11

September 2010

testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

Metrics

Counting defects

by Bert Wijgers

Defect counts are often considered as measurements of product quality. However, the most important defect count in that respect is by definition unknown; the number of undiscovered errors. Defect counts can be used as indicators of process quality. In doing so, one should avoid assuming simple causal relations. Defect counts can provide useful information but have to be presented with care.

In one of the first projects I joined it was my task to bring the defect process under control. There were hundreds of open defects and they kept coming. At the end of every week I exported the data from the defect management tool and made a few cross-tables in a spreadsheet to show the numbers of defects still open for the next release. We had a weekly meeting to classify and assign new defects and to re-assign old ones. Whenever the workload for the next release became too heavy we would assign some defects to the release after the next. In fact, we spend more than half of the meeting time changing the numbers of the release in which defects ought to ship. As a result, the weekly status report was always optimistic. After a few months my services were no longer needed. I showed someone how to generate the desired reports and the project continued for another year and a half before the plug was pulled and the project had to start all over. In retrospect it is clear to me that things had been going in the wrong direction long before I joined the project and that the weekly status reports hadn't helped one bit to change the course of events. I ease my conscience with the thought that I didn't know then what I know now.

The idea that "you can't control what you can't measure" (DeMarco, 1982) has been abandoned for some time now. DeMarco himself apologizes for overstating the importance of all kinds of software metrics in 1995: "I can only think of one metric that is worth collecting now and forever: defect count. Any organization that fails to track and type defects is running at less than its optimal level. There are many other metrics that are worth collecting for a while."

In this article I want to focus on the use of defect counts. Therefore we need to realize what they are and what they are not. We also have to realize why we do and do not need them. Only then defect counts can be used effectively in decision making.

What a defect count is and what it is not

A defect count is a number of defects that have been discovered. In order to be included in a count a defect has to be logged and classified. The number of severe and still open defects caused by specification errors and found during system test is an example of a specific defect count that might be of interest to somebody.

What does a defect count measure? Things are not always what they seem to be. Kaner and Bond (1994) call this "construct validity". This is a well known concept in social sciences. Does an IQ test really measure intelligence or does it only measure the ability to do IQ tests? Software development is a social activity and that implies lots of variables that affect defect counts.

Let's take the example of defects found per week. Typically, in the first weeks of a software development project only few defects are found. Often this is because the test environment is not yet up and running and the test team is preparing itself. Then the test team gets going and there is a peak. After a while the number of defects found per week diminishes. Often this is taken as evidence that there are only a few defects left to discover in the software and that the product is therefore ready for release. This is not valid. Just like there were other causes for the low defect counts in the first weeks there are other causes for the fact that defect counts drop in the last weeks.

Just before a deadline the focus of the test team shifts from bug hunting to other tasks. The testers have to write their reports and conserve test scripts. They elaborate on old defects to help developers fix these. In fact, they have to shift their focus because defect counts are supposed to drop when the deadline is nearing. Bug hunting is discouraged at these stages of the software development process. The number of defects per week is at least strongly influenced by the focus of the test team.

Counting defects is not the same as measuring product quality. There are a lot of definitions of quality but as Kaner (2000) points out: "The essence of 'quality' is 'qualitative.'" This can never be captured by objective quantitative measures. Quality involves the satisfaction of the user. This is something else than the absence of defects. Even if we define software quality as the absence of defects after release, the number and nature of the defects that

occur during the production phase of the software can only be indicative. The defects that are still hidden in the software might or might not surface after shipment. At the moment of release their number is and will always be unknown. The only thing that defect counts directly measure is the amount of defects in the defect management database.

When the user is taken into account, measurement of quality has to be subjective in nature. Therefore, it is not advisable to formulate acceptance criteria in terms of defect counts. Obviously, a piece of software cannot go live with one or more showstoppers, but the number of allowable minor defects cannot be agreed upon in advance since there is no telling how the actual unsolved defects will influence the behavior of the system in interaction with actual users.

Roughly the same metric that is used for measuring product quality is used for measuring the quality of testing. The number of defects found per week or per working day is sometimes used to evaluate the performance of individual testers or testing teams. As pointed out earlier defect counts are influenced by the dynamics of the software production process. Undoubtedly, motivation and testing skills play an important role as well. But so do product quality, testing strategy and a lot of other things. The dangers associated with the use of defect counts in evaluating testers and their teams will be addressed in the next section.

Why we do and don't need defect counts

Defect counts can be very useful in telling us something about the software production process. They can be compared between phases or timely intervals (weeks, months) in order to reveal tendencies in the process. Defects can be counted separately for different modules of the same system. This information can be used to allocate resources to certain modules or phases. That is, defect counts can be used as management information. In doing so, there are a few pitfalls to be avoided.

When using defect counts to evaluate individuals or teams, this should only be done for coaching purposes. When people notice that they are being judged and rewarded on the basis of the numbers of defects they find, their main focus will be on improving these numbers. Testers might be tempted to log different variations of the same defect. If one logged defect is enough to expose a problem and get it solved then there is no need to log additional defects that are caused by the same problem. Testers might be discouraged to look for more complex defects or for defects in more stable areas of the system. It takes more time to look deeper, but the defects that can be found there might well be more significant. Testers might be discouraged to work as a team and to work on process optimization. By improving their individual defect counts they will diminish the overall quality of the service provided, individually and as a team. Therefore, defect counts should never be considered as goals in themselves.

When management doesn't act upon the metrics provided from the defect management database there is no use in providing the metrics in the first place. Far too often the tables and charts with defect information are only used to talk about in the weekly meetings. The Goal Question Metric approach (Basili and others, 1994) provides a framework to define defect counts that will be used. By clarifying the goals of the project and the questions that should be answered defect information can be presented in a way that satisfies the true needs of management.

The first step is to identify the goals. Every goal has a purpose, an issue, and an object. The fourth element of a goal, the viewpoint, doesn't play a role in defect counts, since these are objective metrics that do not depend on point of view. An object may be a product, a process or a resource. An example of a goal is: To improve (purpose) the efficiency (issue) of testing (object). Take your time to identify the goals and questions at the appropriate level. Usually that is the people that you will be sending your reports to. Aim as high as you can. When deadlines are getting close, defect reports go way up in the organization. In order to make these reports you need to identify these high level goals. The next step is to pose the questions that have to be answered. For example: Are the logged defects valid? Or: How many defects are found? The final step is to define the metrics. For example: the number of valid defects divided by the number of logged defects. Or: the number of defects found per testing day. There is a number of ways to come up with these goals, questions and metrics but every which way, they should be verified regularly. Goals and questions can change. If so, the metrics probably have to change as well.

Defect counts can be used to guide resource allocation. When there is a disproportional amount of defects found in a certain subsystem then it might be wise to assign an extra or a more experienced developer to that subsystem. However, Fenton and Neil (1999) point out that a high defect density, that is a high number of defects per line of code, is probably a sign of good testing rather than a sign of poor quality. So it might be wise to assign extra testers when there is a low defect density. Defect counts are only indicators of process quality. They reveal correlations, not causal relations.

Defect counts can be helpful in planning future projects. Let's take the example of the number of defects per injection phase. Generally speaking, the longer a defect remains undiscovered the more it will cost to fix it. Testing effort can be allocated according to the observed distribution for a similar product. If lots of defects were injected in the design phase, then extensive reviewing of design documents can be planned.

Management information

There are endless possibilities to define defect counts and they can all be useful under certain circumstances. It is our responsibility to provide the numbers that really matter. When managers ask for information from the defect management database, make them sweat. The Goal Question Metric approach is a good way to get commitment from managers. They will be forced to think about what they really need to know and in the process they will gain insight in the complexity behind the numbers. What is true for most things in life is true for defect counts; what is given too easily is often not fully appreciated. Managers have to realize that defect counts are not for free. It takes time and effort to define and present the right metrics and they should be valued accordingly. Only then defect counts can become management information.

References

Basili, V.R., Caldiera, G and Rombach, H.D., 1994. The Goal Question Metric Approach, in Marciniak, J.J. (ed.), Encyclopedia of Software Engineering, volume 1, pages 528-532. John Wiley & Sons.

DeMarco, T., 1982. Controlling Software Projects: Management, Measurement, and Estimates. Yourdon Press, New York.

DeMarco, T., 1995. Mad About Measurement, in Why Does Software Cost So Much?, pages 14-44. Dorset House, New York.

Fenton, N.E. and Neil, M., 1999. Software metrics: successes, failures and new directions. Journal of Systems and Software 47(2-3), pages 149-157.

Kaner, C., 2000. Rethinking software metrics. Software Testing & Quality Engineering, Volume 2, Issue 2.

Kaner, C. and Bond, W.P., 2004. Software Engineering Metrics: What Do They Measure and How Do We Know? 10th International Software Metrics Symposium.

Biography

Bert Wijgers is a test consultant with Squerist, a service company in the Netherlands. Squerist focuses on software quality assurance and process optimization. Its core values are innovation, inspiration and confidence.

Bert holds a university degree in experimental psychology for which he did research in the areas of human-computer interaction and ergonomic aspects of the workspace. He has worked as a teacher and trainer in different settings before he started an international company in web hosting and design for which a web generator was developed. There he got the first taste of testing and came to understand the importance of software quality assurance. Bert has a special interest for and the social aspects of software development.
He uses psychological and business perspectives to complement his testing expertise.

In recent years Bert has worked for Squerist in financial and public organizations as a software tester, coordinator and consultant. In these assignments he has made extensive use of metrics, notably defect counts.