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
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 issue reports 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.)

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 [4][21][39]. Issue reports (a.k.a. bug reports) often lack the information necessary for developers to reproduce bugs [17][23]. 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 [38][40].

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 repository1, 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).

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

1https://doi.org/10.5281/zenodo.6071588
We first collected closed issue reports with the method `get_issue` words in the description of issues as a proxy measure for the effort. To select projects as context for our study, we employed PyGitHub provided by 2.3 Data Collection

issue reports from as many projects as possible in the limited time. projects that were randomly selected. We collected all the closed issues satisfying the criteria was 289,115. From November 2021 to December 2021, we collected 770,655 closed issue reports from 4,173 projects around 0.05%). Thus, the total number of downloaded issue reports 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.

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 [9]. Poter and Faulconer [30] showed that, in general, visual images are a more effective approach for describing what people want to communicate 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.

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 \( \geq 10 \), the number of issue reports \( \geq 1 \) and 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.

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 (\( \leq 30 \) seconds) or long periods (\( \geq 1 \) 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.

---

\(^3\)https://docs.github.com/en/rest

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>Images</td>
<td>Number of images in the description</td>
</tr>
<tr>
<td></td>
<td>Videos</td>
<td>Number of videos in the description</td>
</tr>
<tr>
<td></td>
<td>DescriptionLength</td>
<td>Number of words in the description</td>
</tr>
<tr>
<td>Discussion</td>
<td>Comments</td>
<td>Number of comments in the report</td>
</tr>
<tr>
<td></td>
<td>FirstCommentTime</td>
<td>Days until the first comment is made</td>
</tr>
<tr>
<td>Fix</td>
<td>ResolutionTime</td>
<td>Days to resolve the issue</td>
</tr>
</tbody>
</table>

![Table 1: Numbers of issue reports for each category](image1)

![Table 2: Attributes we collected from the issue reports](image2)

![Figure 1: Percentage of issue reports for each category by year.](image3)
On 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 launched the feature to share videos in May 2021 but developers often had uploaded videos before the beta release of the feature [8]. We manually looked into issue reports and then found that GitHub had allowed users to attach GIF files on the descriptions.

2.4 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.

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 the same across the three categories, and no significant differences are observed.

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.

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 type 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 “screenviewcontroller”. 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 Auto-reproduction.** 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 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 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 JSPS KAKENHI Japan (Grant Numbers: JP18H04097, 21H04877, 21K17725) and JSPS International Joint Research Program with SNSF (Project “SENSOR”: JPJSJRP20191502).
REFERENCES