Reflection on Chapter 3, “Functions” in Clean Code by Rober Martin
Author: Jimmy Karlsson, 2023-09-23

In reading chapter three ’Functions’ with ‘Meaningful names’ fresh in mind, indeed, Robert can’t help but retrace a few points, the emphasis on distinctiveness and focus in the chapter reminds me of a saying by Plato... “Each man is capable of doing one thing well. If he attempts several, he will fail to achieve distinction in any”. Both chapters two and three strongly emphasise that functions should be distinct, both in name and purpose.

Ironically after the extensive verbal flow of words about names in the previous chapter, the main focus of ‘functions’ is that functions and methods should be as concise as possible, ideally, one row, but less than twenty.

Indeed the school of singular purposes almost come across as Zen-Buddhistic, “when hungry eat, when tired sleep”, you might be able to sum up chapter three in the proverbial Latin moto “Age quod agis” (Act then you act).
Meaning what you do, let that be your entire focus.

If you follow this to the extreme, all the other rules of the chapter really become redundant. A fully focused method will only do one thing, it will not have side effects, it will by its nature take the minimum of arguments, and the very thought of doing and answering in the same method becomes unthinkable.

Wholly I have and take few exceptions to the ideals portrayed in ‘Functions’, I do think that there is some undefined upper limit to how many abstractions levels is reasonable to delve into to go from the highest function to the actual implementation. Quite frankly becoming apar to Alice rabbit hole, a frustrating dive into the code through minutely coarser levels of abstraction, looking for the final function where the work is conducted.

Indeed any quick google search on ‘critique clean code’ will tell you that there is an active and alive debate in the coding community on the proper level of abstraction, the dangers of losing understanding through over abstraction, and the performance hit from doing X levels of needless function calls.

The ideal balance point I believe is down to a combination of experience and personal preference.

My inclination is to also not agree wholeheartedly with the ideal of zero arguments. The objection I make is, provided they are properly named, the arguments actually clarifies what the function is working on or with, and having that being presented in a clear way on the call line, rather than having to go into the function and look what ‘this.propertie’s are being accessed, improves understanding.

In conclusion, these rules have many good points and following them will improve readability… with the risk of loosing understandability if taken too far.