This is how you find metrics that matter

A software process tends to be more difficult to manage and optimize than an industrial manufacturing process. This has little to do with the “intellectual challenge” or “inherent complexity” of software. The history of industrial manufacturing just happens to be longer, its underlying practices, processes, and tools are much more standardized, and there is less variation to manage than with software.

A business that is only as good as the software driving it can stay competitive by continuously improving and optimizing its software process. And no: this has nothing to do with killing innovation. In fact, the better, faster, and more automated your process is, the more you can afford to innovate.

The software process needs rigorous measurement. We need to see if things are developing for better or for worse. We need to rapidly find the right levers to adjust when improvement or corrections are needed. Luckily, the software process is easy to measure. Almost everything from code to helpdesk involves tools that produce a wealth of data that can be used to understand, control, improve, and forecast the quality of the software and the process creating and releasing it.

People in different roles need different views of the same information. Trying to run the software process by focusing exclusively on defect reports is a bit like trying to run a business by looking at accounts receivable and accounts payable only. Likewise, trying to manage and develop a business by reading fiscal statements is a doomed effort, but analysing the trends of crucial financial indicators gleaned from fiscal statements is much more useful.

The purpose of software metrics is to facilitate the best possible decisions at the right time.

Current literature provides hundreds of handy software quality metrics. The metrics that have served me best in software quality management are the following:

1. Pass/fail statistics.
2. Error accumulation
3. Defect detection percentage
4. Time/Phase distribution of errors
5. Architectural distribution of errors
6. Rework accumulation
7. Wait accumulation
8. Code changes per day/week

But this is just a tip of the iceberg. A comprehensive account of how to make metrics work to your advantage is here.

A word of warning may be necessary, though. Most people hate being measured. They don’t believe management has good intentions when it comes to metrics. This means any measurement effort that requires people to report things manually is doomed to fail. Therefore, automate your metrics.

Attitudes towards measurement will change over time once people learn to use metrics first to understand, and only afterwards to control.

Previous: How to get test automation right

Qentinel Robotic Software Testing

Beginning: How to apply new control attitude to your digital backbone