Name: Chun-Chi Huang
Course: Software Testing
Professor: Michael McKee
Date: 05/05/2018

Chapter 4: Bug Advocacy

"[Taking] the time to make your bug reports valuable" is a very important principle for testers, because bug reports are representative of tester's values. The greater the bug reports, the more respect testers receive. If testers can write their bug reports well and make them clear, understandable, and reproducible, there is a higher possibility of these bugs being fixed. In most situations, programmers usually do not like to read bug reports because issues in bug reports can lead to more work for programmers.

For example, in my previous work experience, I could separate our QA team members into two groups based on the types of bug reports they gave. The first group of testers were ones who generated bug reports without detailed information and steps. With this group of testers, programmers still had to spend a lot of time clarifying the content of the bug descriptions and reproducing the bugs, which they disliked to do. The second group of testers had testers who would write down the detailed and clear procedures to reproduce the issues. Programmers could directly fully understand the status of bugs, and they only needed to focus on finding ways to fix bugs. Obviously, programmers liked to work with this second kind of testers.

Good bug reports significantly help programmers understand bugs, which also means that programmers do not have to waste their time investigating the reproduced procedures again. I entirely agree with the author's opinion that it is worth it for testers to contribute efforts to making bug reports valuable, and companies can also benefit from good bug reports.

The lesson, "Never use the bug-tracking system to monitor testers' performance", is a correct concept which deserves to be considered. In my opinion, I completely agree with this concept because the most important aspect of bugs is the quality, not the number. Good testers can discover an issue and provide a lot of detailed information, such as the ways to reproduce the issue, the various symptoms caused by the issue, and the status of other release versions. All the information mentioned above takes time to generate.

However, if managers only regard the number of bugs to assess testers' performances, the number will become unmeaningful. Testers would think that the higher number of bugs they find, the more credits testers will receive. Testers have several ways to increase the number, but not quality. For example, testers can separate one issue with several symptoms into several separate issues, or they can duplicate the issues that belong to the same root cause. These ways do not have any meaning and only waste programmers' time and the company's resources. This thought should be corrected.

Unfortunately, during my previous work experiences, this situation happened frequently due to manager's misconception. Management tended to use the easier way, assessing by the number of bugs that testers find, to determine the contribution that testers made. This resulted in chaos because testers only wanted to increase the number of bugs found, not the quality of bugs addressed. We should avoid this by creating correct concepts that focus on the quality of bugs.