Nicole P. Satiembre **April 9, 2024**

Cebu Technological University-Main Campus

Clean Code

Chapter 14 - Successive Refinement

In order to write clean code, we must first write dirty code and then clean it. Just like in grade school when our teachers let us write a rough draft, then a second draft, then several subsequent drafts until we had our final version. To write clean compositions is a matter of successive refinement.

Usually, our initials drafts are messy but it's really not that bad. After all, it's compact, simple, and easy to understand. However, this draft of code brings seeds of a later festering pile which can later grow into a big mess. As the code progresses, arguments increases, it starts to get out of hand. It's a pile of mess starting to grow but it's not festering quite yet. At this point, we might as well want to stop. Stop adding features and start refactoring.

On Incrementalism

One of the best ways to ruin a program is to make massive changes to its structure in the name of improvement. Some programs never recover from such "improvements". The problem is that it's very hard to get the program working the same way it worked before the "improvement." To avoid this, use the discipline of Test-Driven Development (TDD). One of the central doctrines of this approach is to keep the system running at all times. In other words, using TDD, I am not allowed to make a change to the system that breaks that system. Every change I make must keep the system working as it worked before. Make incremental changes then test them early.

Conclusion

It is not enough for the code to work. The code that works is often badly broken. Nothing has a more profound and long-term degrading effect on a development project than bad code. Bad schedules can be redone, bad requirements can be redefined. Bad team dynamics can be repaired. But bad code rots and ferments, becoming an inexorable weight that drags the team down. Cleaning up bad code can be very expensive. For final thoughts, we should keep our code as clean and simple as it can be, never let the rot get started.