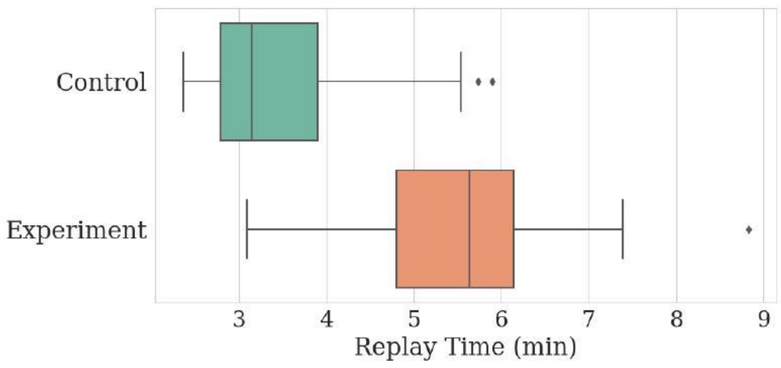

Fig. 9: Bug replay time.

to participate in the experiment. All students have at least one-year experience in developing Android apps and have worked on at least one Android apps project as interns in the company. Two software developers are more professional and have two-year working experience in a large company in Android development. Given that they all have experience in Android app development and bug replay, they are recognized as substitutes for developers in software engineering research experiments [51].

To mitigate the threat of user distraction, we conducted the experiment in a quiet room individually without mutual discussion. We first gave them an introduction to our study and also a real example to try. Each participant was then asked to reproduce the same set of 10 randomly selected bug recordings from real-world issue reports in GitHub, on average, 3.6 TAP, 1.2 SCROLL, and 1.0 INPUT per recording. The experimental bug recordings can be seen in our online appendix[2]. The study involved two groups of four participants: the control group P_1, P_2, P_3, P_4 who gets help with the reproduction steps written by reporters from GitHub, and the experimental group P_5, P_6, P_7, P_8 who gets help with the natural language description generated by our tool. Each pair of participants $\langle P_x, P_{x+4} \rangle$ has comparable development experience, so the experimental group has similar capability to the control group in total. Note that we did not ask participants to finish half of the tasks with our tool while the other half without assisting tool to avoid potential tool bias. We recorded the time used to reproduce the bug recordings in Android. Participants had up to 10 minutes for each bug replay. To minimize the impact of stress, we gave a few minutes break between each bug replay. At the end of the tasks, we provided 5-point Likert-scale questions to collect their feedback, in terms of clearness, conciseness, and usefulness. We further collected participants' feedback through a few open-ended questions, which can help us bring more insight into our tool, including how could the subtitles be improved, are there any software engineering tasks that would benefit from subtitles, etc.

Results: Overall, participants appreciate the usefulness of our approach for providing them with clear and concise step descriptions to describe the actions performed on the bug recordings, so that they can easily replay them. Box plot in Fig. 9 shows that, given our generated reproduction steps, the experimental group reproduces the bug recording much faster than that of the control group (with an average of 3.46min versus 5.53min, saving 59.8% of the time). This brings a preliminary insight of the usefulness of our generated reproduction steps to help participants locate and replay the actions.

[2] https://github.com/sidongfeng/CAPdroid

Measures	Control	Experiment
Clearness	2.50	4.25*
Conciseness	1.75	4.50*
Usefulness	-	4.75

TABLE IV: Performance comparison between the experimental and control group. * denotes $p < 0.01$.

Table IV shows the overall results received from participants. All participants admit that our approach can provide more easy-to-understand step descriptions for them, in terms of 4.25 vs 2.50 in clearness, and 4.50 vs 1.75 in conciseness, compared with the control group. In addition, they demonstrate several advantages of our reproduction steps, such as complete steps, region/text of interest, technical language, etc. Since the steps we generate are matched with each action one-to-one, participants can easily track each step, while the missing steps in the control group may confound participants: whether the step description corresponds to the current GUI. P_5 also finds the absolute positioning and element relationship description particularly useful to him, because such description can narrow down the spatial regions in GUI and easily locate the GUI element in which a bug occurs. P_3 reports that some users may use inconsistent words to describe the steps. For example, users may use "play the film" to describe the button with the text "movie", making the developers hard to reproduce in practice. In contrast, the descriptions we generate are entirely based on GUI content, so it is easy to find the GUI elements.

The participants strongly agree (4.75) with the usefulness of our approach due to two reasons. One is the potential of our structured text to benefit short- and long-term downstream tasks, such as bug triaging, test migration, etc. The potential downstream is discussed in Section V. The other is the usefulness of the subtitle in the recording, revealing the action segmentation of our approach. P_2 in the control group finds the touch indicator to be inconspicuous and sometimes GUI transitions are too abrupt to realize. In contrast, with the help of our approach, P_6 praises the subtitle in the recording as it informs the timing of each action.

To understand the significance of the differences, we further carry out the Mann-Whitney U test [52] (specifically designed for small samples) on the replaying time, clearness, conciseness, and usefulness between the experimental and the control group respectively. The test results suggest that our approach does significantly help the participants reproduce bug recordings more efficiently ($p < 0.01$). There is also some valuable feedback provided by the participants to help improve the `CAPdroid`. For example, participants want higher-level semantic step descriptions, e.g., tap the first item in the list group, which can lead to more insights into the bugs. We will