Information and Software Technology xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Development of a human error taxonomy for software requirements: A systematic literature review

Vaibhav Anu[a,1], Wenhua Hu[b,1], Jeffrey C Carver[c], Gursimran S Walia[a,*], Gary Bradshaw[d]

[a] Department of Computer Science, Montclair State University, Montclair, NJ, United States
[b] Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA, United States
[c] Department of Computer Science, The University of Alabama, Tuscaloosa, AL, United States
[d] Department of Psychology, Starkville, MS, United States

ARTICLE INFO

Keywords:
Systematic review
Requirements
Human errors
Taxonomy

ABSTRACT

Background: Human-centric software engineering activities, such as requirements engineering, are prone to error. These human errors manifest as faults. To improve software quality, developers need methods to prevent and detect faults and their sources.
Aims: Human error research from the field of cognitive psychology focuses on understanding and categorizing the fallibilities of human cognition. In this paper, we applied concepts from human error research to the problem of software quality.
Method: We performed a systematic literature review of the software engineering and psychology literature to identify and classify human errors that occur during requirements engineering.
Results: We developed the Human Error Taxonomy (HET) by adding detailed error classes to Reason's well-known human error taxonomy of Slips, Lapses, and Mistakes.
Conclusion: The process of identifying and classifying human error identification provides a structured way to understand and prevent the human errors (and resulting faults) that occur during human-centric software engineering activities like requirements engineering. Software engineering can benefit from closer collaboration with cognitive psychology researchers.

1. Introduction

Software engineering, especially during the early phases, is a human-centric activity. Software engineers must gather customer needs, translate those needs into requirements, and validate the correctness, completeness, and feasibility of those requirements. Because of the involvement of various people in this process, there is the potential for human errors to occur. Cognitive Psychology researchers have studied how people make errors when performing different types of tasks. This line of research is called *human error* research. In this paper, we apply the findings from human error research to analyze and classify the types of human errors people make during the requirements engineering process.

Because the software engineering literature often has competing definitions for the same terms, we must clearly define our terminology. Based on IEEE Standard 24,765 [13] (ISO/IEC/IEEE 24,765:2010), we use the following definitions in this paper.

- Error (also referred to as Human Error) – A mental error, i.e. the failing of human cognition in the process of problem solving, planning, or execution.
- Fault [or Defect] – The manifestation of an error recorded in a software artifact.
- Failure – The incorrect execution of a software system, e.g. resulting in a crash, unexpected operation, or incorrect output.

The chain from human error to fault to failure is not unbroken and inevitable. Generally, developers either detect and correct faults without investigating the underlying errors or they use software testing to reveal system failures that they then repair. Nevertheless, because many system failures originate in a human error, researchers need to investigate how human errors can help in detecting and fixing software faults and failures. By one estimate, software failures (which often find their origination in human errors) cost $60 billion/year [27].

Practitioners in other domains, i.e. Aviation and Nuclear Power,

* Corresponding author.
E-mail addresses: vaibhavanu.x@gmail.com (V. Anu), whu4@kennesaw.edu (W. Hu), carver@cs.ua.edu (J.C. Carver), gursimran.walia@ndsu.edu (G.S. Walia), glb2@psychology.msstate.edu (G. Bradshaw).
[1] Authors Anu and Hu are co-first authors and contributed equally to leading this paper.

https://doi.org/10.1016/j.infsof.2018.06.011
Received 13 October 2016; Received in revised form 21 May 2018; Accepted 21 June 2018

also faced similar problems in handling problems that result from human errors. An error that originates in the mind of a pilot or of a reactor operator can produce a fault that results in a plane crash or a reactor core meltdown. In both cases, human errors create problems measured not only in billions of dollars but also in lives lost. The work of human error specialists has dramatically improved the safety and efficiency of these domains. The improvement process entailed (1) analyzing incident reports to identify the human error(s) made; (2) finding common error patterns; and (3) organizing those errors into a hierarchical *error taxonomy*. By examining the most frequent and costly errors in the hierarchy, investigators developed and implemented mediation strategies to reduce the frequency and severity of the errors [7,26,30]. Our premise in this paper is that a similar approach can be beneficial in software engineering.

Requirements engineering is largely human-centric. Requirement analysts must interact with customers or domain experts to identify, understand, and document potential system requirements. Numerous studies have shown that it is significantly cheaper to find and fix requirements problems during the requirements engineering activities than during later software development activities. Therefore, it is important to focus attention on approaches that can help developers improve the quality of the requirements and find requirements problems early.

Human error research can be particularly beneficial during requirements engineering. The high level of interaction and shared understanding among multiple stakeholders required at this step of the software lifecycle can result in a number of problems. Applying human error research to categorize the types of errors that occur during requirements engineering will have two benefits. First, it will provide a framework to help requirements engineers understand the types of errors that can occur during requirement development. This awareness should lead requirement engineers to be more careful. Second, it will help reviewers ensure that requirements are of sufficient quality for development to proceed.

Recognizing the potential benefit that a formalized error taxonomy could provide to the requirements engineering process, a subset of the current authors conducted a prior Systematic Literature Review to identify errors published in the literature from 1990–2006. Using a grounded theory approach [9], they organized those requirement errors into a Requirement Error Taxonomy to support the detection and prevention of the errors and resulting faults [28]. They performed that review without reference to contemporary psychological theories of how human errors occur. In this current review, we extend the previous review in two ways. First, rather than developing a requirement error taxonomy strictly 'bottom-up' with no guiding theory, we engage directly with a human error researcher and use human error theory to ensure that we *only include true human errors* and organize those errors based on a standard *human error classification system* from cognitive psychology. Second, this review includes papers published since the first review.

To guide this literature review, we focus on two research questions.

- *RQ1*: What types of requirements engineering human errors does the software engineering and psychology literature describe?
- *RQ2*: How can we organize the human errors identified in RQ1 into a taxonomy?
- The primary contributions of this work are:
- Adapting human error research to a new domain (software quality improvement) through interaction between software engineering researchers and a human error researcher;
- Development of a systematic human error taxonomy for requirements to help requirement engineers understand and avoid common human errors; and
- An analysis of the literature describing human errors in requirements engineering to provide insight into whether a community is forming around common ideas.

The remainder of this paper is organized as follows. Section 2 provides a background on software quality and on human errors (from a psychology perspective). Section 3 describes the systematic review process and its execution. Section 4 reports the results of the review and presents the human error taxonomy. Section 5 provides a discussion of the results and the usefulness of the human error taxonomy. Finally, Section 6 concludes the paper and describes future work.

2. Background

Formal efforts to improve software quality span decades [6,19]. This section briefly reviews some of these efforts as a preface to a more comprehensive discussion of the role human error research can play in identifying, understanding, correcting, and avoiding errors in software engineering. Given the importance of identifying and correcting errors and faults early in the software development process, we emphasize the requirements engineering process.

2.1. Historical perspectives on software quality improvement

Initially, software quality improvement research focused on *faults*. Because a fault that occurs early in software development may lead to a series of later faults, researchers developed Root Cause Analysis (RCA) [19,21] to identify the first fault in a (potentially lengthy) chain. In RCA, developers trace faults to their origin to identify the triggering fault. Then the developer can modify procedures to reduce the occurrence of faults or eliminate the original fault. The goal is to prevent the first fault, thereby eliminating the entire fault chain.

Because RCA is time-consuming, researchers developed the Orthogonal Defect Classification (ODC) to provide structure to the fault tracing process [6]. Developers use ODC to classify faults and identify the *trigger* that causes the fault to surface (not necessarily the cause of fault insertion). This approach accommodates a large number of fault reports and identifies the actions that revealed the fault. By focusing on the action that caused the fault, the ODC avoids the tedious work of tracing faults back to their origins and reduces the amount of time and effort required, compared with RCA.

RCA and ODC are *retrospective* techniques in which the investigative procedure begins with fault reports. Because the ODC relies on statistical analyses to identify the source fault, it requires the presence of a large and robust set of fault reports. As retrospective techniques, RCA and ODC occur late in the development process, after errors and faults have accumulated and are more expensive to fix. In addition, neither approach helps identify the human error underlying the fault. Thus, these approaches focus more on treating the symptoms (manifestations of the error) rather than the underlying cause (i.e., the error).

Noting that it can be difficult to define, classify, and quantify requirements faults, Lanubile, et al. shifted the emphasis from faults to errors. They defined the term *error* as a fault or flaw in the human thought process that occurs while trying to understand given information, while solving problems, or while using methods and tools [17]. Such errors are the cause of faults. This shift is important because it helps developers understand *why* the fault occurred. Because this approach addresses the underlying cause (i.e. the error) rather than the symptoms (i.e. the faults), it is an advance over the RCA and ODC approaches.

Errors and faults have a complex relationship. One error may cause several faults. Similarly, one fault may spring from different errors. Consequently, it is not trivial to identify the error behind the fault. Lanubile, et al. asked software inspectors to analyze the faults detected during an inspection to determine the underlying cause, i.e., the error. The inspectors then used the resulting list of errors to guide a re-inspection to detect additional faults related to those errors. This re-inspection produced only a moderate improvement in fault detection [17].

This error abstraction methodology is also a retrospective process