Chinese Journal of Aeronautics, (2017), xxx(xx): xxx–xxx

Chinese Society of Aeronautics and Astronautics
& Beihang University

Chinese Journal of Aeronautics

cja@buaa.edu.cn
www.sciencedirect.com

Software defect prevention based on human error theories

Fuqun Huang [a,b,*], Bin Liu [a]

[a] School of Reliability and System Engineering, Beihang University, Beijing 100083, China
[b] Institute of Interdisciplinary Scientists, Seattle 98115, WA, USA

Received 16 March 2016; revised 5 July 2016; accepted 23 December 2016

KEYWORDS

Human factor;
Human error;
Programming;
Root cause analysis;
Software defect prevention;
Software design;
Software quality;
Software psychology

Abstract Software defect prevention is an important way to reduce the defect introduction rate. As the primary cause of software defects, human error can be the key to understanding and preventing software defects. This paper proposes a defect prevention approach based on human error mechanisms: DPeHE. The approach includes both knowledge and regulation training in human error prevention. Knowledge training provides programmers with explicit knowledge on why programmers commit errors, what kinds of errors tend to be committed under different circumstances, and how these errors can be prevented. Regulation training further helps programmers to promote the awareness and ability to prevent human errors through practice. The practice is facilitated by a problem solving checklist and a root cause identification checklist. This paper provides a systematic framework that integrates knowledge across disciplines, e.g., cognitive science, software psychology and software engineering to defend against human errors in software development. Furthermore, we applied this approach in an international company at CMM Level 5 and a software development institution at CMM Level 1 in the Chinese Aviation Industry. The application cases show that the approach is feasible and effective in promoting developers' ability to prevent software defects, independent of process maturity levels.

1. Introduction

With the increasing dependence on software to realize complex functions in the modern aeronautic systems, software has become the major determinant of the systems' reliability and safety.[1] Software defect prevention (DP) is an important strategy to improve software quality and reduce development costs by preventing defects from reoccurring.[2,3] Defect prevention can be employed at the early stages of software development to reduce defect introduction rates and thus cut down the effort of defect detection and fixing.[4]

* Corresponding author at: School of Reliability and System Engineering, Beihang University, Beijing 100083, China.
E-mail address: huangfuqun@gmail.com (F. Huang).

Peer review under responsibility of Editorial Committee of CJA.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.cja.2017.03.005

Please cite this article in press as: Huang F, Liu B Software defect prevention based on human error theories, *Chin J Aeronaut* (2017), http://dx.doi.org/10.1016/j.cja.2017.03.005

Existing DP emphasizes software process improvement based on learning from historic failures. It proceeds as a specialized team chooses defect samples from a database, identifies causes of these defects and then produces strategies of process improvement for the current project.[2,5–9] This paradigm works effectively in preventing defects that are caused by process flaws, e.g. insufficient requirement tracking. However, software developers' cognitive failures are insufficiently addressed.[10]

An approach for preventing software developers' cognitive failures is urgently required in the Chinese Aviation Industry. One of the authors' previous studies,[11] which investigated a total of 3747 defects from 70 software systems developed by 29 Chinese aviation organizations, shows that individual cognitive failures cause 87% of the severe residual defects. This study also shows that the process maturity levels of most software development institutions in the Chinese aviation industry are below CMM Level 3; under this situation, the conventional DP process is hard to implement due to the lack of root-cause data and a systematic root-cause taxonomy.[10] Cognitive errors are the primary cause of software defects,[12] since software is the representation of human thoughts.[13,14] If software developers understand why, how, and when they are prone to commit errors, they could prevent errors more effectively.[15] It is scientifically interesting and an urgent demand to promote software developers' cognitive abilities to prevent software defects.

This paper proposes a human-centered defect prevention approach to improve software developers' cognitive abilities with respect to human error prevention. This approach emphasizes learning from software errors based on their underlying causal mechanisms. For convenience, we name this approach briefly as "defect prevention based on human error theories" (DPeHE). In the next section, we review the concepts and related work. In Section 3, we present the design of the approach. Section 4 presents two application cases. Section 5 presents the discussions and conclusion.

2. Related work

2.1. Traditional defect prevention

The DP processes reported in the existing literature are generally similar to the traditional defect prevention process[2] or the corresponding defect causal analysis process (D. N. Card, 2006). The typical process of conventional DP includes the following major steps[6]: (1) select defect samples from historical database, (2) hold causal analysis meeting to identify defect causes and develop prevention action proposals, and (3) implement preventative actions. Therefore, to implement DP, three resource elements are required: (1) a database and tool for defect sample selection and action tracking, (2) a cause classification method for identifying root causes, and (3) an action team with good expertise of root cause analysis. DP has shown to be effective in preventing software defects, especially from a perspective of process improvement.[4,5] Despite the great progress made in past years, some aspects require further study.

The key of conventional defect prevention is how to identify root causes. However, this process strongly depends on expert experience. Root cause taxonomies are proposed to aid root cause analysis, which are summarized in Table 1.

Table 1 Summary of existing root cause classifications.

Author	Classification
Mays, Jones, Holloway and Studinski[3]	Communication, oversight, education, and transcription
Leszak, Perry and Stoll[17]	Change coordination, lack of domain knowledge, lack of system knowledge, lack of tool knowledge, lack of system knowledge, lack of tools knowledge, lack of process knowledge., Individual mistake, introduced with other repair, communication problems, missing awareness for need of communication, and not applicable
Card[6]	Methods, which may be incomplete,
Kalinowski, Travassos and Card[8]	ambiguous, wrong, or unenforced; tools and environment, which may be clumsy, unreliable, or defective; people, who may lack adequate training or understanding; and input and requirements, which may be incomplete, ambiguous, or defective

Unfortunately, these taxonomies seem too abstract to be helpful for those organizations with little experience. Root causes are generally classified into four categories: method, people, tool, and requirement; detailed causes are analyzed by brainstorming with cause-effect diagrams.[7–9] Generally, brainstorming is a good method for gathering new ideas, but it is not as reliable for repeated use. For example, defect causes analyzed by different people may fall into different categories, be overlapped or even be omitted. This problem reduces the possibility of recurring good effects across different organizations. It is no accident that defect causal analysis is often misunderstood and misapplied in the software industry.[7]

The other problem is how to choose defect samples. The common method is choosing according to defect type, which is generally defined by the orthogonal defect classification (ODC).[16] However, there is a lack of scientific criteria to help people make decisions about what historical defects have heuristic meanings for the current project. Again, this process strongly depends on the experience of the team members. As a result, it is hard for those organizations with scarce data to implement DP, as there are hardly any defect samples for reference.

In the authors' previous study,[10] a structural taxonomy of root causes has been recently proposed to address these problems. More efforts are required to provide software developers with explicit knowledge strategies, as well as a training program to improve their awareness and abilities to prevent human errors. DPeHE is proposed to address these needs.

2.2. Human errors

Human errors have been studied along with developments in cognitive psychology since the 1970s. In terms of fundamental theories, Reason's work (1990) is regarded as the most systematic theory that is widely accepted and applied in safety critical domains. However, these theories tend to be "theoretical and less analytical".[18,19] Only after being adapted and integrated to the application contexts can these basic human error modes be very powerful in practice.

Please cite this article in press as: Huang F, Liu B Software defect prevention based on human error theories, *Chin J Aeronaut* (2017), http://dx.doi.org/10.1016/j.cja.2017.03.005