Empir Software Eng (2018) 23:3768–3800
https://doi.org/10.1007/s10664-018-9623-8

Using human error information for error prevention

Wenhua Hu[1] · Jeffrey C. Carver[2] · Vaibhav Anu[3] · Gursimran S. Walia[3] · Gary L. Bradshaw[4]

Published online: 23 May 2018

Abstract Developing error-free software requirements is of critical importance to the success of a software project. Problems that occur during requirements collection and specification, if not fixed early, are costly to fix later. Therefore, it is important to develop techniques that help requirements engineers detect and prevent requirements problems. As a human-centric activity, requirements engineering can be influenced by psychological research about human errors, which are the failings of human cognition during the process of planning and executinge a task. We have employed *human error* research to describe the types of problems that occur during requirements engineering. The goals of this research are: (1) *to evaluate whether understanding human errors contributes to the prevention of errors and concomitant faults during requirements engineering* and (2) *to*

Communicated by: Paul Grünbacher and Anna Perini

✉ Jeffrey C. Carver
carver@cs.ua.edu

Wenhua Hu
whu4@kennesaw.edu

Vaibhav Anu
vaibhavanu.x@gmail.com

Gursimran S. Walia
gursimran.walia@ndsu.edu

Gary L. Bradshaw
glb2@psychology.msstate.edu

[1] Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA, USA

[2] Department of Computer Science, University of Alabama, Tuscaloosa, AL, USA

[3] Department of Computer Science, North Dakota State University, Fargo, ND, USA

[4] Department of Psychology, Mississippi State University, Starkville, MS, USA

identify error prevention techniques used in industrial practice. We conducted a controlled classroom experiment to evaluate the benefits that knowledge of errors has on error prevention. We then analyzed data from two industrial surveys to identify specific prevention and mitigation approaches employed in practice. The classroom study showed that the better a requirements engineer understands human errors, the fewer errors and concomitant faults that engineer makes when developing a new requirements document. Furthermore, different types of Human Errors have different impacts on fault prevention. The industry study results identified prevention and mitigation mechanisms for each error type. Human error information is useful for fault prevention during requirements engineering. There are practices that requirements engineers can employ to prevent or mitigate specific human errors.

Keywords Human errors · Software requirements · Error prevention · Fault prevention · Empirical study · Human factors

1 Introduction

The quality and correctness of software requirements directly impacts the quality and correctness of the final software product. Correct requirements are important because requirements problems: (1) are expensive to fix in later phases of software development (Dethomas and Feo 1987), (2) rank among the most severe kinds of problems (Chen and Huang 2009), and (3) lead to the majority of software failures (Hamill and Goseva-Popstojanova 2009). Rather than finding and fixing these problems after they occur, which can be expensive, requirements engineers need methods to help them prevent problems from occurring in the first place.

Human error research can help address this prevention problem. Cognitive psychologists have long studied failures of human cognition, including human errors, and their effects on critical human-centric tasks. As Section 2.1 explains, human errors occur during the process of developing or executing a plan to solve a problem or complete a task. Human error research has focused on providing insight into how human mental processes can fail in various situations. The primary focus of human error research is identifying types of human errors and developing mechanisms to prevent these errors from occurring in the future. Human error research has supported error prevention in other fields, e.g. medicine and aviation (Wiegmann et al. 2005; Shappell and Wiegmann 2001; Diller et al. 2013; Zhang et al. 2004; Reason 1990; Reason et al. 1990). Examples of error-based prevention techniques include changes to instrumentation and processes (Wickens et al. 2015) and crew cockpit resource management (Wiener et al. 1995). Prevention techniques can be broadly classified into four strategies: Reduce complexity, optimize information processing, automate wisely, and use constraints (Nolan 2000).

Because requirements engineering is a complex, human-centric activity, the fallibility of human cognition during requirements elicitation, analysis, and documentation leads to human errors. These human errors can then lead to various types of requirements faults. In the same way that human error research has benefited other fields by providing error prevention mechanisms, we hypothesize that properly applied human error research can have the same preventative effects in requirements engineering. Therefore, the goals of this research are: (1) *to evaluate whether understanding human errors contributes to the prevention of errors and concomitant faults during requirements engineering* and (2) *to identify error prevention techniques used in industrial practice.*

To address these research objectives, we conducted two types of studies. Relative to our first goal, we hypothesize that the better a requirements engineer understands the types of human errors that can occur during requirements engineering the fewer of those errors that engineer will make. To study this hypothesis, we conducted a classroom study, initially published at REFSQ‘17 (Hu et al. 2017). We trained requirements engineers on human errors by first teaching them human error concepts and then giving them a requirements document to inspect for related problems. After the training, the participants created their own requirements document that we could use to measure whether there was any relationship between the level of understanding human errors (from the training) and the frequency of making those errors when developing requirements.

To build upon the results of the first study and address the second goal, we analyzed data from two surveys of industrial professionals (one conducted by us and one conducted by collaborators). In these surveys, the respondents provided information about the types of requirements problems faced in their organizations and described mechanisms employed to prevent or mitigate these problems. We analyzed this data to develop a list of approaches for preventing and mitigating human error problems in requirements engineering. The results of the survey analysis is a new contribution of this paper beyond our REFSQ’17 paper.

The primary contributions of this paper are (1) an evaluation of whether knowledge of human errors prevents requirements engineers from making errors and associated faults in requirements documents, and (2) a description of industrial error prevention and mitigation mechanisms.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 describes the classroom study that evaluates the utility of error information in preventing errors and faults. Section 4 describes the analysis and results of the industrial surveys. Section 5 concludes the paper and enumerates ideas for future studies.

2 Background

This section provides background information on topics relevant to this paper.

2.1 Errors, Faults, and Failures

As we are studying software quality, the terms *error*, *fault*, and *failure* are important for understanding our work. Because these terms have competing and contradictory definitions in the literature, we provide definitions here that we use throughout the paper. These definitions are consistent with previous work (Lanubile et al. 1998), an IEEE Standard (IEEE 2017), and a software engineering textbook (Sommerville 2010), but are tailored for consistency with requirements engineering.

Error	– *mistakes in the human thought process while trying to understand information, devise a planned solution, or execute a planned solution.* Relative to IEEE Standard 24765 (IEEE 2017), we focus on human errors rather than program errors. An error may lead to 0 or more faults.
Fault	– *manifestation of an error recorded in a document (e.g. requirements specification or use case).* A fault may lead to 0 or more failures.
Failure	– *the actual software behavior differs from the expected behavior (e.g. incorrect output or system crash).*

2.2 Fault Prevention Techniques

Fault prevention techniques use knowledge of likely problems (often using historical data, a sample of faults, or expert opinion) to prevent those problems from occurring in the future. Based on the fault injection rate, fault prevention techniques provide specific strategies to prevent related faults (Graham 2005). Furthermore, training and mentoring can also result in dramatic reductions of fault rates (close to 50%) (Card 1998). Existing fault prevention techniques include:

– **Fault Causal Analysis** – uses a sample of faults to determine their causes and prevent future faults. An empirical study at IBM found that 'people-related factors' were the most common causes of software faults (Card 1998);
– **Software Failure Analysis** – improves the development and maintenance processes by analyzing a representative sample of faults to understand the causes of particular classes of faults. Engineers at Hewlett-Packard used this approach to understand the cause of user-interface faults, but needed expert knowledge to perform an in-depth, per-object fault analysis (Grady 1996);
– **Fault Prevention Process** – determines the source of a fault and suggests preventive actions by classifying fault causes as *oversight* (e.g., requirements engineer overlooked something, or something was not considered thoroughly), *education* (requirements engineer did not understand some aspect), or *transcription* (requirements engineer knew what to do but simply made a mistake) (Mays et al. 1990);
– **Root Cause Analysis** – uses a multi-dimensional fault trigger concept to help requirements engineers determine the root cause of a fault and to identify process improvements (Leszak et al. 2000).

Similar to these approaches, researchers have proposed variations of fault prevention techniques, including: the fault distribution method (Pooley et al. 2002), the defect-based process improvement (DBPI) technique (Kumaresh and Baskaran 2012), and the defect prevention-based process improvement (DPPI) method (Terzakis 2013). However, these techniques do not provide guidelines to help requirements engineers learn from the errors over time.

Fault prevention is part of a bigger discussion on fault avoidance. The aim of fault avoidance is to ensure the production of fault-free software through several approaches that include but are not limited to: formal specifications, software testing, and verification and validation (V&V) (Saha 2006). The goal of formal specifications is to reduce errors during the early stages of software development (i.e., requirements specification and design) (Saha 2006). The aim of software testing is to identify problems missed during early stages or introduced during programming (Suma and Nair 2008). Requirement review techniques (a V&V approach) focus inspectors' attention on requirements defects like ambiguities, inconsistencies, and incorrectness (Raja 2009). Researchers have developed specialized taxonomies to focus inspectors' attention on specific problem types. For example, Kamsties et al. (2001) developed and evaluated a comprehensive taxonomy of ambiguities in requirements (Kamsties et al. 2001). Femmer et al. (2014) proposed automated defect detection techniques that use requirements smells, natural language problems that indicate quality issues in requirements specifications.

While these methods use some representative faults/problem reports to analyze the root cause, they lack an underlying cognitive theory of how and why people make errors. Therefore, there is a need for additional work to formally apply human error research to requirements engineering and build theoretically-sound taxonomies of requirements errors that can support prevention activities.

2.3 Human Error Research

Most psychologists reject the view that errors are idiosyncratic and strongly dependent upon context. James Reason, a noted human error authority proposed that, rather than being random, the occurrence of errors takes recurrent and predictable forms (Reason 2008). This predictability is crucial to enable requirements engineers to use errors from prior projects to anticipate errors that may occur in future projects. By anticipating the errors that may occur, requirements engineers can either prevent the errors or use knowledge of the errors to find the resulting faults.

Central to Reason's comprehensive psychological accounts of human errors (Reason 1990) is the construction of a *plan* (i.e., a series of actions that lead to a goal). Given a plan, people can make three distinct types of human errors. *Mistakes* result in a faulty or inadequate plan. *Slips* and *lapses* are errors of execution in which someone fails to properly carry out a correctly planned step either because of inattention or because of memory failures.

Reason's theory has been extensively adapted to map the causes of accidents to failures of human cognition in several domains. For example, Reason's taxonomy has found diverse application in medical errors (Zhang et al. 2004), problems with Apollo 11 (Reason 1990), driver errors (Reason et al. 1990), and errors that led to the Chernobyl nuclear power plant meltdown (Reason 2008).

2.4 Error Taxonomies

To make human error information useful, researchers classify the errors into taxonomies, logical, hierarchical, organizations of the error information. Without such organization, requirements engineers will find it difficult to successfully employ error information to improve the quality of their work. Previous work that exploited error information to improve software quality (Chillarege et al. 1992; Lanubile et al. 1998; Leszak et al. 2000) provided requirements engineers with methods for using the sources of faults (i.e. errors) to search for evidence of repeated errors and correct them, leading to quality improvements.

While these approaches provide a significant step forward, the lack of a formal error taxonomy to organize and preserve error information means that errors discovered in one project can not be of benefit to future projects. Each project has to start from scratch and cannot capitalize on past error-mitigation efforts. To build upon these methods, we developed taxonomies of human errors in requirements engineering. The remainder of this section explains the approach for development of each taxonomy in more detail.

2.4.1 Requirement Error Taxonomy (RET)

To develop the first taxonomy, we performed a systematic literature review to identify and classify the requirements errors described in the software engineering and cognitive psychology literatures. We developed this taxonomy strictly from the software engineering perspective without input from a human error expert. Therefore, we built it *ad-hoc* (without input from human error theories), based primarily on the errors identified in the literature, without using a formal human error theory as a driver.

The RET (Table 1) includes 14 detailed error classes grouped into three high-level error types: *People Errors* (arise from fallibilities of the people involved in the development process), *Process Errors* (arise while selecting the appropriate processes for achieving the desired goals and relate mostly to the inadequacy of the requirements engineering process), and *Documentation Errors* (arise from mistakes in organizing and specifying the

Table 1 Requirement Error Taxonomy (RET)

High-level Error Type	Detailed Error Type	Error Description
People	Communication Errors	Result from poor or missing communications among the various stakeholders involved in developing the requirements.
	Participation Errors	Result from inadequate or missing participation of important stakeholders involved in developing the requirements.
	Domain Knowledge Errors	Occur when requirement authors lack knowledge or experience about the problem domain.
	Specific Application Errors	Occur when requirement authors lack knowledge or experience about the problem domain.
	Process Execution Errors	Occur when requirement authors make mistakes while executing the requirement elicitation and development processes regardless of the adequacy of the chosen process.
	Other Human Cognition Errors	Result from constraints on the cognitive abilities of the requirement authors.
Process	Inadequate Method of Achieving Goals and Objectives Errors	Result from selecting inadequate or incorrect methods for achieving the stated goals and objectives.
	Management Errors	Result from inadequate or poor management processes.
	Requirement Elicitation Errors	Result from the use of an inadequate requirement elicitation process.
	Requirement Analysis Errors	Commit during the requirement analysis process.
	Requirement Traceability Errors	Result from an inadequate or incomplete requirement traceability process.
Documentation	Requirement Organization Errors	Commit while organizing the requirement during the documentation process.
	No Use of Standard for Documenting Errors	Because the requirement author did not use a standard for documenting the requirements.
	Specification Errors	Occur while specifying the requirements regardless of whether the developers correctly understood the requirements.

requirements) (Walia and Carver 2009). Initial evaluation of the RET demonstrated a relationship between performance during a training exercise and the number of errors committed during requirements creation (Walia and Carver 2013). This result suggests that error information can be helpful for prevention.

Table 2 Human Error Taxonomy (HET)

Reason's Taxonomy	Specific Requirements Engineering Errors	Error Description
Slips	Clerical Errors	Result from carelessness while performing mechanical transcriptions from one format or from one medium to another. Requirement examples include carelessness while documenting specifications from elicited user needs.
	Term substitution	Occur when requirement authors use the wrong term for a concept.
Lapses	Loss of information from stakeholders errors	Result from a requirement author forgetting, discarding or failing to store information or documents provided by stakeholders, e.g. some important user need.
	Accidentally overlooking requirements errors	Occur when the stakeholders who are the source of requirements assume that some requirements are obvious and fail to verbalize them.
	Multiple terms for the same concept errors	Occur when requirement authors fail to realize they have already defined a term for a concept and so introduce a new term at a later time.
Mistakes	Application Errors	Arise from a misunderstanding of the application or problem domain or a misunderstanding of some aspect of overall system functionality.
	Environment Errors	Result from lack of knowledge about the available infrastructure (e.g., tools, templates) that supports the elicitation, understanding, or documentation of software requirements.
	Solution Choice errors	These errors occur in the process of finding a solution for a stated and well-understood problem. If RE analysts do not understand the correct use of problem-solving methods and techniques, they might end up analyzing the problem incorrectly, and choose the wrong solution.
	Syntax errors	Occur when a requirement author misunderstands the grammatical rules of natural language or the rules, symbols, or standards in a formal specification language like UML.
	Information Management errors	Result from a lack of knowledge about standard requirements engineering or documentation practices and procedures within the organization.
	Wrong Assumption errors	Occur when the requirements author has a mistaken assumption about system features or stakeholder opinions.
	Mistake belief that it is impossible to sppcify non-functional requirements	The requirements engineer(s) may believe that non-functional requirements cannot be captured and therefore omit this process from their elicitation and development plans.

Table 2 (continued)

Reason's Taxonomy	Specific Requirements Engineering Errors	Error Description
	Not having a clear distinction between client and users	If RE practitioners fail to distinguish between clients and end users, or do not realize that the clients are distinct from end users, they may fail to gather and analyze the end users' requirements.
	Lack of awareness of requirements sources	Requirements gathering person is not aware of all stakeholders which he/she should contact in order to gather the complete set of user needs (including end users, customers, clients, and decision-makers).
	Inappropriate communication based on incomplete/faulty understanding of roles	Without proper understanding of developer roles, communication gaps may arise, either by failing to communicate at all or by ineffective communication. The management structure of project team resources is lacking.
	Inadequate Requirements Process	Occur when the requirement authors do not fully understand all of the requirements engineering steps necessary to ensure the software is complete and neglect to incorporate one or more essential steps into the plan.

2.4.2 Human Error Taxonomy (HET)

To more closely tie our work to human error research, we collaborated with a human error expert (Author Bradshaw) to develop a new taxonomy. We again performed a systematic literature review to identify specific requirements engineering human errors reported in the literature. But, this time we classified those errors into a predefined set of high-level error types drawn from human error research (Reason 1990) (see Table 2). Those three high-level error types are: *Mistakes* (planning errors in that someone designed an incorrect plan to achieve the desired goal), *Slips* (someone carries out a planned task incorrectly or in the wrong sequence), and *Lapses* (memory related failures; they occur when someone forgets a goal in the middle of a sequence of actions or omits a step in a routine sequence). While Table 2 overviews the error classes in the HET, the details of its development are beyond the scope of this paper and have been reported elsewhere (Anu et al. 2016; Hu et al. 2016).

2.4.3 Error Recovery

The mental nature of errors means that they cannot be observed directly. Thus, identifying the underlying errors based upon observed faults necessarily involves a process of *error recovery*. Errors for which the faults are immediately obvious (such as the slip of pouring orange juice instead of milk on one's breakfast cereal) can usually be described by the person who committed them. Conversely, when the fault is not immediately obvious and only appears after a delay of some time, a *retrospective report* from the person who committed the error cannot be relied upon. The quality of these reports is poor and often reflects an incorrect personal theory about human thought processes (Ericsson and Simon 1993) .

Therefore, it is not always possible to identify the specific error committed. To illustrate this difficulty, consider a situation where an important value is missing in the requirements specification. The requirements engineer could have committed a *lapse*, in which he elicited the information but forgot to record it in the documentation. Conversely, the requirements engineer could have committed a *mistake* by not having an effective plan for eliciting necessary information from the customers in the first place.

Fortunately, other project documentation may provide helpful clues that enable a forensic recovery of the error. If the missing value was present in early documentation but absent later, we can exclude the mistake interpretation in favor of the lapse. Accordingly, the process of error recovery relies heavily on a thorough analysis of context. In our work, both with the RET and the HET, we employ multiple raters who independently analyze all available documentation and classify errors, then compare their ratings. Overall, the raters' level of agreement was high, although occasionally we realize that a fault could be the result of multiple possible errors. In such cases, we drop the fault as unclassifiable.

We previously demonstrated that it was possible for students, trained to use the HET, to successfully analyzed observed requirements faults to identify the underlying errors. In this study, the participants were consequently able to find additional faults in their work, guided by their knowledge of existing errors (Hu et al. 2016). Our previous work did not investigate whether the HET could be used to prevent errors from happening in the first place. Here we report a study on the value of the HET on preventing human errors.

2.4.4 Comparison of the HET and the RET

We developed the RET through a review of the software engineering literature published from 1998-2006. The HET builds upon RET by including requirement errors published since 2006. Because we built the RET bottom-up (without a guiding theory) strictly based on the published software engineering literature, it is strikingly different from psychological taxonomies of human errors in other domains (e.g., aviation, medicine). Conversely, we developed the HET with guidance from human error theory (i.e Reason's Slip, Lapse, Mistake framework) and in consultation with a human error expert.

The biggest difference between the taxonomies is that the RET describes a mix of requirements *problems* (e.g., communication problems among developers) and requirements *errors* (the actual cause of the communication problems) whereas, the HET includes true human errors (i.e., inappropriate communication based on incomplete/faulty understanding of roles).

The following example illustrates the difference between the two taxonomies with a fault traced back to an error type in each taxonomy.

Fault: Definition of an important term (e.g., synchronize) is omitted from the requirements document rendering one or more requirements ambiguous or incomplete.

RET Error: Process (Elicitation Error) - The requirements engineering team did not use the proper method for collecting requirements, resulting in an inadequate elicitation process.

HET Error: Mistake (Information Management Error) - The requirement elicitation process did not have adequate procedures to ensure the proper transfer of information from the client meeting notes to the requirements document.

3 Evaluation Study

The goal of this study was to evaluate whether understanding Human Errors contributes to the prevention of errors and concomitant faults during requirements engineering. The following subsections describe the hypotheses, the variables, the participants, and the study procedures.

3.1 Research Hypotheses

Our previous work indicated that knowledge of requirement errors, as embodied in the RET, can help in fault prevention (Walia and Carver 2010). Subsequently, we developed the HET (as detailed in Section 2.4.2). In the current study, we investigate the effectiveness of this new taxonomy, built based upon human error theory, in preventing errors and compare its performance against the RET.

H1 - *The better a development team understands human errors, the fewer errors they will subsequently commit and the fewer faults they will consequently inject into their requirements document.*

For H1, we have two *participant variables* and two *dependent variables* as defined in Table 3. For the participant variables, we measure the participants' level of understanding of human errors in two ways. First, PV1 (See Table 3) measures the participants' conceptual understanding of human errors based on how accurately they can classify requirements errors into either the RET or the HET. Second, PV2 measures the participants' operational understanding of human errors based on how effectively they can use error information to locate faults in a document. For the dependent variables, the definitions of DV1 and DV2 follow directly from H1.

The three high-level HET error types (slips, lapses, mistakes) represent distinct types of human errors. To study whether the error types affect the results, the second hypothesis is:

H2 - *The better a development team understands each high-level HET error type, the fewer of that type of errors they will subsequently commit and the fewer faults they will consequently inject into their requirements document.*

For H2, we have one *participant variable* and one *dependent variable* as defined in Table 3. PV3 measures participants' understanding of high-level HET error types based upon how accurately they classify requirements errors into the three high-level HET error types. The definition of DV3 follows directly from H2.

3.2 Study Participants

The study included 31 senior-level undergraduate computer science majors enrolled in the Spring 2016 capstone course at the University of Alabama. In this course, students worked in teams to iterate through the software lifecycle and build a software system. The course instructor, independent of the research team, divided the participants into 10 3-person or 4-person teams.

Table 3 Study variables

	Participant variables	Dependent variables
H1	PV1: Accuracy of classifying requirement errors using HET or RET	DV1: # of errors committed during SRS development
	PV2: Effectiveness of using error information to find faults	DV2: # of faults injected in SRS
H2	PV3: Accuracy at classifiying slips, lapses and mistakes in HET	DV3: Corresponding # of slips, lapses and mistakes committed during the SRS development

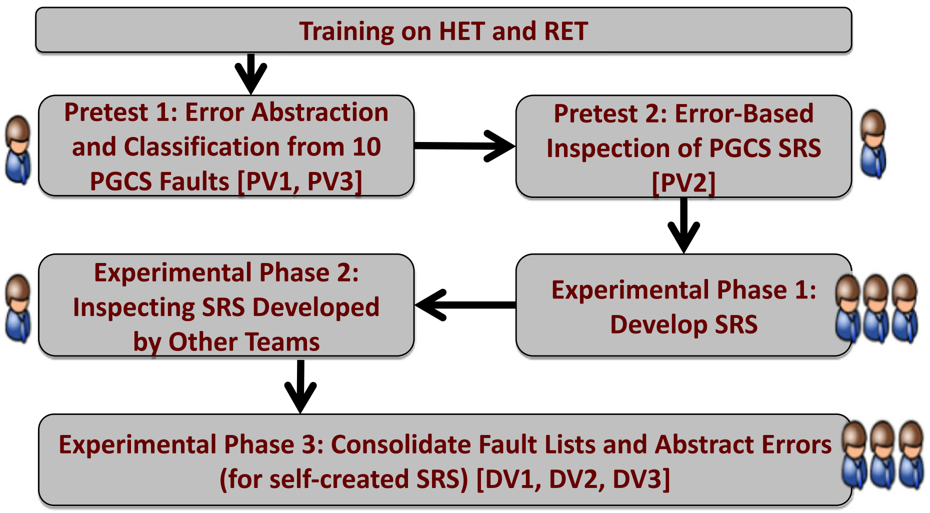

Training on HET and RET
Pretest 1: Error Abstraction and Classification from 10 PGCS Faults [PV1, PV3]
Pretest 2: Error-Based Inspection of PGCS SRS [PV2]
Experimental Phase 1: Develop SRS
Experimental Phase 2: Inspecting SRS Developed by Other Teams
Experimental Phase 3: Consolidate Fault Lists and Abstract Errors (for self-created SRS) [DV1, DV2, DV3]

Fig. 1 Study Procedure (Note that boxes with one person represent individual activities and boxes with multiple people represent group activities)

In order to assess the effectiveness of the HET training in reducing errors and consequent faults, we compared participants trained using the HET with participants that received similar training, but who were exposed to the original RET rather than the HET. The RET training group serves as the control in this experiment, instead of a traditional ‘no training’ control group. The primary reasons for choosing the RET as the control treatment are: (1) in a classroom setting it did not seem fair to expose some students to error training while others received no training at all; and (2) to prevent any bias that might result from comparing a treatment group that received training to a control group that received no training. We randomly assigned each team to either the RET group or the HET group.

3.3 Study Procedure

Figure 1 provides an overview of the study procedure, which included one training session, two pretests, and three experimental phases.

Error Training We conducted separate 45-minute training sessions for the HET group and for the RET group. In each session, author Hu trained participants on requirements inspections, fault detection, fault classes, and how to use either human error information (the HET group) or generic error classes (the RET group) to abstract and classify requirement errors from faults. Finally, we trained the participants on how to use the abstracted errors to guide the reinspection of a requirement document. The detailed plan for the training, along with the slides used, are available online.[1]

Pretest 1 – Error Abstraction and Classification This pretest measured how well the participants conceptually understood the errors from the Error Training. We supplied the participants with (1) the 14-page SRS for the Parking Garage Control System

[1] http://carver.cs.ua.edu/Studies/ErrorAbstraction/

(PGCS), which contained the purpose, scope, definitions, general overview, assumptions, functional requirements, and non-functional requirements; and (2) a list of 10 seeded faults. The PGCS document had a total of 30 seeded faults. We chose the subset of 10 faults to (1) make the task feasible to perform in a reasonable amount of time, (2) leave enough faults for the reinspection step (Pretest 2), and (3) ensure coverage of the three high-level error types from both taxonomies. As a homework assignment, the participants worked individually using knowledge from their Error Training to analyze the context in which each fault occurred in the SRS document. Then they abstracted each fault into its corresponding error and classified the error into the respective taxonomies (HET or RET).

The output of this step was 31 PGCS **Error Forms** (15 from participants who used the HET and 16 from participants who used the RET). We used the contents of these forms to measure *PV1: accuracy of classifying requirements errors using the RET or the HET* as the number of correctly classified errors divided by the total number of errors. Because the participants performed this step independently, we computed the team value for *PV1* by taking the average of the *PV1* values for each team member. We similarly computed *PV3: accuracy at classifying slips, lapses, and mistakes in the HET* as the number of correctly classified errors divided by the total number of errors for each of the three high-level HET error types and taking the average for each team.

Pretest 2 – Error-Based Inspection of PGCS This pretest measured how well the participants operationally understood the human errors from their training. Using the errors identified in Step 1, the participants individually inspected the PGCS SRS to identify additional faults. Note that there were 20 seeded faults in addition to the 10 provided in Pretest. The output of this step was 31 **PGCS Fault Forms** (one per participant). We measured the participants' level of understanding as *PV2: effectiveness of using error information to find additional requirement faults*. To compute the value for *PV2* for each team, we computed the total number of unique faults found by the individual team members (i.e. the union of their individual lists).

Experimental Phase 1 – Development of SRS To meet the requirements of the course, each team created an SRS document for a different system. These documents ranged from 11-31 pages (average of 16) and contained 7-25 requirements (average of 13.6).

Experimental Phase 2 – Inspection of SRS Participants individually used the HET or RET to inspect the SRSs developed by two other teams, one from the RET group and one from the HET group. The output of this step was 62 **Individual SRS Fault Forms** (two per participant).

Experimental Phase 3 – Consolidate Fault Lists and Abstract Errors Each team consolidated the results from Experimental Phase 2 into one comprehensive fault list for their own project. The team then used either the RET or HET (depending upon the group) to abstract those faults into the underlying errors. The output of this step was 10 **Final Team Fault and Error Forms** (one per team). We used these final forms to compute the values for DV1, DV2, and DV3.

3.4 Results and Analysis

This section provides a detailed analysis of the data collected during the study. This section is organized around the hypotheses posed in Section 3.1.

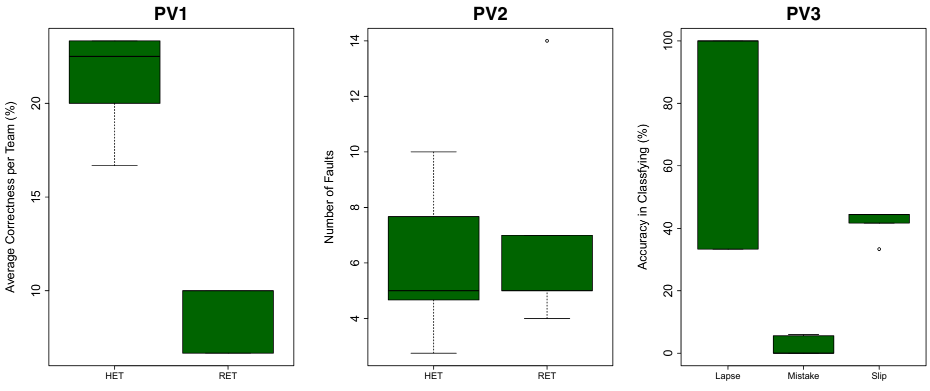
3.4.1 Descriptive Statistics

Before presenting the detailed results for the research questions, we give an overview of the data. Figure 2 shows the distribution of the Participant Variables. Figure 3 shows the distribution of the Dependent Variables. Note that because each team developed a different SRS document with a different level of complexity (which produced the data for DV1, DV2, and DV3), the numbers are not directly comparable across taxonomies. Therefore, we explore more complicated analyses in the sections that follow.

3.4.2 H1 - The Better a Development Team Understands Human Errors, the Fewer Errors They Will Subsequently Commit and the Fewer Faults they Will Consequently Inject Into their Requirements Document

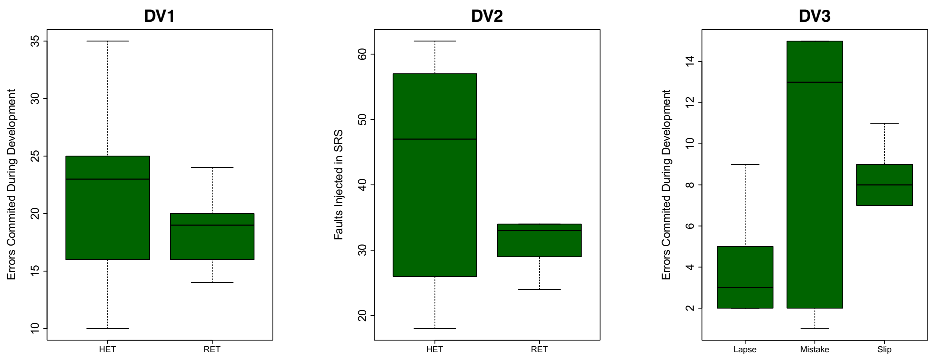
Hypothesis H1, at its core, suggests that an understanding of errors can prevent errors from happening in the future. Error prevention is distinct from error mitigation, in which requirements engineers detect faults that result from errors and correct those faults, perhaps with some corresponding change to the error-prone process that initiated the error-fault cascade. As a first step, we examined whether either error taxonomy (HET and RET) was easier for participants to understand and use. If one taxonomy helps participants gain a better understanding of errors than the other, we predict that taxonomy will consequently prove to be a more effective tool for error and fault prevention during the requirements creation process. Although the complexity and training approach for the taxonomies were similar, we believed that participants would find the HET more natural and easy-to-apply than the RET because it more closely maps onto the participants' understanding of how and why errors are made. Thus, we thought the HET group would be better at classifying errors than the RET group.

Our analysis showed that the teams trained with the HET were more effective at correctly classifying errors (21% correct) than the teams trained with the RET (8.6% correct). A between-subjects t-test showed that this difference was significant ($t_8 = 8.24$, $p < .0001$).



(a) PV1 (b) PV2 (c) PV3

Fig. 2 Descriptive Statistics - PVs



(a) DV1 (b) DV2 (c) DV3

Fig. 3 Descriptive Statistics - DVs

We now analyze H1 directly, taking into account the difference between HET and RET from the above analysis. We employed a multiple regression analysis to identify significant relationships in the data. We predicted that the number of new errors created in Experimental Phase 1 (DV1) would be a function of the group (HET vs RET), PV2 (the skill each team demonstrated in using the HET to find faults), and an interaction term. The overall model fit the data well, with an adjusted R-square of 0.689, $F_{3,6} = 7.644$, $p = .019$. All three predictor variables were significant as well: *HET/RET group* – t = -2.87, $p = .028$; *PV2 - errors classified* – t = -4.581, $p = .004$; and the *interaction term* – t = 2.76 $p = .033$.

The presence of a significant interaction implies that the relationship between how well a team uses error information to find faults (PV2) and the consequent errors teams injected (DV1) into their own projects differs between the two groups. We performed an additional analysis to understand this interaction. We correlated PV2 with DV1 and DV2 separately for the HET and the RET groups. Table 4 shows the results of these correlations. Teams who could effectively use the HET to find additional faults were significantly less likely to commit errors during SRS creation (r = -0.947, $p = .007$) and were less likely to inject faults during SRS creation (r = -0.999, $p = .0001$). The RET group showed no such relationship. In this experiment we found that training with the RET did not help prevent errors, while HET training did.

Table 4 H1 Results: PV2 vs. DV1 and DV2

Variable 1	Variable 2	HET	RET
PV2 – Effectiveness of using error information to find faults	DV1 – Likelihood of committing new errors during SRS creation	r = −0.947	r = −0.269
		p = 0.007	p = 0.331
PV2 – Effectiveness of using error information to find faults	DV2 – Likelihood of injecting new faults during SRS creation	r = −0.999	r = −0.161
		p = 0.0001	p = 0.398

[a]statistical tests are one-sided

3.4.3 H2: The Better a Development Team Understands Each High-Level HET Error Type, the Fewer of that Type of Error they will Subsequently Commit and Fewer Consequent Faults will be Injected Into Their Requirements Dsocument

To test this hypothesis, we performed analyses parallel to row 1 in Table 4. First, we computed the percentage of slips, lapses, and mistakes correctly abstracted in Pretest 1 by each team (PV3). Then we correlated those values with the corresponding errors produced during SRS creation. The results in Table 5 show that Lapses and Mistakes have a modest-to-moderate negative correlation. That is, an increase in a team's understanding of Lapses and Mistakes is related to a decrease in the number of those types of errors the teams commit when creating their own SRS document. These correlations, while in the predicted direction, failed to reach significance.

To better understand why these correlations may have been non-significant, we performed a *post hoc* power analysis. This analysis indicates that a sample size of 5 only has an 11% probability of detecting a correlation of strength *r = 0.5*. Additionally, the 10 errors abstracted in Pretest 1 were divided into the *slips/lapses/mistakes* error types, meaning that each measure was based on only 3 or 4 errors. That number may have been insufficient to measure each team's skill at classifying the three error types. Nevertheless, the moderate correlations we obtained suggest the proposed relationship exists and warrant further investigation using a larger sample size to increase statistical power.

3.4.4 Discussion

Overall, these results indicate that an increased understanding of the HET had more of an effect on error and fault prevention than did an increased understanding of the RET. As the HET is more closely tied to human error research, this result is encouraging. Furthermore, the results confirm that the better someone understand human errors, the fewer of those errors he or she will make when creating his or her own SRS document (Table 4). This finding is useful, going forward, as a basis for determining how best to train people on human error information. Finally, while we did see some moderate relationships within the specific human error types (slips, lapses, and mistakes), the small sample size in this study limited the power in our statistical tests. Therefore, further analysis, with larger datasets is needed to understand whether these relationships truly exist.

Table 5 Analysis Results: PV3 vs. DV3

Variable 1	Variable 2	Correlation

PV3a – accuracy at classifying *slips*	DV3a – # of *slips* committed during SRS development	r =.319
		p = 0.7

PV3b – accuracy at classifying *lapses*	DV3b – # of *lapses* committed during SRS development	r =−0.415
		p = 0.244

PV3c – accuracy at classifying *mistakes*	DV3c – # of *mistakes* committed during SRS development	r =−0.219
		p = 0.362

[a]statistical analyses used one-tailed tests

3.4.5 Threats to Validity

Internal Validity First because we did not observe the participants while they performed the study tasks, we cannot be sure that the they truly followed the specified HET or RET processes. Second, in Step 4, the participants inspected two SRS documents. It is possible that they became fatigued and did not perform as well on the second inspection. To reduce this threat, we gave them enough time to perform both inspections. Third, because the participants correctly abstracted a relatively small percentage of the errors (Pretest 1), it is possible that they did not fully understand the HET or the RET. Even so, the results still show a better understanding of human errors correlated with a reduced likelihood of inserting faults and errors. We can hypothesize that if participants understood the HET and RET even better (by performing better during Pretest 1), the correlation results would be even stronger.

External Validity The participants were undergraduate students. Therefore, the results are not directly applicable in an industrial context. Even so, the students were building a real system, so the activities performed in this study did have relevance to the projects. We will need additional studies to understand how these results apply in industry.

Construct Validity First, we defined understanding of human error in two ways in Section 3.4.2. Based on our study design, these definitions seemed to be the most appropriate. It is possible that other definitions would have provided different results. Second, because the students were building their own systems (rather than using systems with seeded faults/errors) we do not know the total number of errors made during SRS development. Our conclusions are based only on the errors that independent reviewers identified in Experimental Phase 2. Third, because the students were developing different systems, it is difficult to compare the results directly.

4 Industrial Study

The results of the classroom study provided evidence that the better a requirements engineer understands human errors, the less likely that requirements engineer is to make those errors when developing a requirements document. It also showed that the HET was more beneficial for error prevention than the RET. While this information is useful, it does not tell the whole story because error prevention relied primarily upon the vigilance of the requirements engineer to be aware of the potential errors and try, in an *ad hoc* way, not to commit them. This approach may be effective in smaller-scale classroom studies, where there is no time or opportunity to implement detailed error prevention mechanisms. Conversely, we hypothesized that because industrial projects tend to be more complex and longer-lasting than classroom projects and because industrial projects are often evolution or re-engineering efforts, they would require more detailed methods of handling errors and the resulting faults.

Because of the complexity of their projects, and the multi-tasking nature of their work, practitioners are likely to encounter some error types that do not manifest in a classroom setting (e.g. errors related to project management and customer interactions). In many cases practitioners may not be able to prevent errors, so they must also develop mechanisms for mitigating errors that have occurred. In the context of our work, *prevention* applies to approaches used to reduce or eliminate the likelihood of errors occurring, while *mitigation* applies to approaches used to reduce or eliminate the impact of errors that have already occurred (e.g. preventing related faults or failures). Therefore, we ask the following research question:

RQ - *What specific prevention/mitigation mechanisms do industrial practitioners employ for each HET error type.*

4.1 Data Sources

To investigate the research question, we analyzed data from two industrial surveys. We conducted the first survey (Section 4.1.1) specifically to address this research question. We are collaborating with a large team who conducted the second survey (Section 4.1.2) and were able to analyze a portion of the data relevant to our research question. The following subsections describe each of these surveys in more detail.

4.1.1 Survey of CAPS Requirements Engineers

CAPS (the Center for Advanced Public Safety) is a software development organization at the University of Alabama. The primary focus of CAPS work is developing software for public safety with specific applications to traffic safety, homeland security, motor vehicles, analytics, health and human services, weather and disaster responses, emergency medical services, and law enforcement. To elicit requirements and build these systems, CAPS personnel work with many state agencies in Alabama and in other states. CAPS employs a core staff of business analysts, software engineers, and software requirements engineers who work closely with clients to ensure that the systems contain features required by the customers and users. CAPS requirements engineers have rich experience about software requirements development. For this survey, we identified seven requirements engineers who had experience in requirements development or evaluation and conducted two surveys.

For the first survey, we performed a face-to-face introduction about requirements errors and their importance in the software development process by describing examples of requirements errors and the cost of these errors. This introduction helped the participants begin to think about the kinds of requirements errors that occurred in the past that could lead to project failures or increased costs. We then gave the participants the first survey containing questions 1 and 2 from Table 6, describing their requirements problems and prevention strategies for those problems. Note that we deliberately used the term *requirements*

Table 6 CAPS Survey Questions

Survey #	Question #	Question Text
1	1	What kinds of requirements problems have you encountered in your past software development processes?
	2	For those problems, what kinds of prevention strategies you have taken or plan to take to address these problems?
2	3	Based on the human error taxonomy that we introduced, are there additional human errors that you encountered which you did not report during the first survey?
	4	For the errors described in the previous question, what kinds of prevention strategies you have taken or plan to take to address these problems?

problems in the survey, rather than *requirements errors*. We made this choice so as not to discourage the respondents from providing as many answers as possible and not limiting themselves to their understanding of human errors.

For the second survey, we performed a face-to-face introduction of the HET by giving a detailed explanation of each high-level error type and each low-level error class. We then asked the participants to answer the second survey, which contained questions 3 and 4 from Table 6, describing any additional requirements problems that came to mind when considering the HET along with prevention strategies for those errors.

4.1.2 Naming the Pain in Requirements Engineering (NaPiRE) Survey

The NaPiRE project (Fernández et al. 2017) is an international consortium of researchers who conduct an ongoing investigation of requirements engineering in industry. One of the key interests of this consortium is understanding the problems that requirements practitioners face while performing requirements engineering tasks. The primary effort of the NaPiRE consortium is conducting bi-annual surveys of requirements practitioners across the world.

As a result of our participation in the NaPiRE consortium, we had access to data from the most recent survey. This data comes from 226 respondents, each from a different company, across 10 countries. While the survey covers a broad range of requirements engineering topics, our analysis focuses on the section of the survey dealing specifically with the problems respondents face during requirements engineering.

Previous NaPiRE surveys identified 21 common requirements problems (Table 7) faced by industrial practitioners. In the current survey, the respondents chose up to five of those problems that were the most common in their organization. The respondents then described the causes of those problems and any mechanisms undertaken to mitigate the problems. Note that because the NaPiRE survey addressed broader research questions than ours and we were not able to explain the concepts of *human errors* directly to the respondents, the survey used the term *requirements problems*.

Upon initial analysis of the survey results, we determined that some of the *problem causes* provide insight into human errors that may have occurred during the requirements engineering process. Most importantly, for the sake of our analysis, the information about how the respondents mitigated the problem causes helps us identify common practices for each type of error that can be useful to other requirements engineers. Our initial review of these responses indicated that they contained a mix of *prevention* and *mitigation* mechanisms, therefore providing relevant data for our analysis.

4.2 Data Collection and Analysis Procedure

We performed the data analysis in two phases. First, we identified which *requirements problems* were truly *human errors*. Then, we analyzed the prevention and mitigation approaches given for those errors. To ensure a valid data analysis process, three of the authors, each with different expertise and perspectives, participated in the analysis process:

1. *Wenhua Hu* – a PhD student with knowledge of software engineering, requirements engineering, and human error taxonomies;
2. *Jeffrey Carver* – a software engineering expert with experience in requirements quality and empirical data analysis; and
3. *Gary Bradshaw* – a cognitive psychologist with expertise in human error analysis and requirements quality.

Table 7 NaPiRE Survey: Common Requirements Problems (Fernández et al. 2017)

Problem #	Problem description
1	Communication flaws within the project team
2	Communication flaws between us and the customer
3	Terminological problems
4	Unclear responsibilities
5	Incomplete and / or hidden requirements
6	Insufficient support by project lead
7	Insufficient support by customer
8	Stakeholders with difficulties in separating requirements from previously known solution designs
9	Inconsistent requirements
10	Missing traceability
11	Moving targets (changing goals, business processes and / or requirements)
12	"Gold plating" (implementation of features without corresponding requirements)
13	Weak access to customer needs and / or (internal) business information
14	Weak knowledge of customer's application domain
15	Weak relationship to customer
16	Time boxing / Not enough time in general
17	Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements
18	Technically unfeasible requirements
19	Underspecified requirements that are too abstract and allow for various interpretations
20	Unclear / unmeasurable non-functional requirements
21	Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements

4.2.1 Phase 1 - Identification of Human Errors

The goal of this phase was to analyze all of the requirements problems reported in the CAPS survey and the NaPiRE survey to isolate only those that truly described human errors. We performed the following five steps separately on each data source.

1. *Step 1 – Individually identify human errors:* The CAPS interviewees described a total of 45 *requirements problems* commonly faced in their development experience. The NaPiRE survey results provided a total of 92 *problem causes*. We analyzed each reported problem to retain only those that truly described human errors. Each reviewer individually analyzed the reported problems from each data source to determine which were Slips, Lapses, or Mistakes. We also used the respondents' description of the prevention/mitigation strategies to help us understand the problems.
2. *Step 2 – Consolidation of the individual results from Step 1:* First, Hu combined the results of the individual analysis for each data source and marked any differences among the three reviewers. Then, all three reviewers met to discuss the differences in their Step 1 results. Based on these discussions, we resolved any discrepancies to produce an agreed-upon list of Slips, Lapses, and Mistakes for each data source.

3. *Step 3 – Map human errors to the specific requirements engineering errors in the HET:* Using the error description and the prevention/mitigation information, Hu mapped the errors identified in Step 2 from each data source into the Specific Requirements Engineering Errors in the HET (Table 2). The prevention and mitigation information often provided insight into what the respondent meant by each error description. In some cases, a single error description mapped to multiple specific HET errors. During this mapping, Hu identified *new human errors* that did not fit existing HET Specific Requirements Engineering Errors. She also identified errors that she could not categorize due to a lack of detailed information provided.
4. *Step 4 – Validation of Step 3:* The other two reviewers individually checked the results of Step 3 and marked any disagreements found.
5. *Step 5 – Finalization of Specific Requirements Engineering Errors:* Finally, the three reviewers met to resolve their disagreements from Step 4. While analyzing the industrial data, we encountered new Requirements Engineering human errors not already included in the HET. The industrial respondents reported Mistakes resulting from the presence of larger teams who interacted directly with customers. To accommodate these new mistake errors, we added four Specific Requirements Engineering Errors to the previous contents of the HET (see Table 8).

4.2.2 Phase 2 - Analysis of Prevention and Mitigation Mechanisms

After identifying the human errors in Phase 1, the goal of this phase was to analyze the prevention and mitigation mechanisms reported on the surveys (referred to as just *mechanisms* for the remainder of this discussion). Because the survey respondents may have had different understandings of the terms *prevention* and *mitigation*, we needed to perform our own analysis and classification, based on our definitions in this paper. We performed the following steps:

1. *Step 1 – Individually categorize mechanisms:* Phase 1 resulted in a total of 128 human errors. For each error, each reviewer individually analyzed the reported *mechanisms* to

Table 8 New Human Errors

Reason's Taxonomy	Specific Requirements Engineering Errors	Error Description
Mistakes	Inadequate communication plan between client and team	Result from the lack of an adequate plan to ensure all necessary information is communicated between the project team and the clients
	Inadequate plan for time management	The team management fails to properly match the project time requirements with the available time of the project team members
	Inadequate plan for selecting qualified team members	The team management does not have a process in place to ensure that all of the knowledge and skill required is present among the team members
	Inadequate plan for ensuring necessary resources	The team management fails to ensure that resources such as tools are available to the project team when needed

classify each as either *prevention* or *mitigation*. We used the error description and the prevention/mitigation information to help us understand the mechanisms.

2. *Step 2 – Group meeting:* During Step 1, we realized that it was not feasible to classify each mechanisms using only two categories (prevention or mitigation). Some mechanisms were both prevention and mitigation. Other mechanisms lacked enough detail to determine whether they were prevention, mitigation, or both. To address these situations, we added two categories: *both* (the mechanism is clearly both prevention and mitigation) and *not specified* (the description indicated the metric could be either prevention or mitigation or both, but lacked detail to specifically classify it into one category). Using this larger 4-category scheme, we collectively analyzed a small set of mechanisms as a pilot to ensure it was valid and that we had a consistent understanding.
3. *Step 3 – Individually recode the mechanisms:* Each reviewer independently repeated Step 1, this time using four categories instead of only two.
4. *Step 4 – Consolidation of the individual results from Step 3:* First, Hu combined the results of the individual analyses and marked any discrepancies among the three reviewers. Then, all three reviewers met to resolve the discrepancies and produce a final, agreed-upon category for each mechanism.

4.3 Results

This section describes the detailed information provided by the survey respondents. As we analyzed the prevention and mitigation mechanisms, we noted five high-level groups based upon the type of problem addressed or the strategy employed. These groups are – 1) Communication problems; 2) Changes to the requirements engineering procedures; 3) Additional project resources; 4) Changes to the broader management/administration process; and 5) Initiate an investigative process to better understand errors that were made. We then organized each of the four categories of mechanisms (i.e. *prevention*, *mitigation*, *both*, and *not specified* into these groupings).

Tables 9, 10, 11 and 12 show the detailed information for strategies that apply to both *prevention* and *mitigation*. Note that because the mechanisms in the *both (B)* and *not specified (NS)* categories all generally address both prevention and mitigation, we combined them into one set of tables. Tables 13, 14 and 15 describe the *mitigation* mechanisms. Tables 16, 17, 18 and 19 describe the *prevention* mechanisms. In these tables, the **Type** column (Tables 9 through 12 only) indicates whether the mechanism is categorized as *both* or as *not specified*. The **Reported Strategy** column contains the specific mechanisms as provided by the survey respondent, without any modification by us. The **Error Type** indicates which Specific Requirements Engineering Error from the HET the Reported Strategy addresses (note that the same value can appear more than once in this column if multiple participant-reported strategies map to the same Error Type). Note that we marked the new Specific Requirements Errors identified in the industrial surveys with '[New]'. Table 8 defines these error types. The **Freq** column indicates how many respondents provided the answer. The **Resource** column indicates which data source provided the mechanism.

4.4 Discussion

The survey respondents provided a total of 75 prevention and mitigation strategies across all error types (including the new human errors listed in Table 8). Of these, we classified 19

Table 9 Prevention/Mitigation Mechanisms for Communication Problems

Type	Reported strategy	Error type	Freq	Resource
B	Make sure all changes are communicated correctly with the client and team	[New] Inadequate communciation plan between client and team	1	CAPS
B	Ask questions to ensure the client is on the same page as you.	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
B	Don't assume that what you receive is what is needed, always ask for clarification and new requirements when inheriting software.	Inadequate Requirements Process	1	CAPS
B	Have the correct individuals present in the requirements-gathering meetings.	Inadequate Requirements Process	1	CAPS
B	Having the correct people involved, as with error 1 will help mitigate this problem as well.	Inadequate Requirements Process	1	CAPS
B	Make sure answers given make sense and don't leave more questions than answers.	Inadequate Requirements Process	1	CAPS
NS	Creation of glossaries	Inadequate Requirements Process	2	NaPiRE
NS	Guidance of stakeholders	[New] Inadequate communication plan between client and team	1	NaPiRE
NS	Introduction of communications tools	[New] Inadequate plan for time management	1	NaPiRE
NS	Planning and execution of regular communication events/ meetings	Information Management errors	1	NaPiRE
		Inadequate Requirements Process	1	NaPiRE
		Inappropriate communication based on incomplete/faulty understanding of roles	6	NaPiRE
		[New] Inadequate plan for selecting qualified team members	1	NaPiRE
NS	Promotion of knowledge transfer within project team	[New] Inadequate plan for time management	1	NaPiRE
NS	Repeatedly state that the developer does not have any domain knowledge	Application error	1	CAPS

Table 10 Prevention/Mitigation Mechanisms through an Investigative Process

Type	Reported strategy	Error type	Freq	Resource
NS	Evaluation of completed projects in order to derive lessons learned	Inadequate Requirements Process	1	NaPiRE
		[New] Inadequate plan for selecting qualified team members	1	NaPiRE
		Application errors	1	NaPiRE
		[New] Inadequate plan for ensuring necessary resources	1	NaPiRE

Table 11 Prevention/Mitigation Mechanisms through Changes to Procedure

Type	Reported strategy	Error type	Freq	Resource
B	Document models and solutions	Inadequate Requirements Process	1	NaPiRE
B	Elaborate dependencies	Inadequate Requirements Process	1	NaPiRE
B	Implementation of change management process	Inadequate Requirements Process	1	NaPiRE
B	Introduction and use of a requirements quantification approach	Inadequate Requirements Process	1	NaPiRE
B	Thoroughly document every client meeting.	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
NS	Promotion of knowledge transfer within project team	Inappropriate communication based on incomplete/faulty understanding of roles	1	NaPiRE
NS	Introduction of specification work shops	Information Management errors	1	NaPiRE
NS	Evaluation and introduction of tools	[New] Inadequate plan for ensuring necessary resources	2	NaPiRE
NS	Introduction and use of a requirements quantification approach	Inadequate Requirements Process	1	NaPiRE
NS	Introduction of an early feedback loop with customer	Inadequate Requirements Process	1	NaPiRE

as mitigation strategies, 31 as prevention strategies, and 25 as strategies for both prevention and mitigation. We make some observations from the results as follows.

Prevention and Mitigation Strategies (Tables 9 to 12) Of the 25 prevention and mitigation strategies, 12 belonged to Both (B) category and 13 to not specified (NS) category. All these strategies address *Mistake* errors from the HET (i.e., planning failures). This result is consistent with findings of prior human error research that people tend to catch *Slips* and *Lapses* (i.e., execution errors) in real-time, whereas it is more difficult for people to find mistakes. Those mistakes require system-wide changes to reduce their occurrence or mitigate their impact (Reason 1990).

Almost half of these strategies address various types of "Communication" problems (Table 9). This result is consistent with our previous findings that communication-related errors occur more frequently than errors related to process or management (Walia and

Table 12 Prevention/Mitigation Mechanisms through Changes to Resources

Type	Reported strategy	Error type	Freq	Resource
NS	Acquisition of (external) requirements experts	Information Management errors	1	NaPiRE
		[New] Inadequate plan for ensuring necessary resources	1	NaPiRE
		Inadequate Requirements Process	1	NaPiRE

Table 13 Mitigation Mechanisms through Changes to Procedure

Reported strategy	Error type	Freq	Resource
Cross checks with solution designs	Inappropriate communication based on incomplete/faulty understanding of roles	1	NaPiRE
Introduction of customer approvals	Inadequate Requirements Process	1	NaPiRE
Definition of quality criteria to make requirements testable and measurable	Mistaken belief that it is impossible to specify non-functional requirements	1	NaPiRE
Involve end users in the software development process from the beginning	Inadequate Requirements Process	1	CAPS
Use stronger formal reviews	Inadequate Requirements Process	5	NaPiRE
	Information Management errors	1	NaPiRE
Implementation of a monitoring approach to ensure the coverage of user expectations/requirements	Inadequate Requirements Process	1	NaPiRE
Implementation of a release process to ensure that requirements are final	Inadequate Requirements Process	1	NaPiRE
Integrate Testing and RE	[New] Inadequate plan for selecting qualified team members	2	NaPiRE
Introduction and use of check lists for monitoring requirements along their life cycles	Inadequate Requirements Process	2	NaPiRE
	Environment errors	1	NaPiRE
Introduction of a quality assurance approach for specifications	Information Management errors	1	NaPiRE
	Inadequate Requirements Process	1	NaPiRE
Introduction of an artifact based quality management (and traceability) approach	Information Management errors	1	NaPiRE
	[New] Inadequate plan for ensuring appropriate resources	1	NaPiRE
	[New] Inadequate plan for ensuring appropriate resources	1	NaPiRE
Introduction of an early feedback loop with customer	Inappropriate communication based on incomplete/faulty understanding of roles	1	NaPiRE
	Inadequate Requirements Process	1	NaPiRE
Designed prototypes for focus group to ensure the software would meet the needs of the users.	Inadequate Requirements Process	1	CAPS
Send out notes after all client meetings	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Specific Prototyping phases and guaranteed grant/project time for starting over after these phases.	Inadequate Requirements Process	1	CAPS

Table 13 (continued)

Reported strategy	Error type	Freq	Resource
When receiving requirements, think in terms of algorithms to make sure requirements are specific enough	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Ensure that we have deliverable dates (even soft dates)	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS

Carver 2013). Strategies to address the communication problems resulting from an incomplete or faulty understanding of roles (i.e., incorrect knowledge based plan) were the most frequently reported. These strategies included: careful planning, asking questions to ensure clear communication, and promotion of knowledge transfer within development team. The other frequently reported strategies were related to inadequate requirement process and information management errors. Therefore, based on the survey results, it seems as though ensuring proper communication and properly involvement of people during the requirements engineering process could be of great benefit.

Mitigation Strategies (Tables 13 through 15) Most of reported mitigation strategies focus on changes to procedures (Table The most commonly reported mitigation mechanism includes introduction (or use) of formal reviews or checklists for monitoring requirements and other process-level changes (e.g., integrating testing and RE). There is ample evidence in literature, and in other disciplines, that using stronger peer reviews and checklist can help find problems early and mitigate their impact on the later phases of development (Boehm and Basili 2001). Therefore, based on the survey results, it appears that gathering early feedback from clients and using formal quality assurance approaches may be beneficial.

Prevention Strategies (Tables 16 through 19) The strategies for prevention include those related to changing requirements engineering procedures (Table 16), changes to administration process (Table 17), and other communication problems (Table 18). This result is a consistent theme in other disciplines (aviation and medicine) that changes in administration (e.g., enforcing better training and using management plans) can help reduce the occurrence of common human errors (Nolan 2000).

The most frequently reported prevention strategies are related to inadequate requirements process (Table 16) and communication problems (Table 18). The strategies to prevent these problems include developing elicitation plans, using specification templates, implementing change management procedures, measuring non-functional requirements. The suggested strategies for preventing communication problems include use of clear documentation, asking follow up questions, and clearly defining the roles for clients and team members.

Table 14 Mitigation Mechanisms for Communication Problems

Reported strategy	Error type	Freq	Resource
People involved in the project just need to talk	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS

Table 15 Mitigation Mechanisms for Administration Problems

Reported strategy	Error type	Freq	Resource
Customers need to be coached on the importance of this stage of software development and how much it costs in support and maintenance to do otherwise	[New] Inadequate communication plan between client and team	1	CAPS

Table 16 Prevention Mechanisms through Changes to Procedure

Reported strategy	Error type	Freq	Resource
Have a clear process of handling change requests from client	Inadequate Requirements Process	1	CAPS
Determine if the new requirements are in scope	Inadequate Requirements Process	1	CAPS
Create a change board control group to discuss changes and potential impacts to the system	Inadequate Requirements Process	1	CAPS
Create a requirements specification template	Inadequate Requirements Process	1	NaPiRE
	Information Management errors	1	NaPiRE
Define a common structure to describe and explain requirements	Environment errors	1	NaPiRE
	Inappropriate communication based on incomplete/faulty understanding of roles	1	NaPiRE
Developers should be involved in the pitch and signing of any contract/ grant. They will spot vague and undecipherable requirements before a contract is signed or grant funded.	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Full requirements gathering along with wireframes supplied to developers	Application error	1	CAPS
Implement change management process	Inadequate Requirements Process	2	NaPiRE
Implement measurement techniques for non-functional requirements	Mistaken belief that it is impossible to specify non-functional requirements	1	NaPiRE
Introduce a standard	Inappropriate communication based on incomplete/faulty understanding of roles	1	NaPiRE
Introduce a standard	Inadequate Requirements Process	1	NaPiRE
Make sure the end user is involved in requirements gathering	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Plan elicitation before project begins	Inappropriate communication based on incomplete/faulty understanding of roles	1	NaPiRE
Perform a risk analysis is important and often completely overlooked. Requirements should allow time for heavy unit testing of high risk code, etc.	Inadequate Requirements Process	1	CAPS

Table 16 (continued)

Reported strategy	Error type	Freq	Resource
Go over the dataset with the clients and users to make sure everything is correct prior to creating the requirements document	Clerical error	1	CAPS
Explicitly ask the client what environment is desired if there is a technical option	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Notify client when a requirement is given that is not technically achievable	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Sign-offs before implementation	[New] Inadequate plan for time management	1	NaPiRE

Table 17 Prevention Mechanisms for Administration Problems

Reported strategy	Error type	Freq	Resource
Identify process gaps	Wrong Assumption errors	1	NaPiRE
Clearly define roles and responsibilities	Inadequate Requirements Process	1	NaPiRE
Increase awareness to focus development on customer requirements	Wrong Assumption errors	1	NaPiRE
Involve production team	Inappropriate communication based on incomplete/faulty understanding of roles	1	NaPiRE
Focus on requirements and not on solution	Information Management errors	1	NaPiRE
Plan and execute training to improve skill and performance of roles	Inappropriate communication based on incomplete/faulty understanding	1	NaPiRE
Better time management / planning	[New] Inadequate plan for time management	1	NaPiRE

Table 18 Prevention Mechanisms for Communication Problems

Reported strategy	Error type	Freq	Resource
Make sure to ask the right questions when talking with the client	Inadequate Requirements Process	1	CAPS
Make sure everything is clearly documented and outlined in the project plan and meeting minutes	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Make sure to ask questions when the requirements are unclear or seems too complex/simple.	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Present highly technical details in an easy to understand way	Inappropriate communication based on incomplete/faulty understanding of roles	1	CAPS
Specify up front that changes cost time	Inadequate Requirements Process	1	CAPS

Table 19 Prevention Mechanisms through Changes to Resources

Reported strategy	Error type	Freq	Resource
Acquisition of (external) requirements experts	Inadequate Requirements Process	1	NaPiRE

4.5 Threats to Validity

Internal Validity The information reported regarding prevention and mitigation mechanisms is based only upon what the respondents answered in the surveys. We had no way to conduct any objective studies to determine how useful any of the approaches are for preventing or mitigating errors in practice. We follow-up with the CAPS participants, who validated the result of our analysis. Because we did not select the NaPiRE respondents, we were not able to follow-up with them directly. Therefore, we need more studies to validate the usefulness of the prevention and mitigation approaches.

Construct Validity The NaPiRE survey did not provide any description of human errors or the HET, nor did it ask directly about human errors. The responses given fit well into the concepts of human error, but were done *post hoc* by us. It is not clear how the results might have differed had we been able to ask the NaPiRE respondents directly about human error or the HET. Similarly, in the CAPS survey, even though we explained human errors and the HET, it is possible that they did not have the same understanding of the concepts as we intended. To mitigate these threats, our analysis process excluded any responses that did not appear to truly represent human errors. However, even with our multi-person analysis team, it is possible that we misinterpreted some of the responses.

External Validity The data for this analysis came from two industrial surveys, one a large international survey and the other a focused survey in a local company. The responses from these participants may not be representative of all requirements engineering contexts. This threat is mitigated by the fact that the respondents to the NaPiRE survey represent over 200 companies from multiple countries.

5 Conclusion and Future Work

From the classroom study, we found the better the students understood human errors from the training process the fewer errors they made in their own documents. Therefore, we can conclude that knowledge of the HET is beneficial for students. For both the classroom data and the industry data, Mistake errors are the most prevalent. Furthermore, comparing the HET and the RET in the classroom study showed that knowledge of the HET had a greater fault and error prevention effect than knowledge of the RET. The industrial studies revealed some specific error prevention and mitigation mechanisms for different human errors in the HET. These results showed that an array of strategies related to communication problems, changes to requirements engineering process, and changes to management/administrative process can have a great effect on error prevention and mitigation. The industrial studies also provided a new type of Mistake error (i.e. a management error) to help extend the HET.

The primary contributions of this paper are (1) conclusions about the type of human error knowledge that helps prevent and mitigate errors and faults during software development, (2) evidence that a taxonomy based directly on human error information (the HET) is more effective in fault and error prevention, (3) insights into how the specific error types in each taxonomy contribute to the overall result, and (4) specific prevention and mitigation mechanisms for human errors.

As future work, we intend to perform more validation of the HET for error and fault prevention. While the results of the classroom study showed a positive impact of knowledge of the HET, we need to perform similar studies in professional settings to validate whether these results hold with participants who are more experienced. In the studies that we conducted with industrial professionals, we did not ask them to use the HET, only to report errors they had encountered. To address this shortcoming in our current work, we will conduct further, controlled studies with industrial practitioners to understand the effects of the HET and identify any aspects that it does not currently address. In this study, as we have done in classroom setting, we will first train the professional participants with HET/RET, then ask them to develop a real system with the knowledge of HET/RET. Then we can analyze whether the results obtained in the industrial setting are consistent with those obtained in the classroom setting.

Acknowledgements This work was supported by NSF awards 1421006 and 1423279.

References

Anu V, Hu W, Carver JC, Walia GS, Bradshaw G (2016) Development of a human error taxonomy for software requirements: A systematic literature review. Technical Report NDSU-CS-TR-16-001, North Dakota State University. http://vaibhavanu.com/NDSU-CS/TR-16-001.eps

Boehm B, Basili VR (2001) Software defect reduction top 10 list. Computer 34(1):135–137. https://doi.org/10.1109/2.962984

Card DN (1998) Learning from our mistakes with defect causal analysis. IEEE Softw 15(1):56–63

Chen JC, Huang SJ (2009) An empirical analysis of the impact of software development problem factors on software maintainability. J Syst Softw 82(6):981–992

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 Softw Eng 18(11):943–956

Dethomas A, Feo P (1987) Technology requirements of integrated, critical digital flight systems. In: Proceedings of the AIAA Guidance, Navigation and Control Conference, pp 1579–1583

Diller T, Helmrich G, Dunning S, Cox S, Buchanan A, Shappell S (2013) The human factors analysis classification system (HFACS) applied to health care. Am J Med Qual 29:181–190

Ericsson KA, Simon HA (1993) Protocol analysis : verbal reports as data /Revised Edition. MIT Press, Cambridge

Femmer H, Fernández DM, Juergens E, Klose M, Zimmer I, Zimmer J (2014) Rapid requirements checks with requirements smells: Two case studies. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, pp 10–19

Fernández DM, Wagner S, Kalinowski M, Felderer M, Mafra P, Vetrò A, Conte T, Christiansson MT, Greer D, Lassenius C, Männistö T, Nayabi M, Oivo M, Penzenstadler B, Pfahl D, Prikladnicki R, Ruhe G, Schekelmann A, Sen S, Spinola R, Tuzcu A, de la Vara JL, Wieringa R (2017) Naming the pain in requirements engineering. Empir Softw Eng 22(5):2298–2338

Grady RB (1996) Software failure analysis for high-return process improvement decisions. Hewlett Packard J 47:15–24

Graham M (2005) Software defect prevention using orthogonal defect prevention. https://www.umsec.umn.edu/sites/www.umsec.umn.edu/files/ODC_TwinSPIN_010605%20%281%29.ppt

Hamill M, Goseva-Popstojanova K (2009) Common trends in software fault and failure data. IEEE Trans Softw Eng 35(4):484–496

Hu W, Carver JC, Anu VK, Walia GS, Bradshaw G (2016) Detection of requirement errors and faults via a human error taxonomy: A feasibility study. In: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '16, pp 30:1–30:10

Hu W, Carver JC, Anu V, Walia G, Bradshaw G (2017) Defect prevention in requirements using human error information: An empirical study. In: Proceedings of the international working conference on requirements engineering, foundation for software quality. Springer, pp 61–76

IEEE (2017) 24765:2017 – Systems and Software Engineering – Vocabulary. IEEE

Kamsties E, M Berry D, Paech B (2001) Detecting ambiguities in requirements documents using inspections. In: Proceedings of the First Workshop on Inspection in Software Engineering

Kumaresh S, Baskaran R (2012) Experimental design on defect analysis in software process improvement. In: Proceedings of the IEEE International Conference on Recent Advances in Computing and Software Systems, pp 293–298

Lanubile F, Shull F, Basili VR (1998) Experimenting with error abstraction in requirements documents. In: Proceedings of the Fifth IEEE International Software Metrics Symposium, pp 114–121

Leszak M, Perry DE, Stoll D (2000) A case study in root cause defect analysis. In: Proceedings of the 22nd International Conference on Software Engineering, pp 428–437

Mays RG, Jones CL, Holloway GJ, Studinski DP (1990) Experiences with defect prevention. IBM Syst J 29(1):4–32

Nolan TW (2000) System changes to improve patient safety. BMJ Br Med J 320(7237):771

Pooley R, Senior D, Christie D (2002) Collecting and analyzing web-based project metrics. IEEE Softw 19(1):52–58

Raja UA (2009) Empirical studies of requirements validation techniques. In: 2009 2nd International Conference on Computer, Control and Communication, pp 1–9

Reason J (1990) Human error. Cambridge University Press, New York

Reason J (2008) The human contribution: unsafe acts, accidents and heroic recoveries. Ashgate Publishing

Reason J, Manstead A, Stradling S, Baxter J, Campbell K (1990) Errors and violations on the roads: a real distinction? Ergonomics 33(10-11):1315–1332

Saha GK (2006) Software fault avoidance issues. Ubiquity 2006(November):5:1–5:15

Shappell S, Wiegmann D (2001) Applying reason: The human factors analysis and classification system (HFACS). Human Factors and Aerospace Safety 1:59–86

Sommerville I (2010) Software Engineering, 9th edn. Addison-Wesley, Harlow

Suma V, Nair T (2008) Defect prevention approaches in medium scale it enterprises. In: Proceedings of the National Conference on Recent Research Trends in Information Technology, pp 134–138

Terzakis J (2013) Reducing requirements defect density by using mentoring to supplement training. Int J Adv Intell Syst 6(1):102–111

Walia GS, Carver JC (2009) A systematic literature review to identify and classify software requirement errors. Inf Softw Technol 51(7):1087–1109

Walia GS, Carver JC (2010) Evaluating the use of requirement error abstraction and classification method for preventing errors during artifact creation: A feasibility study. In: 21st IEEE International Symposium on Software Reliability Engineering, pp 81–90

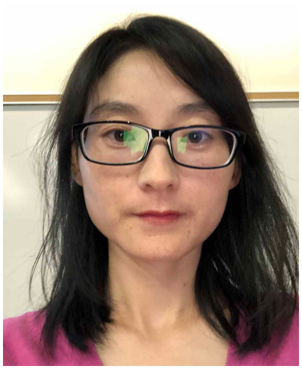
Walia GS, Carver JC (2013) Using error abstraction and classification to improve requirement quality: conclusions from a family of four empirical studies. Empir Softw Eng 18(4):625–658

Wickens CD, Hollands JG, Banbury S, Parasuraman R (2015) Engineering psychology & human performance. Psychology Press

Wiegmann D, Shappell S, Boquet A, Detwiler C, Holcomb K, Faaborg T (2005) Human error and general aviation accidents: A comprehensive, fine-grained analysis using HFACS. In: DOT/FAA/ AM-05/24

Wiener EL, Kanki BG, Helmreich RL (1995) Cockpit resource management. Gulf Professional Publishing

Zhang J, Patel VL, Johnson TR, Shortliffe EH (2004) A cognitive taxonomy of medical errors. J Biomed Inform 37(3):193–204



Wenhua Hu earned her PhD degree in Computer Science from the University of Alabama. She is an assistant professor in the Department of Software Engineering and Game Development at Kennesaw State University. Her main research interests include empirical software engineering, requirements engineering, human factors in software engineering and software testing. Contact her at whu4@kennesaw.edu

Jeffrey C. Carver is a professor of Computer Science at the University of Alabama. He earned his PhD from the University of Maryland. His research interests include empirical software engineering, requirements engineering, software quality, and software engineering for science. He is a senior member of IEEE and the ACM. Contact him at carver@cs.ua.edu.

Vaibhav Anu is a research assistant and a PhD candidate at the Department of Computer Science, North Dakota State University. He holds a Master of Science degree in Software Engineering. His research interests are on application of Cognitive Psychology research to improve human performance during the software development process, and application of natural language processing techniques for automation of the procedures involved in requirements engineering. He is a member of the IEEE Computer Society. Contact him at vaibhavanu.x@gmail.com.

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

Gary L. Bradshaw received his Ph.D. in Psychology from Carnegie-Mellon University. Currently a Professor of Psychology at Mississippi State University, Dr. Bradshaw's research interests include ambiguities and human error in software engineering, human adaptation in multitasking environments, and the testing effect. Contact him at glb2@psychology.msstate.edu.