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

Cebu Technological University-Main Campus

97 Things Every Programmer Should Know
Chapters 11-15

11. Code in the Language of the Domain - Dan North

I have learned that making domain concepts explicit in our code helps other programmers gather the intent of the code much more easily than by trying to supplement an algorithm with what they understand about a domain. It also means that when the domain model evolves—which it will, as our understanding of the domain grows—we are in a good position to evolve the code. Coupled with good encapsulation, the chances are good that the rule will exist in only one place, and that we can change it without any of the dependent code being any the wiser. The programmer who comes along a few months later to work on the code will thank us. The programmer who comes along a few months later might be us

12. Code Is Design-Ryan Brush

I have learned that we have a software crisis where we don't really put utmost importance into the process of designing our systems. We often take this important step for granted and just focus on the mechanical or technical part instead. Getting design done fast becomes the central push of engineering firms. Inevitably, someone not deeply familiar with the design will see an unvalidated version, see the market advantage of releasing early, and say, "This looks good enough."

Utilizing unfinished designs could lead to more problems. In many cases, consumers/clients suffer from the defects caused by incomplete designs. Companies will then compensate for this by sending out patches to the broken buildings or products that they sell. By doing this, we greatly compromise the quality of the products in hopes of minimizing the cost and speeding up releases. We now have a design crisis: the demand for quality, validated designs exceeds our capacity to create them.

It is a fact that therefore, great designs are produced by great designers dedicating themselves to the mastery of their craft. Code is no different and we shouldn't take this important step for granted, rather, hold utmost importance into it.

13. Code Layout Matters - Steve Freeman

I have learned about research suggesting that we all spend much more of our programming time navigating and reading code—finding where to make the change—than actually typing, so that's what we want to optimize for:

Easy to Scan. It is our instinct to be good at pattern matching, thus, if our code is organized neatly and it behaves the same then it is easy for us to catch the differences. During our freshman years while starting to learn programming, we were taught to follow proper coding conventions. It used to be a hassle to me but I greatly appreciate it today.

Expressive Layout. Finding a good name for variables is one way to be expressive in our code. On the other hand, it is also a good practice to use a code formatter when coding. I personally use the VS Code Prettier extension which helps clean the layout of my code at compile time.

Compact Format. Aside from an expressive layout, it is also important to have a compact format so we can easily identify parts of our code. Today most IDEs provide color coding so our lives are made a little bit easier.

14. Code Reviews-Mattias Karlsson

Aside from following proper coding conventions and using code formatting tools, it is also essential to conduct Code Reviews. Although some developers don't like this additional process because they feel like being judged, this is a crucial step into making beautiful maintainable code that is less prone to bugs.

Code reviews aren't solely for correcting mistakes but it is also a good way to share knowledge and establish common coding guidelines among teams. During code reviews, it is crucial to be gentle and make our comments constructive, not caustic. We want to prevent conflicts as much as possible. It is also recommended to assign different roles where one member focuses on the documentation, another on exceptions, and another on the functionality. This way, the load is being shared among the team.

For a good coding practice, it is advised to regularly review code each week. This is also a good chance for freshers to share their knowledge and to gain insights. Lastly, make code reviews fun and light-hearted, leaving all the sarcastic comments away.

15. Coding with Reason-Yechiel Kimchi

This chapter taught me that creating guidelines such as sectioning functionalities will make code reasoning simpler. Such endpoint properties generalize concepts like preconditions and postconditions for functions, and invariants for loops and classes . We need to strive to make sections as independent of one another as possible so it can be easily modified.

Many of the coding practices that are well known and considered “good” make reasoning easier. I have gained insights about some of the most common best practices such as:

- Avoiding “goto” statements
- Avoiding modifiable global variables
- Smallest Possible Scope for each variables
- Making objects immutable whenever relevant
- Making code easy to read; organized.
- Making the code self-documenting.
- Making nested sections a function
- Make functions short and focused on a single task.
- Functions should have few parameters
- Each unit of code should have a narrow interface.
- Usage of Setters is discouraged.

Finally, coding with reason isn’t limited to the correctness of these guidelines. Arguing the reason of your code is one way to communicate the insights you gain for everyone’s benefit.