QA metrics

In the previous video we discussed types of QA metrics. In this article we will look at examples of metrics that fall into each category.

The three categories of metrics are:

- **Technical quality:** the degree to which a product meets specification.
- **Business quality**: the degree to which a product achieves business KPIs.
- **Process quality**: the effectiveness and efficiency of the QA process.

Technical Quality

Quite often software teams are using the amount of bugs as a measure of technical quality, because it seems easy. What is a bug, really?

> *A bug is a defect or an error that causes the product to produce an incorrect or unexpected result.*

So, a bug is an error. Fewer errors equal better quality. Sounds simple, right? Unfortunately, it's not that easy.

Of course, it's important to find and document bugs so the overall quality of the solution improves. But where it goes wrong is when the teams stop focusing on how much value/quality they have delivered, and start fixating on how many bugs they’ve found.

According to the definition of quality, it doesn’t really matter

how many bugs are found before end users see the product. Some people may think that if testers don’t find bugs during testing, they’re not doing their job well. Others may believe that if testers do find bugs, then developers aren't doing their job well.

I advocate that if software is shipped and delivers value, then the team is doing a good job regardless of how many bugs were recorded during the process. That number is irrelevant as a measurement of quality.

What is relevant, however, is the number of bugs that escape the process, called escaped defects, or **defect leakage**. In an ideal world, this number should aim for zero.

However, defect leakage in absolute terms may not be very informative. What does it mean to have leaked 5 or 10 bugs? If those 10 bugs were introduced by a single feature, that’s a significant issue. But if they occurred across a multi-year, multi-million-dollar project, it might be perfectly fine.

This is where **defect density** becomes useful. Defect density measures the number of defects found per unit size of a software module (for example, the number of defects per feature or per lines of code). In this context, we’re specifically referring to leaked defects that escaped our quality gates.

The lower the defect density, the better the overall quality. Monitoring this metric helps QA teams evaluate not only the quality of the codebase but also, indirectly, the effectiveness of the QA process itself.

Not all defects are equal in terms of impact. A defect can be a wrong choice of font colour on a webpage that customers will barely notice; or it can be a wrong colour of key control that can cause an airplane pilot to make a mistake. That’s why, in addition to tracking the number of leaked defects, we also

measure the **severity distribution** of those defects, specifically how many are high-priority versus low-priority issues. This helps us better understand the real-world impact of what's slipping through our quality gates.

Another valuable metric is **test coverage**. Test coverage is a metric that quantifies the extent of testing by evaluating the proportion of system components tested out of the total available. Generally, the higher the test coverage, the better. It means you don't have components not covered by testing.

Test coverage can also be extended to **Automated test coverage**, which shows how many components are covered by automated testing. Test automation coverage is a useful metric to identify areas of the system that may require additional manual testing due to lack of automated test coverage.

When you have a high level of test coverage, you can also use the number of passed tests as an indicator of value delivered. For example, if you have 50 test cases that collectively represent 100% test coverage, and 45 of them pass, you could reasonably say that 90% of your product is functioning as intended—and therefore, delivering value.

This approach can be a useful way to **measure the amount of value shipped**, but it only works reliably when test coverage is high. Without comprehensive coverage, the percentage of passed tests may give a misleading picture of the product's overall readiness or quality.

Business Quality

Depending on the project's KPIs and the benefits outlined in the business case, the key **business quality indicators** will vary. From a business perspective, a KPI should directly support the overall business goal.

For example, if the objective of the project is to increase the conversion rate during the checkout process, then there should

be a corresponding KPI that measures conversion performance. The chosen metrics must align with the intended outcomes to ensure that quality is evaluated in terms of actual business impact.

In addition, there are some standard measures that are just good to collect, including **customer satisfaction or CSAT**. Although there are many factors that can affect CSAT, it serves as a kind of last-resort indicator. Even if you're not collecting other business metrics, keeping an eye on customer satisfaction is a must. If customer satisfaction drops as you release new features, you are probably not doing something right.

QA Process Quality

Test reliability is a simple concept but a powerful metric. It measures how often your testing produces false positives; that is cases where defects are reported but later turn out not to be defects at all. It’s essentially a reverse metric: the fewer false positives, the higher the reliability of your testing process. One key point to note is that test reliability applies to all reported defects, not just leaked defects.

Mean Time to Detect (MTTD) measures how long it takes to discover a defect, assuming one exists. Once a defect is introduced, it remains unknown until a tester or an automated system uncovers it. When a defect is detected, we can trace it back to the point in time it was created and calculate how long it took to find. By doing this across all identified defects, we can determine the mean (average) time it takes to detect them. The shorter the MTTD, the better—it indicates a faster feedback loop and a more responsive testing process.

Mean Time to Repair (MTTR) is a similar type of calculation, measuring how long it takes to resolve a defect and re-validate it to confirm that it has been fixed. To calculate MTTR, you take the average of the resolution times for all identified issues.

In general, the shorter the MTTR, the better, as it indicates faster turnaround and responsiveness. However, be cautious when interpreting this metric—some bugs may be intentionally delayed if they're considered low priority, which can skew the average.

Wait time to start testing is used to measure how connected your processes are. For example, you might have a piece of functionality developed and ready for verification, but it just sits there waiting. Collecting this wait time across all instances and calculating the mean will show you the degree of disconnectedness in the process: time delays that could have been avoided, leading to faster time to market.

Finally, **test execution speed** is the time it takes to complete the execution of the test suite. The faster it is, the better (especially for regression tests) provided that **test reliability** and **defect leakage** do not suffer.

Summary of QA metrics

Technical quality metrics:

- **Defect Leakage**: amount of bugs that made their way to production.
- **Defect Density**: the number of defects leaked per unit size of product.
- **Defect Severity Distribution**: a proportion of severe defects vs low impact ones.
- **Test coverage**: proportion of components tested out of total number of components.

Business quality metrics:

- **Business KPIs**: a measure of achievement of a business goal in business terms.
- **CSAT**: a measure of customer satisfaction from our

product.

QA process quality metrics:

- **Test reliability:** the proportion of false positives out of the total amount of defects.
- **Mean Time to Detect:** time it takes to uncover an introduced defect (on average).
- **Mean Time to Repair:** time it takes to fix an uncovered defect (on average).
- **Wait time to start testing:** time the functionality sits waiting to undergo testing.
- **Test execution speed (cycle time):** time it takes to finish execution of a test suite.