TABLE VI: Participants' positive and negative opinions on APR tools

Positive	Negative
It is effective to accelerate debugging process initially. **P1:** It helps to identify suspicious code elements quickly when developers are not familiar with the source code. **P2:** Participants could immediately recognize the buggy code elements and acquire the plausible solution to the problem without understanding all of the source code. **P3:** It can quickly identify buggy elements and further provide candidate fixes for developers to choose. **P4:** It is able to help me repair bugs faster.	***It is difficult to get started at the beginning.*** **N1:** It is difficult to adopt the tool at the beginning when developers are not familiar with it. **N2:** It is costly for developers to extract the content of the repair report. **N3:** An important point of the tool is to make the repair report easy for developers to understand. **N4:** The provided patch is generated based on frequent trial and error, without understanding the functionality of the code.
It can provide multiple suspicious buggy code elements. **P5:** It can guide me to indentify potentially risky code elements. **P6:** I could be more likely to identify the buggy code elements when the patches are provided, despite most of them being incorrect. **P7:** It provide multiple buggy code elements, and they are beneficial for me to repair the bug. **P8:** It can help me to identify buggy code snippets. **P9:** It provides accurant buggy locations, which is convenient for me to understand the bug. **P10:** It can identify the location where a bug may appear.	***It provide reports with a low accuracy.*** **N5:** The accuracy of the repair report is not high, and it also suffer from poor readability and usability. **N6:** When developers are provided with the repair report, the accuracy should be improved. **N7:** The accuracy of buggy locations and the understandability of reports are low. **N8:** The accuracy of the tool is too low, and it even attempts to generate patches on code elements, which are obviously correct.
It can provide useful patches. **P11:** The key is that the tool can generates a usable patch. **P12:** It can always provide patches and suggest useful guidelines for repairing. **P13:** It can provide useful suggestions about how to repair the bug, and sometimes it can even provided the correct patch. **P14:** It can provide patches and buggy statements. **P15:** It can indentify buggy statements and sometimes even provide plausible patches directly. **P16:** It can provide me with plausible patches and suspicious buggy statements.	***Its report is less understandable.*** **N9:** For the tools that generate bytecode-level patches, I hope the patches can be presented to developers in source-level to aid readability. **N10:** When generating repair reports, tools should eliminate irrelevant information as much as possible to facilitate quick understanding. **N11:** Tools should provide decompiled source-level patches if they are able to fix the bug, otherwise the suspiciousness value for each code element should be presented. **N12:** Displaying abundant process data is of little significance to developers. **N13:** Some repair reports have complex content, in fact, providing the most critical information is enough. **N14:** Repair reports in bytecode format need to be decompiled in advance, as they are inconvenient for me to understand.