The Journal of Systems and Software 113 (2016) 27–43

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

How do bugs surface? A comprehensive study on the characteristics of software bugs manifestation

Domenico Cotroneo[a], Roberto Pietrantuono[a,*], Stefano Russo[a], Kishor Trivedi[b]

[a] Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, Università di Napoli Federico II Via Claudio 21, 80125 Naples, Italy
[b] Duke High Availability Assurance Lab, ECE Department, Duke University, 27708 Durham, NC, USA

ARTICLE INFO

Article history:
Received 29 October 2014
Revised 9 October 2015
Accepted 11 November 2015
Available online 2 December 2015

Keywords:
Bug manifestation
Failure
Bug characterization

ABSTRACT

The impact of software bugs on today's system failures is of primary concern. Many bugs are detected and removed during testing, while others do not show up easily at development time and manifest themselves only as operational failures. Besides the importance of understanding the bug features from the programmer perspective (i.e., what is wrong in the code), a key role in counteracting bugs is played by the chain that from the bug activation leads to failure.

This article investigates the characteristics of the bug manifestation process. Through an extensive empirical study, a set of failure-exposing conditions is first identified as bug manifestation characteristics; 666 bug reports from two applications are then analyzed with respect to these characteristics under several perspectives. Findings highlight: (i) the main occurrence patterns of bug triggering conditions in the selected case studies and the role played by the workload, the application and the environment where it runs; (ii) how such conditions evolve over time; (iii) how they relate to bug exposure and fixing difficulty; (iv) how they impact the user. Results provide a fine-grain characterization of bug manifestation that is expected to increase the perceived importance of this dimension in testing, debugging, and fault tolerance strategies.

1. Introduction

Understanding software bugs is of paramount importance to improve software quality and development processes. Researchers, across years, analyzed bugs from different viewpoints to improve the knowledge about their characteristics. Regardless of the semantics of the error committed by a developer, a fundamental aspect in bug comprehension is related to the process by which a bug manifests itself as a failure. Indeed, while static properties of a bug (e.g., its type, or origin) are related to how a bug is introduced in the code, there are different causes for a bug provoking a failure. Expectedly, many bugs systematically cause the same failure on a given (sequence of) input(s). Conversely, there is a non-negligible set of bugs that cause a failure depending on the state of the execution environment, appearing as non-deterministic or transient, in which the failure does not occur unless the environment is in a certain state (Gray, 1985; Grottke and Trivedi, 2007). The latter category contains bugs that likely escaped testing, since their exposure may be a rare event, and different V&V techniques (e.g., static analysis) or runtime fault tolerance are the means to cope with them. In general, the effectiveness of fault detection and tolerance strategies are strictly tied to how bugs manifest themselves.

In the past, some broad classifications took into account the properties of bugs related to the reproducibility of the failures they cause. Gray (1985) distinguished Bohrbugs and Heisenbugs depending on whether the failure caused by the bug is systematically reproducible or not (also called *hard* and *soft* failures, respectively). Later, Trivedi and Grottke define Mandelbug in lieu of Heisenbug (Grottke and Trivedi, 2007), considering the complexity of the bug-failure process in terms of macro-conditions required for a bug to cause a failure (i.e., influence of the execution environment, timing or ordering of inputs or operations, time lag between bug activation and failure occurrence). Several researchers conducted empirical studies and indirectly highlighted the importance of distinguishing the environment as collateral cause of bug exposure (Chandra and Chen, 2000; Lee and Iyer, 1995; Grottke et al., 2010). Others have been highlighting some factors of the execution environment, such as memory (Sullivan and Chillarege, 1991), concurrency (Lu et al., 2008), or resource management (Cotroneo et al., 2013b), as relevant failure causes, and motivated subsequent research on developing proper countermeasures (e.g., static and dynamic analysis tools). A few papers clearly distinguish the need for studying some characteristic of the bug manifesta-

* Corresponding author. Tel.: +39 081676770.
E-mail addresses: cotroneo@unina.it (D. Cotroneo), roberto.pietrantuono@unina.it (R. Pietrantuono), stefano.russo@unina.it (S. Russo), ktrivedi@duke.edu (K. Trivedi).

http://dx.doi.org/10.1016/j.jss.2015.11.021
0164-1212/© 2015 Elsevier Inc. All rights reserved.

Table 1
Summary of findings by type of analysis.

Triggering conditions analysis	
#1	Most of the reported bugs (80.71%) needs only workload conditions to surface, with no environmental condition required
Workload (WL) triggering conditions	
#2	For 57.87% of bug reports, the bug manifestation requires a specific *request type* For 35.04%, a further additional condition is required
#3	For 35.22% of bug reports, the bug manifestation requires a *requests sequence* For 24.78%, a further additional condition is required
#4	For 50.62% of bug reports, the bug manifestation requires a second WL condition, related to the *input type*, to the *application configuration*, or to the *input value*
#5	For 11.68% of bug reports, the bug manifestation requires a specific *input value* (e.g., boundary values); for 4.78%, it requires a specific *range of values*; for 3.54%, it requires a *class of values* with a property in common
#6	For 10.79% of bug reports, the bug manifestation requires a specific *request type* together with a specific *application configuration*; for 9.02%, it requires a specific *request type* together with a specific *input value*; for 7.08%, it requires a specific *request type* together with a specific *input type*
Environment triggering conditions	
#7	63.33% of bug reports with execution environment triggers are caused by "indirect" environmental conditions
#8	For 7.25% of bug reports, the bug manifestation requires a *concurrency* condition
#9	For 6.19% of bug reports, the bug manifestation is related to *memory management*
Workload and environment triggering conditions	
#10	For 76.15% of *environment-dependent* bug reports, the bug manifestation requires, besides the environmental condition, a sequence of request as workload condition
#11	For 7.26% of bug reports, the bug manifestation requires a *transient* environmental conditions
#12	For 21.24% of bug reports, the bug manifestation requires a specific *environment configuration*
#13	When a specific *environment configuration* is required, the necessary factor is: a specific OS (55.83%); specific system-level or application-level software (24.17 %), a specific hardware/network configuration (20.00%)
Temporal analysis	
#14	*Environment-dependent* bug reports tend to start appearing later and have a slower increasing rate than *workload-dependent* ones
#15	The ratio of environment-dependent over workload-dependent bugs in MySQL is lower in the first 4 years than in the period 4–8 years after the release time
Complexity analysis	
#16	For 45.66% of bug reports, the bug manifestation requires at least two conditions to surface, more often two *workload* conditions (37.87%)
#17	Bugs requiring one triggering condition (25.49%) are less common than bugs requiring two conditions (45.66%)
#18	For 7.61% of bug reports, the bug manifestation requires 4 conditions together to surface; for 1.06%, it requires 5 conditions
#19	We cannot state that there is a relation between the bug manifestation, expressed by triggers, and the time to fix a bug
Impact analysis	
#20	The manifestation of most of total bugs ended up in an incorrect response provided to the user (62.12%), or in a crash of the application (26.90%). The remaining bugs resulted in performance issues (5.84%) or omission failures (3.72%)
#21	The *failure mode* is affected (p-*Value* < 0.01) by the type of bug, being *environment-* or *workload-dependent*
#22	The relative proportions of high and low severity bugs are influenced by the bug type, with the percentage of environment-dependent bugs being more relevant for the high severity class (p-*Value* < 0.02).

tion, such as the number of inputs required for a bug to surface, case framed within a wider context (e.g., concurrency bugs Lu et al., 2008, or server bugs Sahoo et al., 2010). Despite the merit of these studies in pointing out the need for examining the entire bug-failure chain, none of them sets the goal of systematically investigating the characteristics of the bug manifestation process. In this work, we present the results of a comprehensive empirical study whose aim is to examine the fine-grain conditions that make a bug cause a failure. A set of bugs are analyzed in terms of "*triggers*" – that is, of necessary failure-exposing conditions that make a bug surface. These are related both to the input workload and to the environment necessary for a bug exposure. By abstracting factors that potentially affect the bug activation and/or propagation process, a set of conditions are identified as triggers, and used to categorize the bug manifestation characteristics. On a set of 666 bug reports taken from the Apache Web Server and the MySQL DBMS, we first analyzed the occurred *patterns of bug triggers*, to figure out the most common factors exposing bugs in the considered case studies. Then, we have investigated the relation of such triggers with complexity in terms of bug exposure difficulty and fixing time, to see if they are related only to the detection or also to the bug fixing process. Their *evolution over time* is also investigated, so as to detect possible differences in the way such triggers appear over time (e.g., if triggers related to the environment differ from triggers related only to the workload in terms of occurrence time). Finally, the relation of triggers with the *impact* of the failure they cause is studied, so as to assess if the exposure characteristics of a bug are also related to the end-user perception of the caused failure. The study provides a set of findings, referred to the selected case studies, summarized in Table 1, which highlight: (i) the impact of each workload condition on bug surfacing, both when it is a *necessary* condition and when it is a *necessary and sufficient* condition; (ii) the additional impact caused by the environment conditions, and which factor of the environment is more relevant; (iii) the impact of the combination of a number of conditions together; (iv) the occurrence pattern of different triggering conditions at operational time; and (v) the relation of workload and environment bug triggers with the failure modes and the perceived severity. Far from claiming the generality of what we observed, the study serves to point out the many differences among bugs in terms of manifestation characteristics. We believe this can foster further investigation along such an important dimension, contributing to figure out how to exploit the knowledge of bug-failure chain for improving development processes. In the following, we first survey past studies on bug characteristics in the literature (Section 2); in Section 3, we present the study methodology; Sections 4, 5, 6, and 7 discuss, respectively, the results of the *bug trigger analysis, temporal analysis, complexity analysis*, and *impact analysis*; Section 8