Nicole P. Satiembre   BS Computer Engineering-Intern   February 20, 2024

Cebu Technological University-Main Campus

97 Things Every Programmer Should Know
Chapters 36-40

36. The Guru Myth - Ryan Brush

The author speaks about his observation about the prevalent “guru myth” in the world of tech. I have observed the same thing as well. We often consider other people as extremely smart that almost magically they know the solution to every problem so instead of thinking deeply, we rely on the internet gurus who might *magically* solve our issues. As working in the tech industry, we shouldn’t rely on the magic solutions to our problems. Often than not, we are more capable of solving our problems than the internet gurus because we know better about the context and system requirements of our project. When facing errors, we must learn to how to troubleshoot things before asking it in forums. It is an important skill for a developer to troubleshoot and learn how to trace logs. Every guru on the internet is also just a normal person like all of us. Becoming a guru or an expert is a result of years dedicated to learning and refining thought processes. The most important thing is keeping ourselves humble and having a beginner mindset so as not to be filled with ego the more knowledgeable we become.

37. Hard Work Does Not Pay Off - Olve Maudal

I feel like this chapter resonates with me in a lot of ways. When I was starting to code, I pressured myself and worked for long hours, even having very little time for rest, entertainment, and recreation. One thing I noticed, I got burnt out quickly and I was less productive even though I worked for long hours. This chapter is very realistic about how hard work does not always pay off. The truth is by working less, we achieve more-sometimes much more. Becoming a professional programmer does not mean you overwhelm yourself by grinding too hard. We shouldn’t focus on the length of the time but rather on how impactful we are. Though we might impress some by running off very fast that doesn’t work in the long run, what’s important is we keep a sustainable pace and learn new things every day; it doesn’t have to be as much, a little but impactful daily achievement is better than none. In the field that we work in, it is important to constantly update ourselves by reading books, joining communities, and forums, watching videos, and making projects. To be a professional developer means also shaping your character, it makes one to become a disciplined and responsible person.

38. How to Use a Bug Tracker - Matt Doar

As programmers, it is the nature of our jobs to encounter bugs and errors which is why it is crucial to know how to submit a good bug report, as well as what to look for in one, are key skills for keeping a project moving along nicely. A good bug report should convey these three things: How to reproduce the bug, as precisely as possible, and how often this will make the bug appear. What should have happened, at least in your opinion? What actually happened, or at least as much information as you have recorded. Bugs are like a conversation, with all the history right there in front of everyone. We need to ask for more information or consider what you could have missed. It is also important to take the time to explain why we think the bug should be closed will save tedious hours spent later on justifying it to frustrated managers and customers. Make sure that everyone knows how to find the bugs that the team is supposed to be working on. This can usually be done using a public query with an obvious name. Make sure everyone is using the same query, and don't update this query without first informing the team that you're changing what everyone is working on. Finally, remember that a bug is not a standard unit of work any more than a line of code is a precise measurement of effort.

39. Improve Code by Removing It - Pete Goodliffe

In this chapter, I have learned a really good principle of "YAGNI" (You ain't Gonna Need It). The author speaks about his observation that the product was taking too long to execute certain tasks—simple tasks that should have been near instantaneous. This was because they were overimplemented—festooned with extra bells and whistles that were not required, but at the time had seemed like a good idea. One of the improvements he had made to their codebase over the last few weeks was to remove chunks of it. He simplified the code, improved the product performance, and reduced the level of global code entropy simply by removing the offending features from the codebase. As humans, our imagination is wild and someone thought that a certain feature might be needed in the future, so felt it was best to code it now. This is one reason why our program becomes complex than it should be. We must print YAGNI straight in our minds and hearts. If you don't need it right now, don't write it right now. Lastly, we can apply a good principle for efficient programming: Less is more.

40. Install Me - Marcus Baker

When making a program, our ultimate goal is for the users to download it. Programming itself is very hard, enticing the clients to install it is another tricky part. So, when building our projects, we need to put first our users in mind. We should make sure our program is not skeptical or the users will doubt to install it. We must also make the downloading process simple by using a good UI that highlights the download button without many other nuances that will astray the users from where the real download button is. Even I, am annoyed to download apps from sketchy websites, and when you click the button, it’s routing me to a different link before I finally get to the real download button. We must also consider the Cost-Benefit Analysis: Instant gratification is important; the user may abandon a project if it doesn't meet their expectations immediately. We must also make the installation process simple. The user wants to know exactly where the program files will be placed and desires the ability to remove the program easily if dissatisfied. In summary, this chapter reflects the user's desire for a seamless, user-friendly experience from downloading and installing the software to using it effectively. Clear communication, simplicity, and addressing the user's needs are essential for winning their trust and satisfaction.