

WHITEPAPER FRONTENDART



CODE CLONES

Good, bad or ugly?

Code clones (the products of code cloning or copy and paste programming) can be a menace when it comes to long-term source code maintainability. While they seem harmless at first, they will do their damage after a few development iterations – exactly when it is too late for an easy fix. Neglecting to manage clone instances along with each other during a bug fix or a feature development of any of them will likely result in the inconsistency of the source code.

The easiest way to deal with clones would be of course to prevent their existence beforehand, but when talking about a software development project, this is not the usual case. Removing them subsequently is a much more complicated issue. The instances have to be identified first, which might not be easy after most of them evolved. But does eliminating every clone from the project even worth the required expense?

```
506         return 0;
507     }
508
509     int int_board_io(st_board *bio, char *c)
510     {
511         init_menu_text(bio);
512         bcm2835_init();
513         bcm2835_gpio_fsel(PIN_SW_CANCEL, GPIO_FSEL_OUT);
514         bcm2835_gpio_set_pud(PIN_SW_CANCEL, PUD_10K);
515         bcm2835_gpio_fsel(PIN_SW_ENTER, GPIO_FSEL_OUT);
516         bcm2835_gpio_set_pud(PIN_SW_ENTER, PUD_10K);
```

Code clones have proven negative impact on source code maintainability, so they are simply considered „bad“. One would assume that by removing all the „bad“ from the project would leave it all „good“. This is a straightforward equation; not having clones implies a good maintainability rating, seamless development, solid consistency. It's simple as that. Or is it?

The Department of Software Engineering, University of Szeged conducted a research around this topic and came up with a result that might seem somewhat contradictive at first glance. The Coupling Between Object Classes (or CBO for short) source code metric symbolizes the extent of interconnectedness between code elements. A low CBO metric typically implies good maintainability. However, after removing code clones, the CBO value increased (see Illustration), which would mean decrease in maintainability.

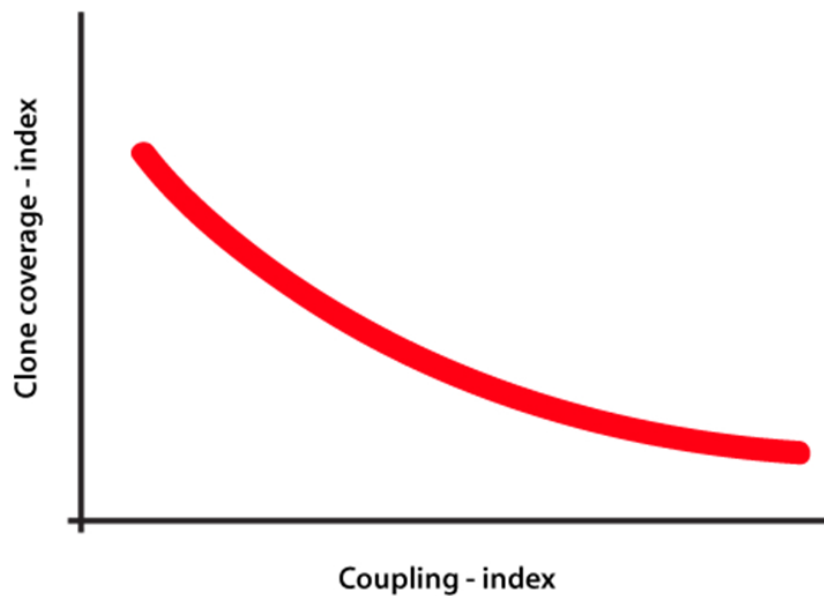


Illustration: The relation of Clone Coverage and Coupling

Despite identifying and removing every single clone instance (the „bad“), the maintenance has been negatively affected (to „worse“). The explanation to this anomalous phenomenon is that code clones may mostly be harmful, but when it comes to source code coupling, having clones can effectively lower it. This is because – thanks to code clones – separate parts of the source code have no common sections that would raise the coupling between them while lowering maintainability.¹

The final conclusion is that not every clone serves useful purpose, although not every clone is harmful either, so each existing clone instance should be detected, tracked, examined, categorized and – finally – dealt with accordingly. This task may seem impossible to do by hand. Depending on the given bundle of circumstances (project size, number of clones, complexity, staff, available resources etc.), it is likely to be – however it is most certainly a costly process always, therefore it is highly recommended to manage code clones with the help of dedicated software tools instead.

¹ <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.183.378&rep=rep1&type=pdf#page=38>