Nicole P. Satiembre **March 25, 2024**

Cebu Technological University-Main Campus

Clean Code

Chapter 8 - Boundaries

Using Third-Party Code. This passage talks about the tension between making interfaces that work for everyone and making them fit specific needs. It uses the example of the `Map` interface in Java, which is quite flexible but can also cause problems. For instance, if we pass around a `Map` that stores sensors in our code, we might end up with a lot of messy code because we have to cast objects back to the right type whenever we get them out of the map. To make things cleaner, we can create a class like `Sensors` that hides the messy parts and only gives us what we need. This makes our code easier to understand and less likely to break if things change in the future. So, the lesson here is that it's often better to keep complex interfaces like `Map` inside specific classes rather than passing them around everywhere in our code.

Exploring and Learning Boundaries. We use third-party code to help us get more functionality delivered in less time. While it is not our responsibility to test third-party code it may be in our best interest to write tests for the third-party code we use. Instead of experimenting and trying out the new stuff in our production code, we could write some tests to explore our understanding of the third-party code. Jim Newkirk calls such tests learning tests.

Learning Tests are Better than Free. The learning tests end up costing nothing. We had to learn the API anyway, and writing those tests was an easy and isolated way to get that knowledge. The learning tests were precise experiments that helped increase our understanding.

Using Code that Does Not Yet Exist. In software development, there are often parts of the code where we're not quite sure what's going on or how things work. To keep moving forward, even when we're unsure, we create our own boundaries. We define our own interfaces or ways of interacting with the unknown parts. This helps us stay productive and focused on our work without getting stuck. When the unknown parts become clearer later on, we can adapt and connect our boundaries with the actual interfaces. This approach also makes testing easier and helps ensure our code works correctly. Overall, it's a way of managing uncertainty and complexity in software projects.

Clean Boundaries. A good software design can accommodate change without huge investments and rework. We should avoid letting too much of our code know about the third-party particulars. It’s better to depend on something you control than on something you don’t control, lest it end up controlling you.