WIKIPEDIA

Rule of three (computer programming)

Rule of three (*"Three strikes and you refactor"*) is a code refactoring rule of thumb to decide when similar pieces of code should be refactored to avoid duplication. It states that two instances of similar code don't require refactoring, but when similar code is used three times, it should be extracted into a new procedure. The rule was popularised by Martin Fowler in *Refactoring*[1] and attributed to Don Roberts.

Duplication is considered a bad practice in programming because it makes the code harder to maintain. When the rule encoded in a replicated piece of code changes, whoever maintains the code will have to change it in all places correctly.

However, choosing an appropriate design to avoid duplication might benefit from more examples to see patterns in. Attempting premature refactoring risks selecting a wrong abstraction, which can result in worse code as new requirements emerge[2] and will eventually need to be refactored again.

The rule implies that the cost of maintenance certainly outweighs the cost of refactoring and potential bad design when there are three copies, and may or may not if there are only two copies.

See also

- Copy and paste programming
- Don't repeat yourself (DRY)

References

1. Martin Fowler; Kent Beck; John Brant; William Opdyke; Don Roberts (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional. ISBN 978-0201485677.
2. Sandi Metz, *"The Wrong Abstraction"* (2016) (https://www.sandimetz.com/blog/2016/1/20/the-wrong-abstraction)

External links

- DRY (c2.com) (http://c2.com/cgi/wiki?DontRepeatYourself)
- Once and Only Once (c2.com) (http://c2.com/cgi/wiki?OnceAndOnlyOnce)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Rule_of_three_(computer_programming)&oldid=878015824"

This page was last edited on 12 January 2019, at 12:23 (UTC).