On Failure Classification: The Impact of "Getting it Wrong"

Davide Falessi
Fraunhofer CESE
College Park, MD USA
dfalessi@fc-md.umd.edu

Bill Kidwell
University of Kentucky
Lexington, KY USA
bill.kidwell@uky.edu

Jane Huffman Hayes
University of Kentucky
Lexington, KY USA
hayes@cs.uky.edu

Forrest Shull
Software Engineering Institute
Arlington, VA USA
fjshull@sei.cmu.edu

ABSTRACT
Bug classification is a well-established practice which supports important activities such as enhancing verification and validation (V&V) efficiency and effectiveness. The state of the practice is manual and hence classification errors occur. This paper investigates the sensitivity of the value of bug classification (specifically, failure type classification) to its error rate; i.e., the degree to which misclassified historic bugs decrease the V&V effectiveness (i.e., the ability to find bugs of a failure type of interest). Results from the analysis of an industrial database of more than 3,000 bugs show that the impact of classification error rate on V&V effectiveness significantly varies with failure type. Specifically, there are failure types for which a 5% classification error can decrease the ability to find them by 66%. Conversely, there are failure types for which the V&V effectiveness is robust to very high error rates. These results show the utility of future research aimed at: 1) providing better tool support for decreasing human errors in classifying the failure type of bugs, 2) providing more robust approaches for the selection of V&V techniques, and 3) including robustness as an important criterion when evaluating technologies.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics-Product Metrics

General Terms
Experimentation, Measurement

Keywords
Bug classification, software quality, testing, metrics, verification and validation, human factor

1. INTRODUCTION
A best practice and well-established activity in industry is to classify bugs according to some accepted (and possibly company-specific) taxonomy or schema. The resulting bug repository, if mined, supports several important V&V activities such as test planning, root cause analysis, process improvement planning, bug location prediction, and resource allocation [1][2]. We are motivated by our work in an industrial context in which the company under study has several V&V techniques from which to choose including peer review, user acceptance testing, and automated unit testing. In this setting, significant effort is spent in creating, maintaining, and using a bug classification schema. The resulting bug repository is used to choose the V&V technique to be applied by adopting the criterion of maximizing the number of bugs of certain important failure types. This paper regards the *failure type* aspect of a bug to be the incorrect software system behavior that was observed as a result of the execution of the software *bug* (error in the code that led to the failure). In this scenario, errors in the bug repository can lead to the selection of inappropriate V&V techniques. We strive to maximize the *effectiveness of the V&V strategy* in this context; in other words, we aim to maximize the number of bugs found, of the specific failure type of interest, by choosing the V&V technique according to historical detection rates. Just as ultra-high reliability developers wonder about the fault tolerance of their software, we wonder how tolerant our V&V technique selection approach is to failure misclassifications.

Focusing the V&V strategy on finding specific failure types is an important topic. Specifically, certain failure types might be highly undesirable to the customers or more likely than others given the type of release. For instance, in a release where implementation focused on changing the user interface, failures related to formatting are more likely than failures related to accessibility or security. In software organizations with mature development processes it becomes important to use bug classification to find bugs of specific failure types, or that are difficult to find (perhaps applying V&V techniques that have "Hard Power" as opposed to "Broad Power" [3]).

The present paper is, to our knowledge, the first investigating the value of bug classifications (specifically, based on failure type) for maximizing V&V effectiveness, by examining how sensitive that value is with respect to failure misclassifications. Understanding how robust such strategies can be to errors in the underlying data set is especially important given the increasing reliance on software analytics; it is not useful to develop and deploy increasingly sensitive data mining techniques if small errors in the underlying data can lead to radically wrong conclusions.

2. RELATED WORK
The general problem of the low quality of data in empirical software engineering has been already stressed [4][5]. The present paper sits between two research directions: defect classification and evaluating bug-fix datasets for bias. Defect classification has been investigated in past studies [6][7][8][9], where the *bug classification error rate* of specific subjects under specific circumstances was reported and enhanced. However, so far it is unclear the extent to which a specific error rate (e.g., 10%) decreases the *value* of the bug repository and for which uses. Bias in bug-fix datasets has been studied in some detail [10][11][12] [13][14] [15]. The main differences between these studies and the present one include the type of investigated correctness (i.e., failure type classification vs. bug-fix link) and the use of the bug repository (V&V technique selection vs. bug localization estimation). Moreover, all but one of these studies analyzed only open source projects whereas we analyze industrial data. Thus,

ICSE'14, May 31 – June 7, 2014, Hyderabad, India.

this paper differs to these studies in all dependent and independent variables, and type of adopted dataset.

3. METHODOLOGY

In this section we describe the design of our study, including the dataset, the variables, and data analysis.

3.1 Dataset

The study reported in this paper analyzes an industrial database of over 3,000 bugs spanning five years and multiple projects at a medium sized, CMMI Maturity Level 5, company called Keymind. Keymind is the technology and creative services division of Axiom Resource Management, a professional consulting firm.

At Keymind, the failure that led to detection of the bug can be classified via one, and only one, of the following *nine types*: Accessibility, Incorrect Formatting, Missing Incorrect or Incomplete Functionality/Results, Non-Compliance of Program Behavior to Standard, Low Performance, Security/Vulnerability, Unexpected Termination, Usability, Other. Because the rate of occurrence of these bug types is company-sensitive, the bug types are referred to only with a random letter, as Type A through Type J, in the remainder of this paper.

Bugs can be found in one, and only one, of the following *five phases*: Development, Internal System Testing, User Acceptance Testing, Production, Other. Bugs can be found via one and only one of the following *11 V&V techniques*: Accessibility/Compliance Testing, Automated UI Testing, Internal System Testing, Manual Testing, Product Usage, Usability Testing, User Acceptance Testing, Peer Review, Mixed, Security Testing, Other.

At Keymind, all categories are mutually exclusive. Moreover, all aspects have an "Other" category: its use is deprecated, but it is still useful if the bug classifier is undecided.

Figure 1 summarizes the adopted bug repository, i.e., which failure types have been found by each V&V technique, on average. Figure 1 shows that it is important to choose the right V&V technique in order to find bugs exhibiting a specific failure type. For instance, if bugs of failure type C are of interest, six different V&V techniques can be chosen. Per Figure 1, their effectiveness in finding bugs of that type ranges between 1% (i.e., Automated UI Testing) and 42% (i.e., Usability Testing).

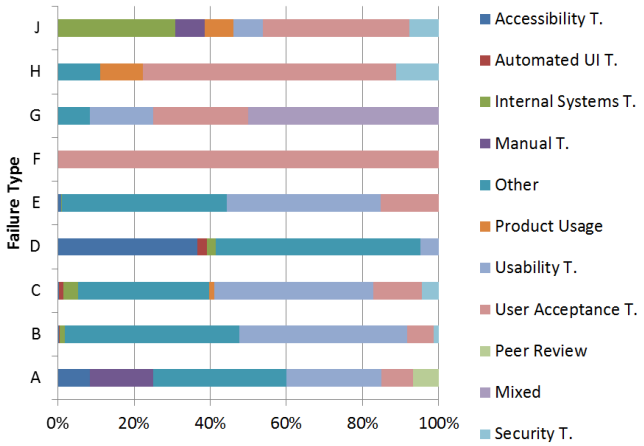

Figure 1: Bug profile for each of the different V&V techniques.

3.2 Variables

In this paper we focus on selecting V&V techniques that maximize the detection of bugs of a given failure type. For a given failure type, we define *Effectiveness* as the percentage of bugs that are found when choosing the optimal V&V technique according to the bug repository. For instance, according to Figure 1 if the failure type of interest is J, the effectiveness is 38% based on the application of the optimal technique, User Acceptance Testing.

Human errors during failure classification can result in an erroneous bug repository leading to the selection of a technique that does not actually find the most bugs of the failure type of interest and hence may result in some decreased V&V effectiveness. Thus for a given failure type, we define *EffectivenessWithError* as the percentage of bugs with the failure type of interest that are found when choosing a V&V technique according to a bug repository with a specific error rate. For instance, according to Figure 1, if the failure type of interest is J and, due to misclassifications in the historical data, Manual Testing is selected instead of User Acceptance Testing, then EffectivenessWithError is 8%.

Finally, for a given failure type and error rate, we define *Loss* as the percentage of bugs, of the failure type of interest, that are lost when choosing the V&V technique according to a bug repository with a given error rate; versus choosing the technique according to a correct repository. Loss is computed as: (Effectiveness - EffectivenessWithError) / Effectiveness. A Loss of 50% means that half of the bugs, of the failure type of interest, could have been found but were not found due to selection of the wrong V&V technique. For instance (see failure Type J in Figure 1), if the effectiveness decreases from 38% to 8% due to errors in the bug repository, then the Loss is 79%. Loss is zero if Error is zero. Moreover, if the optimal V&V technique is selected, despite there being misclassified bugs, the loss is 0%. Loss is our *dependent variable* and it reliably describes the impact of misclassifications on the number of bugs found of the failure type of interest.

We note that, given the definition of Loss, the impact of Error on Loss depends on the degree of difference among V&V technique detection rates. For a given failure type, if all feasible techniques have the same detection rate then the Loss is zero regardless of failure classification error. Counter-intuitively, the number of bugs of a specific failure type does not influence anything, mainly because Loss is expressed as a percentage. Thus, our *independent variables* are failure type, phase, and error rate.

3.3 Data analysis

Our data analysis procedure consists of three main steps: preprocessing, error injection, and computation of Loss.

Preprocessing. The 3,000+ bugs were pre-processed to facilitate analysis and anonymize data. Moreover, unfeasible testing techniques and failure types were removed from the analysis in specific phases. For instance, Product usage and Formatting are a technique and a failure type, respectively, which are unfeasible in the Development phase. Table 1 reports the number of feasible V&V techniques and failure types for each of the five phases.

Table 1: Number of feasible V&V techniques and failure types for each specific phase

	Phase				
	1	2	3	4	5
Number of feasible V&V techniques	8	9	3	9	8
Number of feasible failure types	6	7	6	9	8

Error injection algorithm. The industrial bug repository is used as the gold standard and random failure misclassifications have been applied to it. We used the following error rates: 0%, 5%, 10%, 25%, 50%, 75%, 90%, 95%, and 100%. The error rate describes the percentage of misclassified failure types. Because phase is a factor (see Preprocessing), each error rate was applied to one phase at a time. Because preliminary data analysis and past studies [6] [7] [8] [9] show the absence of correlation among failure types, the failure misclassification was randomly chosen from among the other feasible failure types within that phase. Because specific misclassifications could threaten results validity, we repeated the error injection algorithm 10,000 times, for each phase and for each error rate.

Computation of Loss. For each phase, once a specific error rate was introduced, the V&V technique with the highest effectiveness was chosen and its effectiveness was computed as the number of bugs found of the failure type of interest. This value is the EffectivenessWithError and is used to compute the Loss, as described in Section 3.2. Afterwards, the value of Loss, for a specific failure type, phase, and error rate, was computed by averaging the values of the 10,000 runs.

We implemented this data analysis in a script in R that is available for download: https://github.com/bill-kidwell-uk/GettingItWrong.

4. RESULTS AND DISCUSSION

Figure 2 reports, for each failure type, the average Loss over all relevant phases, over the spectrum of Error rates. According to Figure 2, the impact of Error Rate on Loss clearly depends on the specific failure type of interest. For instance, the Loss for failure type E is 0% even in the case of 100% erroneous classification. This is probably because the feasible techniques for type E have the same detection rates. Vice versa, the Loss for failure type J is 66% when the error rate is only 5%. We note that according to past studies [6][7][8][9], the failure classification error is likely more than 5%. Thus, in a realistic context, bugs of certain failure types are likely very hard to detect, even when choosing the V&V technique that historically caught most of them.

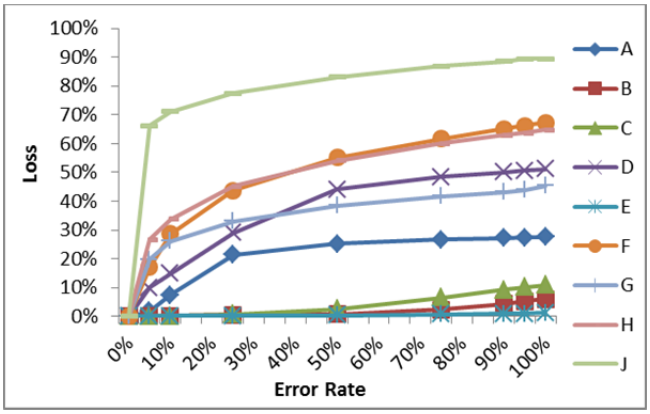

Figure 2: Percentage of bugs lost (y-axis) when choosing the V&V technique in the presence of a specific percentage of misclassifications (x-axis), on average across phases.

Because we have an average of seven failure types per phase (see Table 1), then a purely random failure classification approach would lead to an error rate of 86% (i.e., 6/7). The Loss for an Error rate of 86% is more than 50% for four failure types. Thus, it is important to classify failure types to be able to find the bugs of the failure type of interest.

According to prior work [6] [7] [8] [9], the error rate of humans in classifying failure types under realistic circumstances is around 10%. Moreover, the impact of Error on Loss, for a given failure type, depends on the distribution of the detection rate of the V&V techniques that are feasible in a given phase. Thus it is interesting to see how a realistic classification error rate (10%) impacts Loss in specific phases. Figure 3 reports the Loss of the Error rate 10%, in each specific phase for each failure type. According to Figure 3, there are some phases (i.e., Phase 1 and 5) where the Loss is very low for all failure types. Vice versa, there are phases such as 2 and 5 where a realistic error rate makes it impossible to find bugs of a specific failure type (i.e., Failure type J).

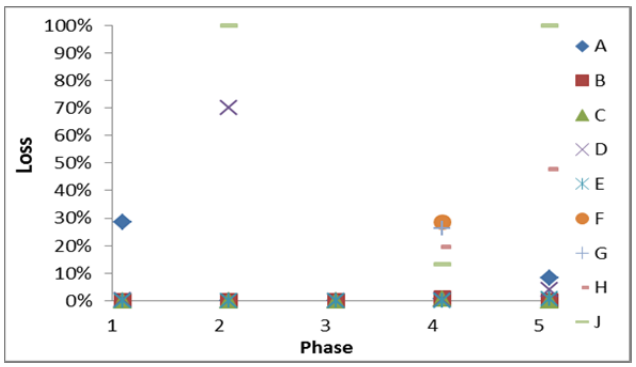

Figure 3: Percentage of bugs lost (y-axis) across the different phases (x-axis) when choosing the V&V technique by having a realistic error rate (i.e., 10%).

We statistically tested the impact of Error, Phase, and Failure types on Loss. Table 2 reports the p-value of a 3-way ANOVA test. Results from Table 2 show that the Loss significantly depends on all three variables and also on their interaction. The interaction factor means that the impact of Error on Loss depends on the specific combination of Failure type and Phase. For instance, by analyzing Figure 3, Loss is high for Failure type D only in Phase 2.

Table 2: Statistical test results on the impact on Loss of Phase, Failure type, Error rate, and their combination

	F Ratio	p-value
Phase	13.0766	<.0001
Error	40.9353	<.0001
Failure type	45.1572	<.0001
Phase * Error* Failure type	2.3184	0.0005

4.1 Threats to validity

Threats to our study fall into four main categories: internal, external, construct, and conclusion.

Internal. Our study uses the 3,000 bug reports as the gold standard (we assume that the failure types of bug classifications are correct). Although there is evidence of the low quality of *open source* projects [10][11], we justify this assumption based on a preliminary analysis of the dataset showing a very low value of error. Thus, the use of an industrial dataset, rated CMMI Maturity Level 5, is the best available dataset for our use.

Construct. We assumed the absence of correlation across failure types, i.e., that the error in a failure classification is random. We

justify this assumption given the absence of any available data about correlation among failure types [6][7][8][9] and according to a preliminary analysis of the dataset. In studies of this type, a further threat to construct validity is that of experimenter expectation, i.e., researchers wanting the study to have a certain outcome may have biased the study to produce that outcome. To address this threat, we made conservative decisions which in turn threaten conclusion validity.

External. Our study has a threat to external validity as we only examined one set of bug reports from one company. However: 1) the bug reports do represent multiple projects across multiple domains, and 2) the specific failure types, V&V techniques, and phases are reported (see Section 2.1) and hence their applicability to external contexts can be analyzed.

Conclusion. In general, when trying to tradeoff the different threats to validity we chose to be conservative. For instance, during error injection, for each specific phase, we considered only feasible V&V techniques and failure types (see Table 1). Thus, the main threat to conclusion validity is that the Loss could actually be more sensitive to Error than what current results show. This would actually make the case supporting correct failure classification stronger. Finally, mistakes in data analysis have been avoided by inspecting the script and making it available.

5. CONCLUSIONS AND FUTURE WORK

Our results show that the impact of classification error on V&V effectiveness significantly varies with failure type. Most importantly, there are phases where even a very low classification error rate (close to what we expect in practice) makes finding bugs of a specific failure type very hard, even when choosing the technique with the expected highest detection rate for that type of bug. Clearly, it is not useful to develop and deploy increasingly sensitive data mining techniques if small errors in the underlying data can lead to radically wrong conclusions. We need to adopt technologies that work well with data of realistic quality.

From an empirical perspective, technologies, including defect prediction models or information retrieval techniques, should be evaluated not only according to their accuracy but also according to their sensitivity to the quality of the underlying data. Technologies that are robust to inaccuracies in the data may, in some contexts, provide more benefits than technologies that are highly accurate but only under ideal conditions.

From a technology development perspective, technologies should provide as output not only one value (i.e., the value produced by considering the underlying data as completely correct) but also a related "confidence interval" which takes into account its sensitivity to realistic correctness of the underlying data.

From a research perspective, there is no shortage of prior work that has developed technologies for different uses of the bug repositories for supporting decisions during software development. We suggest that promising areas of future work include: 1) investigating how different uses of bug repositories (e.g., V&V technique selection vs. bug localization prediction) are sensitive to the quality of the underlying data, 2) evaluating what factors can influence sensitivity (e.g., phases or failure type), 3) studying how to improve data quality (e.g., better bug classification schema), and 4) investigating mechanisms to make technologies more robust to data quality.

6. REFERENCES

[1] B. Freimut, "Developing and using defects classification schema." Fraunhofer IESE, IESE-report No. 072.01/E, 2001.

[2] M. Felderer and A. Beer, "Using Defect Taxonomies to Improve the Maturity of the System Test Process: Results from an Industrial Case Study," vol. 133, D. Winkler, S. Biffl, and J. Bergsmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[3] L. Miller, S. Mirsky, and J. H. Hayes, "Guidelines for the Verification and Validation of Expert System Software and Conventional Software," 1995.

[4] A. Mockus, "Missing Data in Software Engineering," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. London: Springer London, 2008.

[5] G. A. Liebchen and M. Shepperd, "Data sets and data quality in software engineering," in *Proceedings of the 4th international workshop on Predictor models in software engineering - PROMISE '08*, 2008, p. 39.

[6] D. Falessi and G. Cantone, "Exploring Feasibility of Software Defects Orthogonal Classification," in *Software and Data Technologies*, vol. 10, J. Filipe, B. Shishkov, and M. Helfert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 136–152.

[7] A. Vetro, N. Zazworka, C. Seaman, and F. Shull, "Using the ISO/IEC 9126 product quality model to classify defects : a controlled experiment," in *16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, 2012, pp. 187–196.

[8] K. El Emam and I. Wieczorek, "The repeatability of code defect classifications," in *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No.98TB100257)*, 1998, pp. 322–333.

[9] K. Henningsson and C. Wohlin, "Assuring fault classification agreement - an empirical evaluation," in *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04.*, 2004, pp. 95–104.

[10] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?," in *Proceedings of the 2008 conference of the center for advanced studies on collaborative research meeting of minds - CASCON '08*, 2008, p. 304.

[11] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein, "The missing links," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering - FSE '10*, 2010, p. 97.

[12] K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: how misclassification impacts bug prediction," in *2013 International Conference on Software Engineering (ICSE '13)*, 2013, pp. 392–401.

[13] F. Rahman, D. Posnett, I. Herraiz, and P. Devanbu, "Sample size vs. bias in defect prediction," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013*, 2013, p. 147.

[14] T. H. D. Nguyen, B. Adams, and A. E. Hassan, "A Case Study of Bias in Bug-Fix Datasets," in *2010 17th Working Conference on Reverse Engineering*, 2010, pp. 259–268.

[15] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, 2011, p. 481.