title: **Please share your steps in your development process.**
num_comments: 17
num_up_votes: 32
upvote_ratio: 0.92

Post Text

I have been struggling with the quality of my work and the comments from my CTO are just pushing me into self doubting spiral I can't find a way out of. When I try to fix the edge cases the code readability is questioned , focusing on code readability leads to questions on performance and so on and so forth. I have been trying very hard to figure out a process, where I can follow a step by step instruction manual for every story I pick up to maintain some consistency. Pardon me if this doesn't make a lot of sense and any help would be much appreciated.

Comments

Commenter_2
ID: REDACTED! ~(o.o)~ <3, Upvotes: 16
I'm going to assume here that you're correct in assuming you are at fault (and not just dealing with an asshole who thinks nobody else is good enough - in which case you need to copy their aesthetic perfectly to pass).

My process varies considerably, but the key mindset that has helped me is "Slow is smooth and smooth is fast". Each round of review costs the team 1+ hours of lost productivity due to context switching (yours and the reviewers), so it's better to spend an extra 1-2 hours getting your changes ready so they pass the first time, rather than requiring 2-3 rounds of review.

Before you show your work to someone else (interrupting their focus): let it sit for some hours (grab a coffee, write some documentation, review someone elses work). Then review it yourself - not from an "is the change correct" perspective, but instead an "is a reviewer (with no context) going to instantly understand this change".

Some things you can do to improve the chances of passing first time:

* Copy a summary of the issue into the PR description, add a summary of the approach to solving.
* Adjust the commit history with `rebase` etc, so that each commit has a sensible message & contents. CI should pass at each commit.
* Prevent your main change getting hung up discussing issues in unrelated code by using `git cherry-pick` to put unrelated changes into their own branch.
* Add review comments in github anytime it's unclear why something needs to change.

All this seems like a lot of unnecessary work to most devs I've spoken to - but I find that because it forces me to rehash my changes several times, I find loads of minor issues before review.

> ###### Commenter_3
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 3
> I'm a git noob, definitely need to check out rebase and cherry-pick, those sound extremely useful
>
> > ###### Commenter_2
> > ID: REDACTED! ~(o.o)~ <3, Upvotes: 6
> > Learning git well has been one of the most valuable things I've done.
> >
> > For most working programmers, chapters 3 and 7 of [the official manual](https://www.git-

scm.com/book/en/v2/) are particularly relevant.

Commenter_4
ID: REDACTED! ~(o.o)~ <3, Upvotes: 3
This is how I graduated from git n00b to git “slightly more experienced than a n00b”

https://learngitbranching.js.org

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
This was incredibly helpful. I do suck at the last step of self review of the code. I think I'll need to knowingly switch context during self review. Thank you .

Commenter_6
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
Personally, I benefit greatly from "sleeping on it". When I have the time available, I read over my code once before dev testing, once after dev testing, and then again the next day. Having some downtime from looking at code helps me see it with fresh eyes.

I also have a mental list of "common pitfalls" that I go through. Issues that often come up in code reviews in our team or that I know are easy to miss and can cause major problems. For example, I read over the changes once looking only for potential concurrency problems; then paying attention to names; then searching if I forgot to write changes to the database. If you can identify what those common pitfalls are for you, you could write them out into a list and use it for future self-reviews.

Unknown_User
ID: REDACTED! ~(o.o)~ <3, Upvotes: 8
[deleted]

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
The team is super helpful, and I can't blame the CTO because of the valid reasons coming from my work.

Commenter_7
ID: REDACTED! ~(o.o)~ <3, Upvotes: 3
Tracer bullets, as coined by the authors of The Pragmatic Programmer:

https://flylib.com/books/en/1.315.1.25/1/

Unknown_User
ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
Am I correct in understanding that this is essentially building a skeleton, that you can adjust before adding the meat to it?

Commenter_7
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
Correct. I often start with a program that it nothing but print statements saying what it *would* do, if it were doing it.

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
I'll have to read through the article. Thank you

Commenter_8
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
I work as an architect and the codebase I'm currently working on is absolutely horrendous (I am a consultant, so I tend to get hired when the codebase is a big ball of mud and needs serious work to save it!)

The process I follow during refactoring is roughly:

1. Remove dead code
2. Rename variables and methods where necessary for clarity/readability.
3. Simplify code by removing unnecessary tricks (this step isn't normally necessary, but the codebase I'm working on is full of them and it really hampers readability!)
4. Refactor to reduce cognitive complexity (mainly larger methods, I use SonarLint plugin)
5. Refactor to improve readability
6. Assess the public API, identify anything that can be removed or should be separated (e.g. breaks encapsulation, duplicated code) and so so.
7. Check/improve component structure and update internal methods
8. Update comments (usually removing most inline comments, and adding or fixing public API comments)

> ###### OP
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
> Thank you for the details. I will try to implement these steps to my coding pricess.

Commenter_9
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
- procrastinate
- panic
- perform

> ###### Commenter_10
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
> I definitely have the first two down
>
> ###### OP
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
> The 3rd point never really hits.. I panic and instead of perform i go the route of fuck shit up.