Reducing your error rate

Tom Rochette <tom.rochette@coreteks.org>

November 2, 2024 — 36c8eb68

1 Question

How can I reduce my error rate when using my skills?

2 Answer

When using skills in which you can make mistakes, it is important to monitor what you do and where you make your mistakes. Like any performance optimization problem, you want to figure out where you make the most mistakes and where you'll benefit the most from fixing those mistakes. If you make the same mistakes 100 times at the cost of a minute each time, that may be preferable to making 1 mistake that cost 100 minutes to fix, given that once the mistake happens, the cost stays the same.

Document your skills. Write down what you do, when you do it (triggers) and what kind errors you make during those steps. Evaluate how many times you make that mistake and how long it costs to fix. Then when you use your skills, track when you make mistakes and how long it takes you to fix the mistake.

As an example, think of a software engineer doing code reviews. Reviewing code requires going through a variety of checks: is the build passing? is the functionality properly implemented? are there tests? are the new files in the proper location? Without a list, the engineer is left looking at the code without any clear checklist. If he is methodical he will have a list he goes through in his head. If not, then he will most likely only look at the code and give it a summary opinion, that is, whether he likes what he sees, or not.

Given a non-explicit methodology it is hard to assess where the mistakes are made and which mistakes cost the most to fix. If you don't check that tests were written for the new functionality or changes, what impact will it have in the future? Depending on how likely the code is to change, the likelihood that something gets broken may be significant. While adding a test may require a few minutes, one has to judge how much time would be wasted if a bug were introduced in the piece of code. As code complexity increases, so does the cost of fixing issues in that code.

With an explicit checklist you will be able to track the things that you want to verify before code is merged. As your checklist covers more and more cases, this list will reduce the likelihood that the person who wrote code made a mistake that gets to production. By the same token it will increase your effectiveness as a software engineer to produce quality code.