The Journal of Systems and Software 125 (2017) 197–206

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Creating an invalid defect classification model using text mining on server development

Yihsiung Su[a,*], Pin Luarn[b], Yue-Shi Lee[c], Show-Jane Yen[c]

[a] Graduate Institute of Management, National Taiwan University of Science and Technology, 2F, 196 Hopien N. St., Sanchung District, New Taipei City 241, Taiwan
[b] Department of Business Administration, National Taiwan University of Science and Technology, No. 43, Keelung Rd., Sec. 4, Da'an Dist., Taipei City 10607, Taiwan
[c] Department of Computer Science and Information Engineering, Ming Chuan University, No. 5, The-Ming Rd., Gwei Shan District, Taoyuan County 333, Taiwan

ARTICLE INFO

Article history:
Received 24 July 2016
Revised 28 November 2016
Accepted 5 December 2016
Available online 6 December 2016

Keywords:
Invalid defect
Classification
Text mining
Server development
Project management
BIOS

ABSTRACT

Invalid defects, which are often overlooked, reduce development productivity and efficiency. This study used exploratory study and text mining to answer three research questions related to invalid defects in two research stages.

In the first stage, we filtered 231 invalid BIOS (basic input/output system) defects from the 3347 defects of three server projects. These defects were from numerous function areas owned by virtual teams located in Taiwan, China, and the United States. Results indicated that BIOS firmware demonstrates the maximum number of defects and invalid defects. This firmware accounted for 43.3% defects and 33% invalid defects in server development. Results determined that invalid defect classification that includes four types, namely, working as designed (WAD), user error, duplicate, and others. All of these types can be grouped under the term WUDO. WAD accounts for the maximum of 45% of invalid defects in the WUDO classification. In the second stage, this study determined a stable classification algorithm, namely, decision tree C4.5, to classify the invalid defect types.

This study helps project teams for information technology products to classify the different invalid defect types that developers and testers face. Results can improve project team productivity and mitigate project risks in project management.

1. Introduction

Invalid defects are often overlooked by development teams, and developers do not have to provide any solution for these defects when the products are launched. However, teams still need to spend resources on investigating these invalid defects. Defects are unavoidable during product development, and they are a key factor in affecting project success. Developers must investigate the root causes of valid defects, and solutions must be provided to fix these defects. Test engineers must validate these defects to achieve quality goals. However, using resources to investigate invalid defects is impractical. Teams must determine the root causes of invalid defects to improve development efficiency and to reduce project risks, and project teams must apply lessons learned for invalid defect prevention and reduction.

Few studies have been conducted to identify the causes of invalid defects, and no study has attempted to demonstrate the causes of invalid defects in server development, which involves hardware, software, firmware, operation system (OS), and other adapters. This study aims to classify invalid defect types for invalid defect elimination and reduction by identifying the root causes of invalid defects and by creating an invalid defect classification (IDC) model.

First, this study aims to identify invalid defects and to determine which functional area demonstrates the maximum number of defects and invalid defects. Second, developers and testers review and study the defect reports in this functional area and then determine the causes of invalid defects. Third, we establish the IDC based on the causes of invalid defects. Finally, we use five classification algorithms, namely, decision tree C4.5, naïve Bayesian classification, Bayesian network, logistic regression, and neural network, to build the classification model (Wu et al., 2008). These text mining algorithms are used in demonstrating the accuracy of each algorithm in the IDC model, and the best solution for this model

* Corresponding author.
E-mail addresses: simonyhsu@gmail.com (Y. Su), luarn@ba.ntust.edu.tw (P. Luarn), leeys@mail.mcu.edu.tw (Y.-S. Lee), sjyen@mail.mcu.edu.tw (S.-J. Yen).

http://dx.doi.org/10.1016/j.jss.2016.12.005

is identified. The following three research questions (RQ) are explored in this study.

RQ1: Which function area reflects the maximum number of defects and invalid defects in server development projects?

RQ2: What are the classifications of invalid defects during server development?

RQ3: Can we create an IDC model to classify invalid defect types?

We design two research stages to answer the three research questions. In the first stage, we collect samples from the three server projects, extract invalid defects, have experts decide the root causes of the invalid defects, and establish an IDC to answer RQ1 and RQ2.

In the second stage, this study uses the results of Stage 1 and text mining to build the classification models and to compare the performance for answering RQ3.

In brief, the process of text mining in the second stage includes the following steps:

1. Dataset collection (from the results of Stage 1)
2. Keyword extraction
3. Feature vector creation
4. Using two methods to train and evaluate the classification models
   i. In **Method 1**, we use the datasets of Project 1 (Proj 1) and Project 2 (Proj 2) to train the classification models, and the dataset of Project 3 (Proj 3) to evaluate the accuracy of the classification models.
   ii. In **Method 2**, we use the datasets of Project 1 (Proj 1), Project 2 (Proj 2), and Project 3 (Proj 3) to train the classification models and apply a 10-fold cross validation (Kohavi, 1995) to evaluate the accuracy of the classification models.

2. Review of literature and related work

Software engineering is also concerned with software project management, development process, tools, and methods to support software production (Sommerville, 2011). For the embedded software, one of the differences in software engineering is that the engineer needs additional knowledge related to electronic devices (Oshana and Kraeling, 2013). Embedded software must react to any event generated by the hardware. For example, basic input/output system (BIOS), which is a boot firmware, performs power initiation on a server. The BIOS in this server system is embedded in a system hardware. It is often in a read-only memory and usually responds in real time. Embedded software is very important because it constitutes nearly all electronic devices (Sommerville, 2011).

Poppendieck and Poppendieck (2003) introduced lean software development (LSD) and stated that a defect is one of seven wastes, and eliminating waste is one of seven principles in LSD. The performance index of project success aims to achieve project scope on time and within budget. The majority of projects are managed to meet these targets using the book *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, which presents a set of standard terminology and guidelines for projects as well as introduces the related knowledge, tools, processes, skills, and techniques (Project Management Institute, 2013).

In addition to achieving the performance index of project success, increasing productivity and efficiency is another project team and organization goal. Defect reduction and prevention is a major factor to achieve the aforementioned goal. Causal analysis and resolution (CAR) is a process of maturity level 5 in Capability Maturity Model Integration (CMMI). Teams can use CAR to identify causes of selected defects and to prevent defects in the future. These effective process changes can result in process improvement to meet the optimized target (CMMI Product Team, 2010).

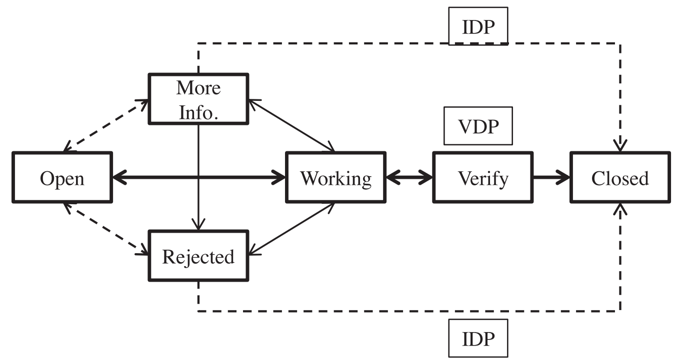

Fig. 1. Defect state diagram.

2.1. Invalid defects

Invalid defects, which are often overlooked by a project team, are rarely discussed at meetings before a project closes. Wang et al. (2011) described the percentage of unclear defect reports and invalid defect reports in software products. They also tested the improvement practices based on the causes of unclear defect reports and invalid defect reports. According to their research, the proportion of invalid defect reports is approximately 26% and 35% of test engineers submit these reports. Invalid defect reports usually delay a schedule and increase the amount of team effort to identify problems and re-communication.

Sun (2011) stated that the average percentage of invalid defects in mobile application is 8.31%, and the causes include errors in testing and external systems, misunderstandings on functionality and test environment, and insufficient background knowledge. Kaplan (1993) introduced defect prevention process, which offers an opportunity to save millions of dollars in development cost. First, this process aims to identify defects, and it suggests preventive action by causal analysis teams. Second, action teams prioritize and implement improved process and provide a feedback on the status of pending improvements. Third, team members receive feedback on process changes in project kickoff meetings. Finally, team members track defects and actions using database and data collection.

Li et al. (2012) described that software companies often use a defect tracking system (DTS) as a defect management tool to ensure that a defect is fixed. Software quality can be improved for future projects using the data in a DTS. One example of a DTS that enables defect and change tracking and management is IBM Rational ClearQuest (CQ) (Wahli, 2004). Taiwan Development Center (TDC) used this tool to manage defects created by each team during a product development lifecycle. In this DTS, three major roles exist: owners, submitters, and stakeholders. The defect submitter (submitter) uses CQ to issue change requests, to verify the defects fixed, and to add information to change requests. The defect owner (owner) is responsible for investigating the defect, providing the fixes for the defect, and eliminating the defect. The stakeholders are managers, project managers, or any team member who wants to review the status of the defects. A DTS can be accessed by a web browser; thus, everyone can query the defects anytime and anywhere.

2.2. Definition of invalid defects

Invalid defects are defects determined through an invalid defect path (IDP), and the owner does not provide any fix to close these defects. Fig. 1 shows the defect state diagram on DTS. Two major paths, namely, valid defect path (VDP) and IDP, exist to identify if the defect is valid or invalid.