Please comment your code

Dr. Arne Babenhauserheide

<2022-07-24 So>

Code speaks to the machine and humans, comments only speak to humans. That's why comments can always be easier to understand and give more context than code.

They may not always do that, and if the code gives the information as easy as the comment, I prefer just having the code, but I spent the last years working on a codebase of which a part suffered from having no comments due to adhering strictly to clean code.

The parts with comments are always easier to understand.

Leaving out comments leads to code that replicates the comments but does not actually become clearer — it gets worse, because it mashes together different constraints.

Function names are more visible but more constrained

The main differences between the name of your function and a comment is that

1. you have more constraints in choosing the function name,
2. the function shows up in code-completion directly, and
3. The function is visible at every place your function is used (which can be an advantage, but also adds more constraints).

The three-to-five words you use in a function name are often not capable of capturing the more subtle information you need when using the function. Look at the Javadoc of some of the Java Collections methods. Which exceptions does it throw? Does it accept `null`-elements? What are its performance-guarantees?

You could make the name much longer to capture all that, too, but then it is polluting the code where it is being used with information you only needed when writing the code, but usually not while reading it. When reading code to understand it (not necessarily for reviewing), you can usually assume that the original author made the right choice, and for this a short function name is sufficient — with more information available as comments in its implementation.

So don't mix up domains. Capture information in comments where that's the best way to capture their content. And **use your coding instinct**. This is not a matter of following fixed rules or ideology but rather of using your own well-honed sense for code.

The right information at the right level

Your goal is to record information such that those reading *or* using *or* modifying your code easily get exactly the information they need at that moment. Levels of information:

1. **names** of functions, variables, and the modules that hold them
2. **comments**, inline with your code or as docstrings/apidoc/...
3. **commit messages** people access by checking annotations (often called "blame")
4. **api documentation** (maybe generated from docstrings in your code?)
5. **structure and domain** description (info-manual? exported from orgmode?)
6. **tutorials** for different kinds of tasks

If you skip any of these, the others have to compensate — badly — for the lack of information on the right level.

For ideal levels of information, see the article Teach, Don't Tell.

Conclusion

If all other things are equal, **the parts with comments are easier to understand.**

PS: There's a nice video on this topic: Healthy Software Developer: If Code Is Self-Documenting, Why Do Comments Exist?