Improving Bug Reporting, Duplicate Detection, and Localization

Oscar Chaparro
The University of Texas at Dallas
ojchaparroa@utdallas.edu

***Abstract*—Software developers rely on essential textual information from bug reports (such as Observed Behavior, Expected Behavior, and Steps to Reproduce) to triage and fix software bugs. Unfortunately, while relevant and useful, this information is often missing, incomplete, superficial, ambiguous, or complex to follow. Low-quality content in bug reports causes delay and extra effort on bug triage and fixing. Current technology and research are insufficient to support users and developers on providing high-quality content in bug reports. Our research is intended to fill in this gap, as it aims at improving: (1) the quality of natural language content in bug reports, and (2) the accuracy of Text Retrieval (TR)-based bug localization and duplicate detection. To achieve such goals, our research will identify, enforce, and leverage the discourse that reporters use to describe software bugs.**

***Index Terms*—Discourse Analysis, Bug Reporting, Bug Localization, Duplicate Bug Report Detection**

I. Problem and Research Statement

Bug reports are meant to collect relevant information about the bugs that users find when using software. The information provided in such reports is intended to help developers on diagnosing and removing software bugs [1]. While bug reports contain structured bug data such as operating system, system version or attachments (*e.g.*, stack traces, files, and code), they mostly contain unstructured bug information in natural language form [1], [2], [3]. Unstructured natural language content produced by reporters includes the description of software (mis)behavior (*i.e.*, Observed Behavior or OB), the steps to reproduce the (mis)behavior (*i.e.*, Steps to Reproduce or S2R), and the normal software behavior (*i.e.*, Expected Behavior or EB). Developers identify such content among the most useful information when triaging and fixing bugs [1].

Unfortunately, while considered useful, OB, EB, and S2R content is not always found in bug reports, and when it is, its quality is often low. In fact, our preliminary analysis of nearly 3K bug reports from nine software systems indicates that, while most of the bug reports contain OB (*i.e.*, 93.5%), only 35.2% and 51.4% of the reports explicitly describe EB and S2R, respectively. (see section II-A). In addition, numerous researchers found that textual descriptions in bug reports are often incomplete, superficial, ambiguous, or complex to follow (*i.e.*, bug descriptions contain low-quality content) [1], [4], [5], [6]. The problem is also acknowledged by software engineering professionals. Recently, developers from more than a thousand open source projects signed and sent a petition to GitHub remarking that "issues are often filed missing crucial information like reproduction steps and version tested" [7].

Low-quality content in bug reports impacts negatively software users and developers in many ways. Low-quality content is one of the main reasons for non-reproduced bugs [6], non-fixed bugs [5], and additional bug triage effort [8], as developers have to spend more time and effort understanding bug descriptions or asking for clarifications and additional information [6], [8]. Low-quality bug reports are also likely to gain low attention by developers [9]. In addition, they lead to low-quality queries that cause Text Retrieval (TR)-based approaches for bug localization [10], [11] and bug duplicate detection [12] to have low accuracy. As indicated by developers, absent and wrong information in bug reports are the main causes for delay on bug fixing [1].

One the main reasons for low-quality content in bug reports is insufficient tool support for bug reporting [1], [6], [7]. In the aforementioned GitHub petition [7], developers call for improvements to GitHub's technology to ensure that essential information is reported by users. This problem extends to other bug tracking systems. Most of these systems capture unstructured natural language bug descriptions using web forms without any content verification or enforcement. A handful of bug trackers (for particular projects, *e.g.,* Bugzilla in the Mozilla Firefox project) provide semi-structured reporting of natural language information, in the form of predefined text templates that explicitly ask for OB, EB, and S2R. This solution is insufficient to address the problem as it does not guarantee that users will provide high-quality content if provided at all.

Very little research has been done to support users on bug reporting and improve the natural language content in bug reports. Zimmerman *et al.* [1] propose an approach that predicts the quality level of bug reports. However, this approach does not give actionable recommendations to users on how to improve their bug descriptions. Moran *et al.* [4] developed a technique to augment the steps to reproduce in bug reports via screenshots and GUI-component images. This approach does not focus on improving the textual content of bug descriptions. More recently, the same authors proposed an approach that finds, reproduces, and reports potential crashes in mobile applications without any user intervention [13]. However, this approach is not intended to support users when they textually describe software bugs. Despite these few attempts to support users on bug reporting, the fact remains that *bug reports have* *low-quality textual content and there is limited tool support on bug reporting to improve their quality.*

In this dissertation, we aim at improving the textual content of bug reports by (1) enforcing the presence of OB, EB, and S2R, and (2) recommending discourse elements to describe such content in bug reports. To accomplish this goal, we will design, implement, and evaluate novel techniques that will leverage existent natural language discourse in bug reports and techniques in Natural Language Processing (NLP) and Machine Learning (ML). *By identifying, enforcing, and leveraging Observed Behavior, Expected Behavior, and Steps to Reproduce discourse, bug reports will have higher quality, and TR-based bug localization and duplicate detection will improve.* The remaining sections describe in detail our research plan and anticipated contributions.

II. Proposed Research

The goal of our research is twofold: improve the quality of natural language content in bug reports, and improve the accuracy of TR-based bug localization and duplicate detection. We will achieve our goals by identifying, enforcing, and leveraging the Observed Behavior (OB), Expected Behavior (EB), and Steps to Reproduce (S2R) discourse used in bug descriptions.

Our research goals impose a set of research challenges that we plan to overcome in this dissertation:

A. Determining the OB, EB, and S2R discourse that reporters use when describing software bugs.

B. Automatically identifying and enforcing such discourse in new bug reports.

C. Recommending discourse elements (*e.g.*, terms or sentences) to reporters to improve OB, EB, and S2R descriptions in bug reports.

D. Improving TR-based bug localization and duplicate detection approaches using the OB, EB, and S2R discourse used in bug reports.

The next sections describe in detail our research plan and preliminary results.

A. Determining the Discourse Used in Bug Descriptions

The first major challenge of this research is determining and understanding the discourse used by reporters to describe OB, EB, and S2R in bug reports. Our hypothesis is that users follow a restricted discourse to describe their problems in bug reports, hence it can be leveraged to improve many tasks on bug triage and bug fixing. To test our hypothesis, we conducted an empirical study using a grounded theory-based approach to discover discourse patterns that capture the syntax and semantics of sentences and paragraphs in bug descriptions.

We collected a random sample of 2,912 bug reports from nine software projects from different types and domains[1]. These projects use different issue trackers (*e.g.*, Jira or GitHub). We extracted the text from the bug reports (*i.e.*, title and description), used heuristics to automatically parse the text into paragraphs and sentences, and split the data into two data sets. The first data set, composed of 1,091 bug reports, was used to extract the discourse patterns, and the second data set, composed of the remaining 1,821 bug reports, was used for validation purposes. Once the data was collected and pre-processed, the next step was to code the textual content to discover OB, EB, and S2R discourse patterns. A discourse pattern is a rule that structures a sentence or paragraph to convey either OB, EB, or S2R[2]. Following a grounded theory methodology, five Ph.D. students manually tagged the sentences/paragraphs that capture OB, EB, or S2R with codes that represent discourse patterns. The process included creating new codes or reusing existing ones to tag each sentence/paragraph that encoded at least one of the three types of information. The coding process was fully iterative and included validation among coders of those cases when it was not clear what pattern a particular phrase followed.

The coding process produced a set of 154 discourse patterns. Most of these correspond to sentence-level patterns (*i.e.*, 135), and OB patterns (*i.e.*, 90). A large proportion of the identified S2R patterns are paragraph-level (*i.e.*, 13 out 33), which is expected as sequences of sentences are more suitable to describe steps to reproduce. The number of EB and S2R patterns (*i.e.*, 31 and 33, respectively) is relatively low compared to the number of OB patterns. This means that reporters use narrow discourse to describe the expected behavior and steps to reproduce. In the case of OB, users tend to use wider and more variable (yet narrow) discourse to describe different software misbehavior, compared to expected behavior and steps to reproduce. Our analysis also revealed that while most of the bug reports in our data set contain OB (*i.e.*, 2,724 bug reports - 93.5%), only 1,024 descriptions contain EB (*i.e.*, 35.2%), and 1,498 contain S2R (*i.e.*, 51.4%). The results provide evidence of the lack of essential information in bug reports, fact that contributes to low-quality content in bug reports. The results of our study confirm our hypothesis: *the discourse that reporters use to describe software bugs is restricted to a well-defined set of discourse patterns.*

Different from our research, existing work has mainly focused on analyzing linguistic properties of bug report titles [14], identifying frequently asked questions [8], and studying the structure of bug reports [1], [2], [3]. Other work has focused on identifying linguistic patterns from development conversations in e-mails [15]. None of these works identify the OB, EB, and S2R discourse used in bug descriptions.

B. Detecting Missing Information in Bug Reports

Our research plan is intended to provide actionable feedback to software users when they describe software bugs. One way to do so is by detecting and enforcing OB, EB, and S2R information in bug reports. Given the narrow discourse used to describe such information, our hypothesis is that it is possible

[1]Docker, Eclipse, Facebook, Firefox, Hibernate, Httpd, LibreOffice, OpenMRS, and Wordpress-Android.

[2]For example, the OB pattern "[subj] [neg aux verb] [verb] [compl]" corresponds to negative sentences with auxiliary verbs such as "[*The icon*] [***did not***] [*change*] [*to an hourglass*]".

to accurately predict the presence (or absence) of OB, EB, and S2R in bug reports. In other words, we can alert reporters when they miss such information when writing bug descriptions.

We designed two approaches that predict the absence of EB and S2R (as bug reports are more prone to miss such information). Our first approach is heuristic-based and relies on automated part-of-speech and syntactic dependencies parsing to detect if a bug report matches any of the identified discourse patterns. We implemented each discourse pattern using the Stanford CoreNLP toolkit at sentence- and paragraph-level. Our approach flags a new bug report as containing (or not) EB or S2R if the report contains any (or no) sentence or paragraph that matches any of the corresponding discourse patterns. Our second approach is Machine Learning-based which relies on different textual features. We trained and tested two Support Vector Machines (SVMs) for predicting the absence of EB and S2R, respectively. We used three sets of features to train the SVMs, namely, POS tags, n-grams (*i.e.*, {1, 2, 3}-grams), and the individual discourse patterns (using the implementation from the heuristics-based approach).

We conducted an empirical evaluation to determine the predictive accuracy of our approaches. Our discourse study produced the ground truth data set, *i.e.*, bug reports correctly labeled as containing (or not) EB or S2R. We executed our heuristics-based approach on the 1,821 bug reports that were not used to extract the discourse patterns, and we trained/tested both SVMs using 10-fold cross validation on the same data. We assessed the performance of the SVMs by using each set of features (*i.e.*, POS tags, n-grams, and discourse patterns) and their combination. We report the performance of the most accurate approach for both EB and S2R. The heuristics-based approach is the most accurate for predicting EB, as it achieves 84.4% precision and 78.7% recall. The SVM based on all features (including the identified discourse patterns) is the most accurate for predicting S2R, as it achieves 74.3% precision and 80.1% recall. These results provide evidence to our hypothesis: *it is possible to accurately predict the absence of EB and S2R in bug reports*. Future work will focus on comparing our approaches with other techniques (*e.g.*, language models) and conducting empirical studies with end-users to determine their usefulness.

Our research relates to work on bug report classification [9], [16], [17], which relies on machine learning and textual features to classify new bug reports as features requests, enhancements, or bug descriptions. Similar approaches have been proposed to classify e-mails [15], app reviews [18] and forums [19]. The essential difference between our (SVM-based) approaches and existing software content classifiers is the use of discourse patterns from bug descriptions. Closer to the problem we are solving, Davies *et al.* [2] propose to use explicit search terms (*e.g.*, *"observed behavior"*) to find OB, EB, or S2R content in bug reports. Unfortunately, while simple and straightforward, this approach produces an excessive number of false negatives. Finally, similar to our heuristics-based approach, Sorbo *et al.* [15] use linguistic heuristics to detect various content in software e-mail conversations.

C. Recommending Common Bug Discourse Elements

The next challenge that we plan to address in this research is detecting unusual discourse in bug descriptions and recommending terminology that reflects common discourse to describe OB, EB, and S2R. Our goal is to give granular feedback, at sentence/paragraph level, to reporters when they describe software bugs. We will rely on the discourse patterns identified and how frequently they are found in our bug reports corpus. We will design and evaluate an approach that will use transformation rules from least- to most-common discourse patterns. Our approach will be based on the decomposition of the identified discourse patterns into atomic discourse elements. The transformation rules will define mappings between components from infrequent to frequent discourse patterns. Alternatively, atomic elements of discourse can serve to determine the matching degree of a sentence to a discourse pattern. For example, if a sentence matches four out of five elements for a particular pattern (*i.e.*, the sentence matches 80% of the pattern), then our approach will suggest terms to match the remaining discourse element.

We will conduct empirical studies to determine the recommendation accuracy and usefulness of our approach. We will develop a tool implementing our approach, which will be deployed in open source projects. We will observe users' response and behavior out tool's predictions and recommendations. Our tool will record usage information such as clicks, term selections, and omissions to our recommendations, as well as, user input such as rewrites to our recommendations. We will use surveys to get feedback from the users about the usefulness of our tool and the quality of the final bug reports. We will also use metrics to estimate the readability and other properties of the resulting descriptions [1].

The closest related work to our research is Moran *et al.*'s approach [4], which relies on imperative templates to automatically generate lists of GUI actions (*i.e.*, steps to reproduce) in mobile applications whenever the approach finds application crashes. Different from our focus, other research has been done to summarize bug reports conversations [20], [21], [22]. To the best of our knowledge, no previous work has focused on detecting uncommon discourse in bug reports and suggesting common wording to describe OB, EB, and S2R.

D. Improving Text Retrieval-based Bug Localization and Duplicate Detection

Our final research challenge is improving TR-based bug localization and duplicate detection. TR-based approaches rely on the vocabulary agreement and textual similarity between the queries, created from the full textual content of bug reports, and the source code [23] or past bug reports [12]. One of the main problems of these approaches is the high variability of their accuracy to different quality levels of the input queries [10], [11]. Our earlier work [10] revealed that it is possible to achieve substantial accuracy improvement on TR-based bug localization by removing only one term from the queries. Motivated by this finding, our hypothesis is that specific content from bug reports (used to create the queries) will improve TR-based bug localization and duplicate detection. For instance, since bug reports likely describe the reported problem in the observed behavior content, we can design a duplicate detector that uses the text corresponding to observed behavior as queries. Our goal is defining, implementing, and evaluating an approach that combines different types of content that exhibit specific discourse patterns, to improve the accuracy of TR-based bug localization and duplicate detection. We will conduct standard empirical evaluations with existing and new data, and well-known evaluation metrics [10], [24].

Previous work in TR-based bug localization and duplicate detection has focused on using the whole text of bug reports as queries [10], [24]. Some works have focused on using specific terms from the bug reports, such as nouns [25]. Other works have focused on encoding bug report textual content using word embedding [26] or n-grams [27]. To best of our knowledge, this is the first proposal on exploiting different types of textual information and discourse in bug reports to improve TR-based bug localization and duplicate detection.

III. Anticipated Contributions

Our research proposal is intended to improve the quality of bug reports content and the accuracy of TR-based bug localization and duplicate detection. Our proposed solutions will provide actionable feedback to software users when describing the Observed Behavior, Expected Behavior, and Steps to Reproduce in bug reports. This information and the discourse used in bug reports will be leveraged to further support developers on localizing the code that needs to be changed in response to bugs and on detecting duplicate bug reports to accelerate bug triage. Our contributions include a set of the novel techniques that will support software stakeholders on bug reporting, localization, and duplicate detection. These techniques will be implemented as usable tools that will be delivered to the community as open source software. We expect to produce software data sets containing bug reports, source code, and information inferred from these sources, which will be open to the community for verifiability and further research in this area. The actionable nature of our solutions will allow users to report OB, EB, and S2R content that is clear, explicit, less ambiguous, and following common discourse to describe bugs. Our solutions will also lead to more accurate and reliable bug localization and duplicate detection, by leveraging different bug report content and discourse, *i.e.*, sources of information currently ignored by existing research in these areas [10].

References

[1] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What Makes a Good Bug Report?," *IEEE Trans. on Soft. Eng.*, vol. 36, no. 5, pp. 618–643, 2010.

[2] S. Davies and M. Roper, "What's in a Bug Report?," in *Proc. of the Int. Symp. on Emp. Soft. Eng. and Meas.*, pp. 26:1–26:10, 2014.

[3] T. D. Sasso, A. Mocci, and M. Lanza, "What Makes a Satisficing Bug Report?," in *Proc. of the Int. Conf. on Soft. Quality, Reliability and Security*, pp. 164–174, 2016.

[4] K. Moran, M. Linares-Vásquez, C. Bernal-Cárdenas, and D. Poshyvanyk, "Auto-completing Bug Reports for Android Applications," in *Proc. of the Joint Meeting on Found. of Soft. Eng.*, pp. 673–686, 2015.

[5] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy, "Characterizing and predicting which bugs get reopened," in *Proc. of the Int. Conf. on Soft. Eng.*, pp. 1074–1083, 2012.

[6] M. Erfani Joorabchi, M. Mirzaaghaei, and A. Mesbah, "Works for Me! Characterizing Non-reproducible Bug Reports," in *Proc. of the Working Conf. on Mining Soft. Repositories*, pp. 62–71, 2014.

[7] "An open letter to GitHub from the maintainers of open source projects." Available online: https://github.com/dear-github/dear-github, 2016.

[8] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information Needs in Bug Reports: Improving Cooperation Between Developers and Users," in *Proc. of the Conf. on Computer Supported Cooperative Work*, pp. 301–310, 2010.

[9] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, "Characterizing and predicting which bugs get fixed: an empirical study of Microsoft Windows," in *Proc. of the Int. Conf. on Soft. Eng.*, pp. 495–504, 2010.

[10] O. Chaparro and A. Marcus, "On the Reduction of Verbose Queries in Text Retrieval Based Software Maintenance," in *Proc. of the 38th Int. Conf. on Soft. Eng.*, pp. 716–718, 2016.

[11] S. Haiduc, G. Bavota, A. Marcus, R. Oliveto, A. De Lucia, and T. Menzies, "Automatic Query Reformulations for Text Retrieval in Software Engineering," in *Proc. of the 2013 Int. Conf. on Soft. Eng.*, pp. 842–851, 2013.

[12] O. Chaparro, J. M. Florez, and A. Marcus, "On the Vocabulary Agreement in Software Issue Descriptions," in *Proc. of the Int. Conf. on Soft. Maintenance and Evolution*, pp. 448–452, 2016.

[13] K. Moran, M. Linares-Váquez, C. Bernal-Cárdenas, C. Vendome, and D. Poshyvanyk, "Automatically Discovering, Reporting and Reproducing Android Application Crashes," in *Proc. of the Int. Conf. on Soft. Testing, Verification and Validation*, pp. 33–44, 2016.

[14] A. J. Ko, B. A. Myers, and D. H. Chau, "A Linguistic Analysis of How People Describe Software Problems," in *Visual Languages and Human-Centric Computing*, pp. 127–134, 2006.

[15] A. D. Sorbo, S. Panichella, C. A. Visaggio, M. D. Penta, G. Canfora, and H. C. Gall, "Development Emails Content Analyzer: Intention Mining in Developer Discussions (T)," in *Proc. of the Int. Conf. on Automated Soft. Eng.*, pp. 12–23, 2015.

[16] Y. Zhou, Y. Tong, R. Gu, and H. Gall, "Combining text mining and data mining for bug report classification," *Journal of Soft.: Evolution and Process*, vol. 28, no. 3, pp. 150–176, 2016.

[17] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is It a Bug or an Enhancement?: A Text-based Approach to Classify Change Requests," in *Proc. of the Conf. of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*, pp. 23:304–23:318, 2008.

[18] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A Survey of App Store Analysis for Software Engineering," 2016.

[19] B. Zhou, X. Xia, D. Lo, C. Tian, and X. Wang, "Towards More Accurate Content Categorization of API Discussions," in *Proc. of the Int. Conf. on Program Comprehension*, pp. 95–105, 2014.

[20] S. Rastkar, G. C. Murphy, and G. Murray, "Automatic Summarization of Bug Reports," *IEEE Trans. on Soft. Eng.*, vol. 40, no. 4, pp. 366–380, 2014.

[21] R. Lotufo, Z. Malik, and K. Czarnecki, "Modelling the 'hurried' bug report reading process to summarize bug reports," *Empirical Soft. Eng.*, vol. 20, no. 2, pp. 516–548, 2014.

[22] S. Mani, R. Catherine, V. S. Sinha, and A. Dubey, "AUSUM: Approach for Unsupervised Bug Report Summarization," in *Proc. of the Int. Symp. on the Found. of Soft. Eng.*, pp. 11:1–11:11, 2012.

[23] L. Moreno, W. Bandara, S. Haiduc, and A. Marcus, "On the Relationship between the Vocabulary of Bug Reports and Source Code," in *Proc. of the Int. Conf. on Soft. Maintenance*, pp. 452–455, 2013.

[24] S. Wang and D. Lo, "AmaLgam+: Composing Rich Information Sources for Accurate Bug Localization," *Journal of Soft.: Evolution and Process*, vol. 28, no. 10, pp. 921–942, 2016.

[25] R. Shokripour, J. Anvik, Z. M. Kasirun, and S. Zamani, "Why So Complicated? Simple Term Filtering and Weighting for Location-based Bug Report Assignment Recommendation," in *Proc. of the Working Conf. on Mining Soft. Repositories*, pp. 2–11, 2013.

[26] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, "From Word Embeddings to Document Similarities for Improved Information Retrieval in Software Engineering," in *Proc. of the Int. Conf. on Soft. Eng.*, pp. 404–415, 2016.

[27] A. Sureka and P. Jalote, "Detecting Duplicate Bug Report Using Character N-Gram-Based Features," in *Proc. of the Asia Pacific Soft. Eng. Conf.*, pp. 366–374, 2010.