A metric that is consistently at 100% is probably broken

devblogs.microsoft.com/oldnewthing/20210202-00

February 2, 2021

Raymond Chen

I was working on a scenario that involved multiple components, where success is defined as “completes within 30 seconds”. The team that was responsible for the component that initiated the scenario had a number of quality metrics covering this scenario, each one focusing on a particular variation. Some of the metrics for a particular variation showed that the scenario was succeeding at a rate of 100%. On the other hand, my metrics for the component that completed the scenario showed that the scenario was failing miserably for the same variation on a certain category of systems.

So I took a closer look at the two sets of metrics to see why there was such a huge discrepancy.

For pass/fail metrics, it is common to design them so that each run of the scenario returns returns a score of 0 (fail) or 100 (pass). That way, the average score gives you the pass rate in percent, and then you set a target like “average ≥ 95”. You can also set the result of a run as *ignore*, meaning that it should be discarded from the data set. For example, if the metric is “Time to accept a valid password”, you would mark the run as *ignore* if the password turns out to be invalid.

One of the other team’s metrics used this algorithm: “If event X occurs within 30 seconds, then return 100, else return 0.”

The way the component works is that it initiates the scenario, and records some information about the scenario in progress so that it can do some follow-up work when the scenario completes. Event X is raised when the scenario completes, or when the follow-up object abandons the scenario.

The follow-up object abandons the scenario after 30 seconds.

This means that once the scenario hits the 30 second mark, the follow-up object gives up. And that signals the event that the metric was using. The time interval that the metric is measuring is capped *by its own code* at 30 seconds.

I suspect that the person who wrote the metric was unaware of (or at least had forgotten about) the internal timeout in the follow-up object. As a result, this metric falsely reports that every scenario completed successfully, because the internal timeout causes the "end of scenario" event to fire after 30 seconds at the latest.

There was another metric that I found, whose definition is "If event Y occurs within 30 seconds, then return 100, else ignore." This metric also has a perfect pass rate, because it throws away all the failures!

Actually, this metric had another defect: The target for this metric was set to "average ≤ 30,000", thinking that the metric was measuring time in milliseconds, rather than success rate.

What I learned is that you should be highly suspicious of a metric that gets consistently perfect scores. It probably means that the metric is broken.

Bonus chatter: One of my colleagues created an eponymous rule on this subject: "Alice's rule of data: If your metric shows 100% or 0%, it's broken."

Bonus bonus chatter: One of my now-retired colleagues told me that he spent a lot of his time studying tests that had a pass rate of 0% or 100%. "They were either broken or useless."

Bonus bonus bonus chatter: I was investigating a metric that had started failing. What I found was that another team added a new scenario to their component. My component is the primary client of their component, so the team added a metric to my component to verify that their component was working. Okay, so far so good.

The way their component works is that you tell it to start, and then you monitor its progress. And the way they determined whether the component was working was to see if it reported (within a reasonable amount of time) either "No scenario applies" or "Ran scenario X", where X is on a hard-coded list of valid scenarios.

When they added a new scenario to their component, they forgot to update this hard-coded list. This caused all instances of the new scenario to be treated as unknown failures. With their permission, I revised the metric to get rid of the hard-coded list. As long as it reported the successful run of any scenario or reported that no scenario was applicable, I declared the metric to have passed. Now they can add new scenarios any time they want without having to go in and update my component.

The metric immediately recovered. But it recovered too well: It started passing at 100%, which was suspicious. I was put in the odd position of secretly wishing that my metric would go down, just a tiny bit, to prove that I didn't break it.

I dug into the raw data and found that the metric was passing at 99.97%, but the dashboard rounded it to 100%.

Raymond Chen

Follow