Do Visual Issue Reports Help Developers Fix Bugs?

– A Preliminary Study of Using Videos and Images to Report Issues on GitHub –

Hiroki Kuramoto, Masanari Kondo, Yutaro Kashiwa, Yuta Ishimoto, Kaze Shindo, Yasutaka Kamei, Naoyasu Ubayashi
Principles of Software engineering and programming Languages Lab. (POSL), Kyushu University
Japan
{kuramoto,ishimoto,shindo}@posl.ait.kyushu-u.ac.jp,{kondo,kashiwa,kamei,ubayashi}@ait.kyushu-u.ac.jp

ABSTRACT

Issue reports are a pivotal interface between developers and users for receiving information about bugs in their products. In practice, issue reports often have incorrect information or insufficient information to enable bugs to be reproduced, and this has the effect of delaying the entire bug-fixing process. To facilitate their bug-reproduction work, GitHub has provided a new feature that allows users to share videos (e.g., mp4 files.) Using such videos, reports can be made to developers about the details of bugs by recording the symptoms, reproduction steps, and other important aspects of bug information.

While such visual issue reports have the potential to significantly improve the bug-fixing process, no studies have empirically examined this impact. In this paper, we conduct a preliminary study to identify the characteristics of visual issue reports by comparing them with non-visual issue reports.

We collect 1,230 videos and 18,760 images from 226,286 issues on 4,173 publicly available repositories. Our preliminary analysis shows that issue reports with images are described in fewer words than non-visual issue reports. In addition, we observe that most discussions in visual issue reports are concerned with either conditions for reproduction (e.g., when) or GUI (e.g., pageviewcontroller.)

CCS CONCEPTS

• **Software and its engineering → Maintaining software**.

KEYWORDS

GitHub, Issues, Videos, Images

ACM Reference Format:
Hiroki Kuramoto, Masanari Kondo, Yutaro Kashiwa, Yuta Ishimoto, Kaze Shindo, Yasutaka Kamei, Naoyasu Ubayashi. 2022. Do Visual Issue Reports Help Developers Fix Bugs?: – A Preliminary Study of Using Videos and Images to Report Issues on GitHub –. In *Proceedings of ICPC '22: Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension (ICPC 2022).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ICPC 2022, May 16–17, 2022, Pittsburgh, PA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The question "What makes a good issue report?" has been studied for decades and is still the ultimate research question for many studies aiming to improve the quality of issue reports [21][39][4]. Issue reports (a.k.a. bug reports) often lack the information necessary for developers to reproduce bugs [23][17]. For example, Zimmermann et al. [39] report that stack traces and steps for reproducing a bug are considered to be helpful by developers. But, it is difficult for users to provide this information, and it is often missing or incorrect. This mismatch between what developers need and what reporters can provide can often delay the fixing of bugs [23]. In addition, many studies have reported that the quality of issue reports impacts both the issue resolution time [7][18] and the issue resolution rate [40][38].

To facilitate developers' bug-reproduction work, GitHub launched a new feature that allows users to share videos (e.g., mp4 files) in May 2021 [9]. Using such videos, reports can be made to developers about the details of bugs by recording the symptoms, reproduction steps, and other important aspects of a comprehensive bug report. These visual images can help developers understand the nature of the bug, and what users were doing when the bug occurred. While such visual issue reports have the potential to improve the bug-fixing process, no studies have empirically examined this impact.

In this paper, we conduct a preliminary study to identify the characteristics of visual issue reports by comparing them with non-visual issue reports. In addition, we provide the dataset used in this study on a public repository[1], to promote future studies using visual issue reports. This dataset consists of videos and images in publicly available repositories on GitHub. Specifically, we collected 1,230 videos and 18,760 images from 226,286 issue reports on 4,173 publicly available repositories.

Our initial analysis reveals that (i) issue reports with images contain fewer words than non-visual issue reports; (ii) the number of comments and the first response time for visual issue reports are almost the same as for non-visual issue reports; and (iii) resolution time of visual issue reports is not significantly different from that of other issue reports.

2 STUDY DESIGN

2.1 Research Questions

To identify the characteristics of the visual issue reports, we addressed the following three research questions: focusing on Report (RQ1), Discussion (RQ2), and Fix (RQ3).

[1]https://doi.org/10.5281/zenodo.6071588

RQ1: **Do visual issue reports require less texts to report bugs than non-visual issue reports?**
Developers often find it difficult to reproduce bugs using the reported information [11][31][39]. On the other hand, as reporters are not always developers, it is not easy to tell what they did and what they encountered [10]. Thus, GitHub developed a feature that can easily provide information with videos and officially announced the feature release on May, 2021 [9]. Potter and Faulconer [30] showed that, in general, visual images are a more effective approach for describing what people want to communicate compared with text. We hypothesize that videos or images can reduce the effort for reporting bugs. In this RQ, we measure the number of words in the description of issues as a proxy measure for the effort.

RQ2: **Do visual issue reports lead to active discussions more than non-visual issue reports?**
Joorabchi et al. [23] showed that lack of proper communication between reporters and developers often ends up with reports in which the reported bugs are not able to be reproduced. In addition, many studies claim that comments made to a bug contribute to improving bug-fixing activities [16][29][36]. Visual issue reports might have the potential to attract developers and receive many comments from the developers. In this RQ, we examine the number of comments in the closed issues and the days to receive the first comment.

RQ3: **Do visual issue reports get resolved faster than non-visual issue reports?**
Zimmermann et al. [39] reported that issue reports occasionally have missing or incorrect steps to reproduce bugs, which delays the entire bug-fixing process [9]. Also, Ohira et al. [27] showed that bug-fixing activities are delayed when the reporter and developer are different persons, because this situation requires communication between the two. Visual issues may mitigate this issue by facilitating their communication. In this RQ, we measure the time from reported to closed to evaluate how quickly visual issue reports are resolved, compared with issues without videos or images.

2.2 Context Selection

To select projects as context for our study, we employed `GitHub Search` [14]. GitHub Search can find repositories satisfying specific criteria. To filter out unpopular, inactive repositories, or repositories that have no issues, we set up the following criteria.

- the number of stars ≥ 10
- the number of issue reports ≥ 1
- at least one commit was made in 2021

Consequently, the number of the repositories satisfying the criteria was 289,115. From November 2021 to December 2021, we collected 770,655 closed issue reports from 4,173 projects that were randomly selected. We collected all the closed issue reports from as many projects as possible in the limited time.

Table 1: Numbers of issue reports for each category

	Description	#issues
Img	issue reports containing image(s)	18,760 (9.09%)
Vid	issue reports containing video(s)	1,230 (0.54%)
None	issue reports containing no videos/images	206,415 (91.22%)

2.3 Data Collection

We first collected closed issue reports with the method `get_issue` provided by `PyGitHub`[2] that internally execute GitHub API v3.[3] In total, we collected 770,655 closed issue reports.

Next, we collected videos and images attached to the issue reports. While GitHub users can see videos and images on issue pages, the videos and images are stored in different URLs. As the URLs are written in the text description of issue reports, we parsed them with regular expressions and downloaded them. The regular expressions we used are shown as follows:

https://user-images.githubusercontent.com/[a-zA-Z0-9\-/]+\.[a-zA-Z0-9]+

Each downloaded file was determined by its extension to be an image, a video, or neither of these. We used only images and videos. Specifically, “png”, “PNG”, “jpg”, “JPG”, and “jpeg” are treated as images, and “gif”, “GIF”, “mp4”, “MP4”, and “mov” as videos. Consequently, we downloaded 34,553 images and 3,914 videos with the collected URLs.

Then, we filtered out inappropriate issue reports for our analysis. As the method `get_issue` collects not only issue reports but also pull requests, we excluded pull requests from the original dataset (294,514 issues). In addition, unlike Bugzilla [28] or Jira [1], GitHub issues do not have resolution statuses (e.g., “FIXED”, “DUPLICATED”). Instead, GitHub provides default tags to indicate these resolution statuses. We excluded 42,496 issues with tags indicating invalid issues (i.e., “duplicated”, “invalid”) or tags indicating non-bug (i.e., “document”, “question”, "enhancement"). Also, we removed 25,732 issue reports resolved in too short (≤ 30 seconds) or long periods (≥ one year) because developers leave bugs for long years without addressing or they report issues after bug-fix.

Dataset summary. The final dataset contains 226,286 issue reports, 18,760 images, and 1,230 videos. These issue reports are classified into three categories based on whether they have either image(s) or video(s). Table 1 shows the number of issue reports for each category. Note that issue reports often have both images and videos. These issue reports are counted in both *Img* and *Vid* categories (only around 0.05%). Thus, the total number of downloaded issue reports (i.e., 226,286) is different from the sum of issues (i.e., 226,405). In this paper, we refer to the issue reports in the *Img* and *Vid* categories as *visual issue reports*.

In average, issue reports categorized in *Vid* have 1.1 videos and issue reports in *Img* have 1.5 images. Out of the collected issues, only 9.09% of issue reports have images, and 0.54% have videos. While this number seems to be small, looking into the trend shown in Figure 1, the rate of visual issue reports by year is increasing from 2017 to 2021. The ratio of visual issue reports reached to 13% between 2017 and 2021. Also, we found that GitHub officially

[2]https://pygithub.readthedocs.io/en/latest/index.html
[3]https://docs.github.com/en/rest

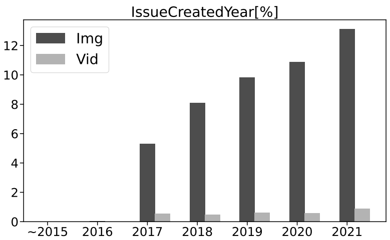

Figure 1: Percentage of issue reports for each category by year.

Table 2: Attributes we collected from the issue reports

Dimension	Attributes	Description
Report	*Images*	Number of images in the description
	Videos	Number of videos in the description
	DescriptionLength	Number of words in the description
Discussion	*Comments*	Number of comments in the report
	FirstCommentTime	Days until the first comment is made
Fix	*ResolutionTime*	Days to resolve the issue

launched the feature to share videos in May 2021 but developers often had uploaded videos before the beta release of the feature [8]. When we manually looked into issue reports, GitHub had allowed users to attach GIF files on the descriptions.

2.4 Analysis

Attributes. We retrieved attributes from the collected issue reports. Table 2 shows six attributes used in this study, which are classified into three dimensions, “Report”, “Discussion”, and “Fix”.

The attributes in the dimension “Report” are extracted from the description of issue reports or attached files when the issue was created. In particular, in RQ1, we count the number of words in reports (i.e., *DescriptionLength*) for *Img*, *Vid*, and *None*. In addition, *Images* and *Videos* are used to compare the average number of the files attached in reports. Note that these attributes are not calculated from either title, not comments (i.e., only descriptions were used). Also, URLs in the descriptions to attach images/videos are not counted as words in *DescriptionLength*.

The dimension “Discussion” has two attributes, *Comments* and *FirstCommentTime*. *Comments* is the number of comments that were made to an issue report. We utilize this attribute as a proxy measure of discussion effort. *FirstCommentTime* is the time difference in days between when the first comment was made and when the issue was reported. We use this attribute for measuring developers’ interest.

The dimension “Fix” has *ResolutionTime* which is the time difference in days between when the issue was closed and when the issue was reported.

Method. For each research question, we measure the median values of the attributes to compare *Img*, *Vid* and *None*. Also we apply a non-parametric test *Steel-Dwass test* [34] to evaluate the difference. *Steel-Dwass test* performs the multiple comparisons while taking into account the number of comparisons to prevent increases in the family-wise error rate. The datasets do not follow normal distributions, and do not satisfy homoscedasticity, and therefore are good candidates for analysis using the Steel-Dwass test.

3 RESULTS

RQ1: Do visual issue reports require less texts to report bugs than non-visual issue reports?

Figure 2 shows the distributions in the number of words written in descriptions of issue reports. The median of *DescriptionLength* was 42.0 words in *Img*, 54.5 in *Vid*, and 60.0 in *None*. Compared with *Vid* and *None*, the number of words in *Vid* is slightly smaller than that of the non-visual ones. However, no statistically significant differences are observed between them ($p > 0.01$). This implies that reporters write as many texts to describe the contents of videos as text-only reports.

Shedding light on *Img*, the number of words in *Img* is smaller (42 words) than that in *None* (60 words) with a statistical significant difference ($p < 0.01$). Compared with *Vid*, the median in *Img* is smaller than that in *Vid* (55 words). However, there is no statistically significant difference between *Img* and *Vid*.

> **RQ1:** Issue reports with images contain fewer words than non-visual issues, but still issue reports with videos require the same amount of words as non-visual issue reports.

RQ2: Do visual issue reports lead to active discussions more than non-visual issue reports?

Figure 3 shows the distributions of the number of comments and the days to receive the first comments. We observed that the interquartile range (i.e., the box) of *FirstCommentTime* in the *Vid* category are the largest, whereas that of the issue reports in the *Img* category are the shortest. However, the median in the three categories are similar and no significant differences are observed (*Img*: 0.24 days, *Vid*: 0.40 days, *None*: 0.32).

Also, in terms of *Comments*, the median and the interquartile range are almost same across the three categories, and no significant differences are observed.

> **RQ2:** Visual issue reports do not lead to active discussions in terms of the number of words and the first response time.

RQ3: Do visual issue reports get resolved faster than non-visual issue reports?

Figure 4 shows the distribution of days between when the issue report was created and when the issue report was closed. The median of non-visual issues are larger (5.96 days) than that of *Img* (4.78 days) and *Vid* (5.70 days). Also, the interquartile range of *Img* is smaller than the others. However, no statistically significant differences between any pairs are observed.

> **RQ3:** The median of resolution time in non-visual issue reports are larger than that in visual issue reports but no statistically significant differences are observed.

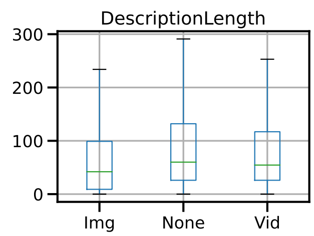

Figure 2: Distribution of # words ("Report" dimension)

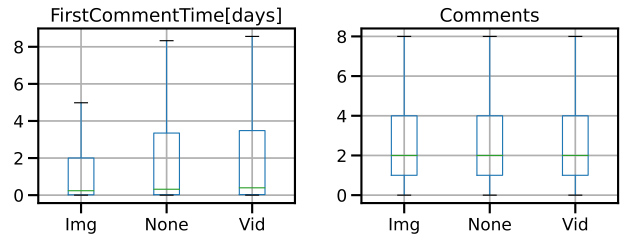

Figure 3: Distribution of days to receive the first comments and the number of comments ("Discussion" dimension)

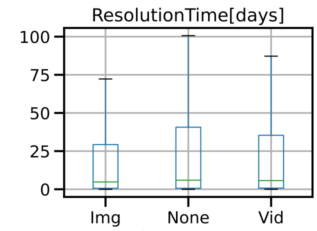

Figure 4: Distribution of resolution time ("Fix" dimension)

Table 3: Top-10 words in terms of TFIDF

	Img	*Vid*	*None*
1	image	packages	file
2	error	view	error
3	screenshot	when	lib
4	when	python	if
5	have	pageviewcontroller	line
6	if	config	java
7	version	local	get
8	get	version	have
9	using	problem	when
10	file	error	version

4 DISCUSSION

4.1 Are Visual Issue Reports and Non-Visual Issue Reports Used for Similar Aims?

In this study, we observed that developers write fewer words in issue reports with images (*Img*) compared to non-visual issue reports (*None*). On the other hands, between *None* and *Vid*, no statistically significant difference is observed (RQ1). Also, we confirmed that there are no statistical significant differences of the resolution time between visual issue reports (i.e., *Img* and *Vid*) and *None* (RQ3). These findings rejected our hypothesis. To better understand the characteristics of visual/non-visual issue reports, we examine the differences in the contents of bugs in this section.

We extracted words from the descriptions of closed issue reports in the dataset, and removed stop words such as "at", "it", and "the" from them. Then, we calculated TF-IDF values [32] to clarify the characteristic words for each types of issues (i.e., *Vid*, *Img*, *None*).

Table 3 shows the top-10 characteristic words in each category, calculated by TF-IDF. First, in the *Vid* and *Img* categories, we observed several words related to GUI, which is close to visual, such as "screenshot" and "pageviewcontroller". On the other hand, "line" and "java" related to source codes are located at the top of *None*. Second, it is worth noting that "when" is shown in all categories but it is located in the top-5 ranks of the lists in *Vid* and *Img*. This implies that visual issue reports are utilized to describe conditions/steps to reproduce bugs. In particular, as "config" is shown only in *Vid*, videos may be used to explain complicated conditions/environment to reproduce bugs. This study does not measure the degree of the difficulties in reproducing bugs but future work should investigate it.

4.2 Future Research Direction

This section discusses what should be considered by future studies.

Fine-grained analysis. In RQ2, we showed that issue reports in *Vid* take longer times to receive the first comment than issue reports in *Img*. As these might be caused by that issue reports with images attract more developers or that issue reports with videos are more difficult problems. Future studies should examine how many developers are involved [2], severity tags [37], and size of changes [19].

In this study, we studied only closed bugs and examined only resolution time in RQ3 (Fix). However, previous studies examined several statuses of bugs. For example, Joorabchi et al. [23] studied "Works For Me", Shihab et al. [33] studied reopen bugs, and Zou et al. [40] examined bug fixing rate (e.g., "Won't fix"). However, most of the studies use other bug tracking systems, Bugzilla [28] or Jira [1]. These bug tracking systems have various resolution statuses such as "Won't Fix" and "Works For Me" in default but GitHub we studied does not. Future studies should collect more issues and show the percentage of each status, etc.

Bug reproduction Automation. Developers often find it difficult to reproduce bugs using the reported information [11][31][39]. Automating this process would support developers to quickly find and fix the cause of bugs. Our final goal of this study is to automate bug reproduction. We believe that we can make use of image processing technique [3][20][25], using the uploaded videos, in order to identify which pages/screens of systems were used and what actions were done by users (e.g., which button was clicked). This approach would reduce efforts for evaluating if reported issues can be enough reproducible.

5 RELATED WORK

While numerous studies have worked on bug resolution time [12][15][22][24][35], in particular, the following studies investigated the relationship between bug resolution time and various elements of bug reports other than videos [5][6][26]. Noyori et al. [26] investigated the relationship between resolution time and topics included in the comments of issue reports. They found that bugs are resolved fast when discussions about symptoms are not needed. Bhattacharya et al. [5] developed bug-fix time prediction models using various metrics. They showed that bug severity and the number of attachments (patches) do not correlate with bug-fix time. In addition, their later work by Bhattacharya et al. [6] compared bug-fix time for high-quality and poor-quality reports. They observed that the text length of descriptions is relatively correlated with bug resolution time.

A few recent studies have utilized visual issue reports for improving bug-fixing process [13]. Cooper et al. [13] used videos and texts included in issue reports to detect duplicate ones. Compared with the study, the contribution of our work is (1) the analysis of the impact of visual issue reports and (2) the public available datasets including 1,230 videos and 18,760 images from 226,286 issue reports.

6 CONCLUSION

In this paper, we conducted a preliminary study to reveal whether visual issue reports help developers or not. Our study demonstrated that issue reports with images are described in fewer words and do not affect their resolution times.

ACKNOWLEDGMENT

This research was partially supported by `double_blind`.

REFERENCES

[1] Atlassian. [n.d.]. *JIRA Software*. Atlassian. Retrieved feb 12, 2022 from https://www.atlassian.com/software/jiras
[2] Gabriele Bavota and Barbara Russo. 2015. Four eyes are better than two: On the impact of code reviews on software quality. In *Proceedings of the International Conference on Software Maintenance and Evolution*. 81–90.
[3] Carlos Bernal-Cárdenas, Nathan Cooper, Kevin Moran, Oscar Chaparro, Andrian Marcus, and Denys Poshyvanyk. 2020. Translating video recordings of mobile app usages into replayable scenarios. In *Proceedings of the 42nd International Conference on Software Engineering*. 309–321.
[4] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiß, Rahul Premraj, and Thomas Zimmermann. 2007. Quality of bug reports in Eclipse. In *Proceedings of the Workshop on Eclipse Technology eXchange*. 21–25.
[5] Pamela Bhattacharya and Iulian Neamtiu. 2011. Bug-fix time prediction models: can we do better?. In *Proceedings of the 8th International Working Conference on Mining Software Repositories*. 207–210.
[6] Pamela Bhattacharya, Liudmila Ulanova, Iulian Neamtiu, and Sai Charan Koduru. 2013. An Empirical Analysis of Bug Reports and Bug Fixing in Open Source Android Apps. In *Proceedings of the 17th European Conference on Software Maintenance and Reengineering*. 133–143.
[7] Silvia Breu, Rahul Premraj, Jonathan Sillito, and Thomas Zimmermann. 2010. Information needs in bug reports: improving cooperation between developers and users. In *Proceedings of the Conference on Computer Supported Cooperative Work*. 301–310.
[8] Lauren Brose. [n.d.]. *Video upload public beta*. GitHub. Retrieved feb 12, 2022 from https://github.blog/changelog/2020-12-16-video-upload-public-beta/
[9] Lauren Brose. [n.d.]. *Video uploads now available across GitHub*. GitHub. Retrieved Jan 22, 2022 from https://github.blog/2021-05-13-video-uploads-available-github/
[10] Oscar Chaparro, Carlos Bernal-Cárdenas, Jing Lu, Kevin Moran, Andrian Marcus, Massimiliano Di Penta, Denys Poshyvanyk, and Vincent Ng. 2019. Assessing the quality of the steps to reproduce in bug reports. In *Proceedings of the 27th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 86–96.
[11] Oscar Chaparro, Jing Lu, Fiorella Zampetti, Laura Moreno, Massimiliano Di Penta, Andrian Marcus, Gabriele Bavota, and Vincent Ng. 2017. Detecting missing information in bug descriptions. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*. 396–407.
[12] Tse-Hsun Chen, Meiyappan Nagappan, Emad Shihab, and Ahmed E. Hassan. 2014. An empirical study of dormant bugs. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. 82–91.
[13] Nathan Cooper, Carlos Bernal-Cárdenas, Oscar Chaparro, Kevin Moran, and Denys Poshyvanyk. 2021. It Takes Two to TANGO: Combining Visual and Textual Information for Detecting Duplicate Video-Based Bug Reports. In *Proceedings of the 43rd International Conference on Software Engineering*. 957–969.
[14] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. 2021. Sampling Projects in GitHub for MSR Studies. In *Proceedings of the 18th International Conference on Mining Software Repositories*. 560–564.
[15] Harold Valdivia Garcia, Emad Shihab, and Meiyappan Nagappan. 2018. Characterizing and predicting blocking bugs in open source projects. *Journal of Systems and Software* 143 (2018), 44–58.
[16] Emanuel Giger, Martin Pinzger, and Harald C. Gall. 2010. Predicting the fix time of bugs. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*. 52–56.
[17] GitHub. [n.d.]. *Dear GitHub*. GitHub. Retrieved feb 12, 2022 from https://github.com/dear-github/dear-github
[18] Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan, and Brendan Murphy. 2010. Characterizing and predicting which bugs get fixed: an empirical study of Microsoft Windows. In *Proceedings of the 32nd International Conference on Software Engineering*. 495–504.
[19] Lile Hattori and Michele Lanza. 2008. On the nature of commits. In *Proceedings of the 23rd International Conference on Automated Software Engineering*. 63–71.
[20] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. 2019. Bounding Box Regression with Uncertainty for Accurate Object Detection. arXiv:1809.08545 [cs.CV]
[21] Kim Herzig, Sascha Just, and Andreas Zeller. 2013. It's not a bug, it's a feature: how misclassification impacts bug prediction. In *Proceedings of the 35th International Conference on Software Engineering*. 392–401.
[22] Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann. 2009. Improving bug triage with bug tossing graphs. In *Proceedings of the 7th joint meeting of the European Software Engineering Conference and the International Symposium on Foundations of Software Engineering*. 111–120.
[23] Mona Erfani Joorabchi, Mehdi MirzaAghaei, and Ali Mesbah. 2014. Works for me! characterizing non-reproducible bug reports. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. 62–71.
[24] Yutaro Kashiwa, Hayato Yoshiyuki, Yusuke Kukita, and Masao Ohira. 2014. A Pilot Study of Diversity in High Impact Bugs. In *Proceedings of the 30th International Conference on Software Maintenance and Evolution*. 536–540.
[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*. 1106–1114.
[26] Yuki Noyori, Hironori Washizaki, Yoshiaki Fukazawa, Hideyuki Kanuka, Keishi Ooshima, Shuhei Nojiri, and Ryosuke Tsuchiya. 2021. What are the Features of Good Discussions for Shortening Bug Fixing Time? *IEICE Transactions on Information and Systems* 104-D, 1 (2021), 106–116.
[27] Masao Ohira, Ahmed E. Hassan, Naoya Osawa, and Ken-ichi Matsumoto. 2012. The impact of bug management patterns on bug fixing: A case study of Eclipse projects. In *Proceedings of the 28th International Conference on Software Maintenance*. 264–273.
[28] Bugzilla Org. [n.d.]. *Bugzilla*. Bugzilla Org. Retrieved feb 12, 2022 from https://www.bugzilla.org/
[29] Lucas D. Panjer. 2007. Predicting Eclipse Bug Lifetimes. In *Proceedings of the 4th International Workshop on Mining Software Repositories*. 29.
[30] Mary C Potter and Barbara A Faulconer. 1975. Time to understand pictures and words. *Nature* 253, 5491 (1975), 437–438.
[31] Mohammad Masudur Rahman, Foutse Khomh, and Marco Castelluccio. 2020. Why are Some Bugs Non-Reproducible? : -An Empirical Investigation using Data Fusion-. In *Proceedings of the 36th International Conference on Software Maintenance and Evolution*. 605–616.
[32] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.
[33] Emad Shihab, Akinori Ihara, Yasutaka Kamei, Walid M. Ibrahim, Masao Ohira, Bram Adams, Ahmed E. Hassan, and Ken-ichi Matsumoto. 2013. Studying re-opened bugs in open source software. *Empirical Software Engineering* 18, 5 (2013), 1005–1042.
[34] Robert G. D. Steel. 1961. Some Rank Sum Multiple Comparisons Tests. *Biometrics* 17, 4 (1961), 539–552.
[35] Shahed Zaman, Bram Adams, and Ahmed E. Hassan. 2011. Security versus performance bugs: a case study on Firefox. In *Proceedings of the 8th International Working Conference on Mining Software Repositories*. 93–102.
[36] Feng Zhang, Foutse Khomh, Ying Zou, and Ahmed E. Hassan. 2012. An Empirical Study on Factors Impacting Bug Fixing Time. In *Proceedings of the 19th Working Conference on Reverse Engineering*. 225–234.
[37] Bo Zhou, Iulian Neamtiu, and Rajiv Gupta. 2015. Experience report: How do bug characteristics differ across severity classes: A multi-platform study. In *Proceedings of the 26th International Symposium on Software Reliability Engineering*. 507–517.
[38] Thomas Zimmermann, Nachiappan Nagappan, Philip J. Guo, and Brendan Murphy. 2012. Characterizing and predicting which bugs get reopened. In *Proceedngs of the 34th International Conference on Software Engineering*. 1074–1083.
[39] Thomas Zimmermann, Rahul Premraj, Nicolas Bettenburg, Sascha Just, Adrian Schröter, and Cathrin Weiss. 2010. What Makes a Good Bug Report? *IEEE Transactions on Software Engineering* 36, 5 (2010), 618–643.
[40] Weiqin Zou, Xin Xia, Weiqiang Zhang, Zhenyu Chen, and David Lo. 2015. An Empirical Study of Bug Fixing Rate. In *Proceedngs of the 39th Annual Computer Software and Applications Conference*. 254–263.