification process is easier to use when the inspectors participate in the development of the requirements***. This hypothesis will be tested in future studies. Second, participants from all four studies agreed that using error information helped them better understand the real problems in the requirements document and that the effort spent during the error abstraction and classification process was extremely

Table 12 Results from evaluation of the requirement error taxonomy

Study #	1. sim	2. usa	3. orth	4. use	5. und	6. int	7. comp	8. aesp	9. aec	10. ece
1	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
2	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
3	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
4	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑

worthwhile. Finally, the participants agreed that the errors described by the requirement error taxonomy were a true representation of the mistakes that can occur during software development. They were in favor of the creation of similar error taxonomies for other lifecycle stages (i.e. architecture/design, coding).

6.3 Hypothesis 3

The requirement error taxonomy provides important insights into the major source(s) of requirement faults

One of the insights provided by the requirement error taxonomy was the relative contributions of the three error types. In Studies 1, 3 and 4, there were significantly more *People Errors* and significantly more faults related to those errors. Conversely, in Study 2, there were more *People Errors*, but the *Documentation Errors* actually lead to significantly more faults. To explain this difference, we pose another hypothesis. ***H7: Participation in the development of the requirements affects the types of errors and faults identified during an inspection.*** In Studies 1 and 3 where *People Errors* led to largest number of faults, the participants developed the requirements. Conversely, in Study 2, where *Documentation Errors* led to the largest number of faults, the participants did not develop the requirements. The results from study 4 (i.e., participants did not develop the requirements but still found largest number of people errors) do not fit this pattern. So, this hypothesis needs further investigation.

Furthermore, the result showed that, in general, across all the studies a significantly higher number of multiple, redundant, important and severe faults were the result of *People Errors*. Further study of the 14 detailed error classes showed that at least one error from each class was found by participants in each study. This result provides confidence that the error classes are valid and provide a good coverage of the requirement error space.

Finally, the insights from the participant observation in Study 3 showed that:

- The participants did not find all of the errors observed during requirement development;
- During the team meetings that occurred after the error abstraction and classification process the participants found more errors and faults than during the meetings that occurred before the error abstraction and classification process;
- The participants better understood the requirements problems after becoming aware of the errors committed during development of the requirements document, and subsequently found more faults; and
- The participants followed the error abstraction and classification process closely and communicated well with other participants during the inspection team meetings.

6.4 Hypothesis 4

The contributions from the human cognition and psychology fields help inspectors locate more faults

In all four studies, a large percentage of the faults and errors (21%–32%) were related to human cognition. Furthermore, most participants found errors and faults that were related to human cognition. These results support the importance of using human cognition research to better understand the source of faults and to improve the effectiveness of software inspectors. Therefore, applying cognitive psychology research to software engineering helps improve software quality.

6.5 Hypothesis 5

Other independent variables (process conformance, performance on a pre-test, usefulness of training, effort, and perceived difficulty) affect the individual performance during the error abstraction and classification process

The correlation between the independent and dependent variables differs across the four studies. The major conclusions about the impact of the independent variables on the performance of the inspectors are:

- A good understanding of the error taxonomy, evidenced by pretest score, combined with an appropriate amount of effort expended during error abstraction and classification is likely to lead to increased effectiveness;
- Process conformance improves effectiveness;
- Effective training on error abstraction and on the use of the requirement error taxonomy for reinspection is necessary for improving effectiveness; and
- An increase in the amount of effort spent during the error abstraction and classification can also increase the fault detection efficiency

6.6 Validity Threats across All Four Studies

By varying the design among the four studies, we addressed many of the internal validity threats. The main unaddressed threat is that all four studies were conducted in a university setting. The participants, undergraduate students in Studies 1 and 3 and graduate students in Studies 2 and 4, are likely not representative of professionals in a real industrial setting. This threat was minimized in Studies 1 and 3 by having the students work with real clients to develop requirement documents that they later used to implement systems. Also, in Studies 2 and 4 participants inspected a real software requirement document developed by external software professionals. To address this threat, we plan to perform a replication in a software company in the near future.

7 Conclusions and Future Work

Table 13 summarizes the results described in Section 6. Based on these results, we conclude that addressing software quality by focusing on errors is more effective than

Table 13 Overall results from the evaluation of the requirement error taxonomy

Hypothesis	Results from all Four Studies
H1	The participants using the error abstraction and classification were significantly more effective during reinspection and overall than the control group, with no significant difference in efficiency.
H2	The requirement error taxonomy was viewed significantly favorably relative to usefulness, understandability, comprehensiveness, ease of classifying errors, uniformity across different products, and containing adequate requirement error classes.
H3	*People errors* led to a significantly larger percentage of redundant faults, time-consuming faults, multiple faults, important faults and severe faults.
H4	On average, one-fourth of the faults were caused by errors related to cognitive psychology.
H5	Performance on a pre-test, overall process conformance, and the overall effort spent were significant predictors of effectiveness during the reinspection.

addressing software quality by focusing only on faults. We also conclude that *Cognitive Psychology* research provides a valuable contribution to software inspection methods. This research reports some useful insights into the cognitive processes employed by software engineers and where these processes are likely to fail. The results show that both the *error abstraction and classification process* and the *requirement error taxonomy* are beneficial to developers.

We have identified some needed improvements in the requirement error taxonomy and the error abstraction and classification process based on feedback from the participants. One major improvement is to focus on the efficiency of using the requirement error taxonomy.

This research investigated the use of the requirement error taxonomy in controlled settings. Our next immediate step is to evaluate the requirement error taxonomy in an industrial setting with professional software developers. Also, our future work is to develop inspection techniques, like PBR techniques, based on the requirement error taxonomy. We plan to perform additional empirical studies to understand the different kinds of errors that are made by developers with different backgrounds and experiences. Finally, we identified two new hypotheses that we will investigate in future studies.

Acknowledgements We thank the study participants. We also thank Dr. Thomas Philip for providing access to his courses. We acknowledge the *Empirical Software Engineering* groups at MSU and NDSU for providing useful feedback on the study designs and data analysis. We thank Dr. Gary Bradshaw for his expertise on cognitive psychology. We thank Dr. Edward Allen and Dr. Guilherme Travassos for reviewing early drafts of this paper. We also thank the reviewers for their helpful comments.

Appendix A

This appendix describes the different errors in each of the fourteen detailed error classes (described in Table 1). A complete description of the requirement error taxonomy (along with examples of errors and faults) has been published in a systematic literature review (Walia and Carver 2009). Tables 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 show each error class along with the specific errors that make up that error class.

Table 14 Communication errors

Inadequate project communications
Changes in requirements not communicated
Communication problems, lack of communication among developers and between developers and users
Poor communication between users and developers, and between members of the development team
Lack of communication between sub teams
Communication between development teams
Lack of user communication
Unclear lines of communication and authority
Poor communication among developers involved in the development process
Communication problems, information not passed between individuals
Communication errors within a team or between teams
Lack of communication of changes made to the requirements
Lack of communication among groups of people working together

Table 15 Participation errors

No involvement of all the stakeholders
Lack of involvement of users at all times during requirement development
Involving only selected users to define requirements due to the internal factors like rivalry among developers or lack of the motivation
Lack of mechanism to involve all the users and developers together to resolve the conflicting requirements needs

Table 16 Domain knowledge errors

Lack of domain knowledge or lack of system knowledge
Complexity of the problem domain
Lack of appropriate knowledge about the application
Complexity of the task leading to misunderstandings
Lack of adequate training or experience of the requirement engineer
Lack of knowledge, skills, or experience to perform a task
Some properties of the problem space are not fully investigated
Mistaken assumptions about the problem space

Table 17 Specific application errors

Lack of understanding of the particular aspects of the problem domain
Misunderstandings of hardware and software interface specification
Misunderstanding of the software interfaces with the rest of the system
User needs are not well understood or interpreted while resolving conflicting requirements
Mistakes in expression of the end state or output expected
Misunderstandings about the timing constraints, data dependency constraints, and event constraints
Misunderstandings among input, output and process mappings

Table 18 Process execution errors

Mistakes in executing the action sequence or the requirement engineering process, regardless of its adequacy
Execution or storage errors, out of order sequence of steps and slips/lapses on the part of people executing the process

Table 19 Other human cognition errors

Mistakes caused by adverse mental states, loss of situation awareness
Mistakes caused by ergonomics or environmental conditions
Constraints on humans as information processors e.g., task saturation

Table 20 Inadequate method of achieving goals and objectives

Incomplete knowledge leading to poor plan on achieving goals
Mistakes in setting goals
Error in choosing the wrong method or wrong action to achieve goals
Some system-specific information was misunderstood leading to the selection of wrong method
Selection of a method that was successful on other projects
Inadequate setting of goals and objectives
Error in selecting a choice of a solution
Using an analogy to derive a sequence of actions from other similar situations resulting in the wrong choice of a sequence of actions
Transcription error, the developer understood everything but simply made a mistake

Table 21 Management errors

Poor management of people and resources
Lack of management leadership and necessary motivation
Problems in assignment of resources to different tasks

Table 22 Requirement elicitation errors

Inadequate requirement gathering process
Only relying on selected users to accurately define all the requirements
Lack of awareness of all the sources of requirements
Lack of proper methods for collecting requirements

Table 23 Requirement analysis errors

Incorrect model(s) while trying to construct and analyze solution
Mistakes in developing models for analyzing requirements
Problem while analyzing the individual pieces of the solution space
Misunderstanding of the feasibility and risks associated with requirements
Misuse or misunderstanding of problem solution processes
Unresolved issues and unanticipated dependencies in solution space
Inability to consider all cases to document exact behavior of the system
Mistakes while analyzing requirement use cases or scenarios

Table 24 Requirement traceability errors

Inadequate/poor requirement traceability
Inadequate change management, including impact analysis of changing requirements

Table 25 Requirement organization errors

Poor organization of requirements
Lapses in organizing requirements
Ineffective method for organizing together the requirements documented by different developers

Table 26 No use of standard for documenting errors

No use of standard format for documenting requirements
Different technical standard or notations used by sub teams for documenting requirements

Table 27 Specification errors

Missing checks (item exists but forgotten)
Carelessness while organizing or documenting requirement regardless of the effectiveness of the method used
Human nature (mistakes or omissions) while documenting requirements
Omission of necessary verification checks or repetition of verification checks during the specification

In addition, the following are representative examples of the faults discovered in the studies that were attributed to different error types in the requirement error taxonomy:

- Missing functionality for viewing reservations in the Starkville Theatre System project (Study 1). This fault was attributed to the *Specific Application Knowledge* error (Table 17).
- The wrong placement of a precondition in the use case. This fault was attributed to the *Process Execution* error (Table 18).
- Inconsistency in the interface description with other sections of the requirement document. This fault was attributed to the *Communication* error among team members (Table 14).
- Main success scenario in a use case is vague. This fault was attributed to the *Domain Knowledge* error (Table 16).
- Inconsistent information about indexing particular user functionality. This fault was attributed to the *Human Cognition* error (Table 19).
- Extraneous functional requirement in Data Warehouse requirements document. This fault was attributed to the *Traceability* process error (Table 24).
- Missing information regarding security requirements. This fault was attributed to the *Requirement Elicitation* process error (Table 22).
- Functionality listed in wrong section. This fault was attributed to the *Requirement Organization* error (Table 25).

References

Basili VR, Green S, Laitenberger O, Lanubile F, Shull F, Sørumgård S, Zelkowitz MV (1996) The empirical investigation of perspective-based reading. Empir Software Eng: An International Journal 1(2):133–164

Basili VR, Shull F, Lanubile F (July 1999) Building knowledge through families of experiments. IEEE Trans Software Eng 25(4):456–473

Bland M (2000) An introduction to medical statistics, Chapter-9, 3rd edn. Oxford, University Press Inc, New York. ISBN 0192632698
Boehm B, Basili VR (2001) Software defect reduction top 10 List. Computer 34(1):135–137
Card DN (1998) Learning from our mistakes with defect causal analysis. IEEE Softw 15(1):56–63
Card SK, Moran TP, Newell A (1983) The psychology of human-computer interaction. Erlbaum, Hillsdale
Carver J (2003) The impact of background and experience on software inspections, PhD Thesis. Department of Computer Science, University of Maryland College Park, Maryland
Chaar JK, Halliday MJ, Bhandari IS, Chillarege R (1993) In-process evaluation for software inspection and test. IEEE Trans Software Eng 19(11):1055–1070
Chillarege R, Bhandari IS, Chaar JK, Halliday MJ, Moebus DS, Ray BK, Wong MY (1992) Orthogonal defect classification-a concept for in-process measurements. IEEE Trans Software Eng 18(11):943–956
Endres A, Rombach D (2003) A handbook of software and systems engineering, 1st edn. Pearson Addison Wesley, Harlow
Field A (2007) Discovering statistics using SPSS, 2nd edn. SAGE Publications Ltd, London
Florac W (1992) Software quality measurement: a framework for counting problems and defects. Technical Reports, CMU/SEI-92-TR-22. Software Engineering Institute
Grady RB (1996) Software failure analysis for high-return process improvement. Hewlett-Packard J 47(4):15–24
IEEE Std 610.12-1990 (1990) IEEE standard glossary of software engineering terminology
Jacobs J, Moll JV, Krause P, Kusters R, Trienekens J, Brombacher A (2005) Exploring defect causes in products developed by virtual teams. J Inform Software Tech 47(6):399–410
Kan SH, Basili VR, Shapiro LN (1994) Software quality: an overview from the perspective of total quality management. IBM Syst J 33(1):4–19
Kitchenham B (2004) Procedures for Performing Systematic Reviews. TR/SE-0401. Department of Computer Science, Keele University and National ICT, Australia Ltd. http://www.elsevier.com/framework_products/promis_misc/inf-systrev.pdf
Lanubile F, Shull F, Basili VR (1998) Experimenting with error abstraction in requirements documents. In Proceedings of Fifth International Software Metrics Symposium, METRICS98 pp 114–121
Lawrence CP, Kosuke I (2004) Design error classification and knowledge. J Knowl Manag Pract (May)
Lezak M, Perry D, Stoll D (2000) A case study in root cause defect analysis. In Proceedings of the 22nd International Conference on Software Engineering. Limerick, Ireland. pp 428–437
Masuck C (2005) Incorporating a fault categorization and analysis process in the software build cycle. J Comput Sci Colleges 20(5):239–248
Mays RG, Jones CL, Holloway GJ, Studinski DP (1990) Experiences with defect prevention. IBM Syst J 29 (1):4–32
Nakashima T, Oyama M, Hisada H, Ishii N (1999) Analysis of software bug causes and its prevention. J Inform Software Tech 41(15):1059–1068
Norman DA (1981) Categorization of action slips. Psychol Rev 88:1–15
Pfleeger SL, Atlee JM (2006) Software engineering theory and practice, 3rd edn. Prentice Hall, Upper Saddle River
Rasmussen J (1982) Human errors: a taxonomy for describing human malfunction in industrial installations. J Occup Accid 4:311–335
Rasmussen, J., "Skills, Rules, Knowledge: Signals, Signs and Symbols and Other Distinctions in Human Performance Models." *IEEE Transactions: Systems, Man, & Cybernetics*, 1983. SMC-13: 257-267.
Reason J (1990) Human error. Cambridge University Press, New York
Sakthivel S (1991) A survey of requirements verification techniques. J Inf Technol 6:68–79
Seaman CB (1999) Qualitative methods in empirical studies of software engineering. IEEE Trans Softw Eng 25(4):557–572
Seaman CB, Basili VR (1997) An empirical study of communication in code inspections. Proceedings of International Conference in Software Engineering, pp 96–106, Boston, Mass. May
Sommerville I (2007) Software engineering, 8th edn. Addison Wesley, Harlow
Walia GS (2006a) Empirical validaton of requirement error abstraction and classification: a Multidisciplinary Approach, M.S Thesis, Comput Sci Eng, Mississippi, Starkville
Walia GS, Carver J (2009) A systematic literature review to identify and classify requirement errors. J Inform Software Tech 51(7):1087–1109
Walia GS, Carver J, Philip T (2006b) Requirement Error Abstraction and Classification: An Empirical Study. In Proceedings of IEEE Symposium on Empirical Software Engineering. ACM Press, Brazil pp 336–345
Walia G, Carver J, Philip T (2007) Requirement error abstraction and classification: a control group replicated study, in 18th IEEE International Symposium on Software Reliability Engineering. Trollhättan, Sweden

Gursimran S. walia received his PhD degree in Computer Science from Mississippi State University. He is an assistant professor in the Department of Computer Science at North Dakota State University. His main research interests include empirical software engineering, requirements engineering, human factors in software engineering, software inspections and software errors. He is a member of the IEEE Computer Society. Contact him at gursimran.walia@ndsu.edu.

Jeffrey C. Carver earned his PhD degree in Computer Science from the University of Maryland. He is an assistant professor in the Department of Computer Science at the University of Alabama. His main research interests include empirical software engineering, software quality, software engineering for computational science and engineering, software architecture, human factors in software engineering and software process improvement. He is a Senior Member of the IEEE Computer Society and a Senior Member of the ACM. Contact him at carver@cs.ua.edu.