A second look at software testing metrics

The question of how to measure the effectiveness of testing procedures fuels heated controversies. In reality, however, testing metrics are subjective. It is recommended, therefore, to adopt a different approach and move to measuring data and processes instead of measuring people.

By Arik Aharoni

There is only one issue on which every member of the software testing community has his own independent opinion - metrics!

This controversial topic fuels emotional debates and most discussions end with no conclusive outcome. It touches on many different issues: How to measure testing efforts? What is the best way to evaluate effectiveness? Which of the different elements should be quantified? How do we estimate the quality of our testing performance, and many more questions.

As providers of software testing management tools, the metrics question has always bothered us, as we try to figure out the best approach. We have read professional literature, actively participated in testing forums and discussed it with industry experts. We have even asked customers to provide their own feedback on how they see its implementation, and what they would like to measure. But, we have still not managed to present the ultimate metrics formula.

Metrics is subjective

As it turns out, there is no one single good practice. The painful truth is that testing metrics is subjective, and can be only decided upon on a per company, or even a per project basis.

While discussing the issue of testing metrics we should consider the following:

- What is the project's scope?
- What is the team's size; is there more than one team involved in the development project?
- Is it a new project or a follow-on?
- Does the project have any business measures of success and how does it reflect on the testing stage?
- Which methodology is used, if at all?
- Are there any previous metrics data to compare with? Do we want to compare?
- What is the available data with which to create metrics?
- Do we want this metrics to define whether the tested application can be issued to the market? Or is there a different goal?
- Who is calling the shots in the project (QA? Development? Marketing?)
- Are we concerned about the process or only about the end results?

The list goes on. And the answers vary from one project to another.

We have no intention of discussing different metrics options, or of dealing with calculations, data gathering, the ways to apply them, and so on. Our sole intention is to raise awareness of one question which all software pros should try to honestly answer: Are you using metrics to improve your software testing, the team's productivity and performance? Or have metrics turned into a target itself, altering the way you work?

Chasing your tail

We strongly believe that the evaluation act itself - collecting specific data elements and creating measurements and criteria - frequently affects people's behavior and as a result the final outcome as well.

Once something is measured, people tend to automatically assume it is of importance, and therefore act to achieve better results. It is human to want to succeed and we will do almost anything to get our numbers right.

This hidden agenda turns out, frequently, to be counter-productive and it negatively impacts the overall work procedure. Some team members may concentrate on their own targets, and ignore, even unintentionally, the wider needs of the team and project.

So, for example, a police unit, which is being rewarded for generating more revenue, may focus its efforts on handing out as many parking tickets as possible at the expense of chasing thieves.

Twisting the tester's job

This is also true in the testing world. Many firms, unfortunately, adopt a simple tool - measuring a tester by the number of defects found. But, the main reason that companies favor this method is because it is widely believed that the tester's job is to find bugs. Managers are fooled into believing that measuring errors will provide a good overall assessment of the job quality as more defects identified early would mean early fixes and a better end product.

In reality, however, the exact opposite might happen. This metric may encourage testers to try to increase the number of bugs simply by reporting more minor defects, splitting possible defects into several different ones, and avoiding making the effort to find bugs that require careful and time-consuming investigation.

In such a case, the tester might pay less attention to describing the defects in detail and might instead spend more time on identifying more defects. This may have a knock-on effect on development or product release. The end result will be that developers deal with more issues, of which many could turn out to be either non-issues or very minor ones.

How to choose a good metrics

There are a few ways to handle the “human factor”:

- Measure a process or a product instead of people.
- Measure things at the end of the process (it is best to measure the process results), and not at the early stage of the process.
- Refrain from using metrics as a way to follow up on people, or worse - as a way to measure them.
- Make sure the group understands the logic behind the measured data and get its commitment.
- Metrics are good decision and management tools, but don’t let them replace human judgment or gut-feeling. Trust your instincts and team feedback, and not just statistics.

After choosing your metrics, you can test it against a set of simple criteria. I believe that these criteria can help determining if the metrics are doing their job.

1. Does this metrics measure important business data?
2. Does this metrics measure a result of a process?
3. What is the impact of the measuring team on achieving the targets?
4. Can the team manipulate the process in a way that will influence the index?
5. Do you have a plan for what to do when you will get the metrics results?

Let’s review one example. For a testing team, a possible approach could be setting business targets, which measure the efficiency of the testing team; it will be set by the number of defects found in the product during the first month after releasing. The logic behind it:

1. Having fewer defects in production is an important factor in a business’s success.
2. It measures the result of the testing process.
3. The testing team is not the only group responsible for this result, but it definitely has a major impact on it.
4. The testing team cannot manipulate the process to achieve this metrics.
5. Plans to improve specific segments of the project process, and the testing in it, can easily be made if necessary.

Of course, this metrics ignores issues such as inefficiencies during the testing procedure, time spent on testing, and even the influence of the tester’s quality on the number of defects. Other types of metrics can cover other such elements, as long as they are built correctly.

We believe that measuring work is important and can give the tools to evaluate the quality of the work done and the product, and improve processes and results. However, it should be carefully thought of.

Standing true to our believes, Testuff decided to let our customers export raw data directly from our test management application, rather than settings ready-made-metrics. This allows them to independently design their own metrics and hopefully will also result in better products.

The writer is the CEO of Testuff, which specializes in developing better tools for the QA and testing industry, with its flagship SaaS application, Testuff Test Management Tool.