Measuring Defect Potentials and Defect Removal Efficiency©

Capers Jones
Software Productivity Research, LLC

There are two measures that have a strong influence on the outcomes of software projects: 1) defect potentials and 2) defect removal efficiency. The term defect potentials refers to the total quantity of bugs or defects that will be found in five software artifacts: requirements, design, code, documents, and bad fixes*, or secondary defects. The term defect removal efficiency refers to the percentage of total defects found and removed before software applications are delivered to customers. As of 2007, the average for defect potentials in the United States was about five defects per function point. The average for defect removal efficiency in the United States was only about 85 percent. The average for delivered defects was about 0.75 defects per function point.*

There are two very important measurements of software quality that are critical to the industry:

1. Defect potentials
2. Defect removal efficiency

All software managers and quality assurance personnel should be familiar with these measurements because they have the largest impact on software quality, cost, and schedule of any known measures.

The phrase *defect potentials* refers to the probable numbers of defects that will be found during the development of software applications. As of 2008, the approximate averages in the United States for defects in five categories, measured in terms of defects per function point and rounded slightly so that the cumulative results are an integer value for consistency with other publications by the author, follow.

Note that defect potentials should be measured with function points and not with lines of code. This is because most of the serious defects are not found in the code itself, but rather in requirements and design. Table 1 shows the averages for defect potentials in the U.S. circa 2008.

The measured range of defect potentials is from just below two defects per function point to about 10 defects per function point. Defect potentials correlate with application size. As application sizes increase, defect potentials also rise.

A useful approximation of the relationship between defect potentials and defect size is a simple rule of thumb: application function points raised to the 1.25 power will yield the approximate defect potential for software applications. Actually, this rule applies primarily to applications developed by organizations at Capability Maturity Model® (CMM®) Level 1. For the higher CMM levels, lower powers would occur. Reference [1] shows additional factors that affect the rule of thumb[1].

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

The phrase *defect removal efficiency* refers to the percentage of the defect potentials that will be removed before the software application is delivered to its users or customers. As of 2007, the average for defect removal efficiency in the U.S. was about 85 percent.

If the average defect potential is five bugs – or defects – per function point and removal efficiency is 85 percent, then the total number of delivered defects will be about 0.75 per function point. However, some forms of defects are harder to find and remove than others. For example, requirements defects and bad fixes are much more difficult to find and eliminate than coding defects.

At a more granular level, the defect removal efficiency against each of the five defect categories is approximate in Table 2.

Note that the defects discussed in this section include all severity levels, ranging from severity 1: *show stoppers*, down to severity 4: *cosmetic errors*. Obviously, it is important to measure defect severity levels as well as recording numbers of defects[2].

There are large ranges in terms of both defect potentials and defect removal efficiency levels. The *best in class* organizations have defect potentials that are below 2.50 defects per function point coupled with defect removal efficiencies that top 95 percent across the board.

Defect removal efficiency levels peak at about 99.5 percent. In examining data from about 13,000 software projects over a period of 40 years, only two projects had zero defect reports in the first year after release. This is not to say that achieving a defect removal efficiency level of 100 percent is impossible, but it is certainly very rare.

Organizations with defect potentials higher than seven per function point coupled with defect removal efficiency levels of 75 percent or less can be viewed as exhibiting professional malpractice. In other words, their defect prevention and defect removal methods are below acceptable levels for professional software organizations.

Most forms of testing average only about 30 to 35 percent in defect removal efficiency levels and seldom top 50 percent. Formal design and code inspections, on the other hand, often top 85 percent in defect removal efficiency and average about 65 percent.

As can be seen from the short discussions here, measuring defect potentials and defect removal efficiency provide the most effective known ways of evaluating various aspects of software quality control. In general, improving software quality requires two important kinds of process improvement: 1) defect prevention and 2) defect removal.

The phrase *defect prevention* refers to

Table 1: *Averages for Defect Potential*

Requirements defects	1.00
Design defects	1.25
Coding defects	1.75
Documentation defects	0.60
Bad fixes	0.40
Total	**5.00**

Table 2: *Defect Removal Efficiency*

Defect Origin	Defect Potential	Removal Efficiency	Defects Remaining
Requirements defects	1.00	77%	0.23
Design defects	1.25	85%	0.19
Coding defects	1.75	95%	0.09
Documentation defects	0.60	80%	0.12
Bad fixes	0.40	70%	0.12
Total	**5.00**	**85%**	**0.75**

technologies and methodologies that can lower defect potentials or reduce the numbers of bugs that must be eliminated. Examples of defect prevention methods include joint application design, structured design, and also participation in formal inspections[3].

The phrase *defect removal* refers to methods that can either raise the efficiency levels of specific forms of testing or raise the overall cumulative removal efficiency by adding additional kinds of review or test activity. Of course, both approaches are possible at the same time.

In order to achieve a cumulative defect removal efficiency of 95 percent, it is necessary to use approximately the following sequence of at least eight defect removal activities:

- Design inspections.
- Code inspections.
- Unit tests.
- New function tests.
- Regression tests.
- Performance tests.
- System tests.
- External beta tests.

To go above 95 percent, additional removal stages are needed. For example, requirements inspections, test case inspections, and specialized forms of testing, such as human factors testing, add to defect removal efficiency levels.

Since each testing stage will only be about 30 percent efficient, it is not feasible to achieve a defect removal efficiency level of 95 percent by means of testing alone. Formal inspections will not only remove most of the defects before testing begins, it also raises the efficiency level of each test stage. Inspections benefit testing because design inspections provide a more complete and accurate set of specifications from which to construct test cases.

From an economic standpoint, combining formal inspections and formal testing will be cheaper than testing by itself. Inspections and testing in concert will also yield shorter development schedules than testing alone. This is because when testing starts after inspections, almost 85 percent of the defects will already be gone. Therefore, testing schedules will be shortened by more than 45 percent.

When IBM applied formal inspections to a large database project, delivered defects were reduced by more than 50 percent from previous releases, and the overall schedule was shortened by about 15 percent. Testing itself was reduced from two shifts over a 60-day period to one shift over a 40-day period. More importantly, customer satisfaction improved to *good* from prior releases where customer satisfaction previously had been very poor.

Measurement of Defect Potentials and Defect Removal Efficiency

Measuring defect potentials and defect removal efficiency levels are among the easiest forms of software measurement, and are also the most important. To measure defect potentials it is necessary to keep accurate records of all defects found during the development cycle, which is something that should be done as a matter of course. The only difficulty is that *private* forms of defect removal such as unit testing will need to be done on a volunteer basis.

Measuring the numbers of defects found during reviews, inspections, and testing is also straightforward. To complete the calculations for defect removal efficiency, customer-reported defect reports submitted during a fixed time period are compared against the internal defects found by the development team. The normal time period for calculating defect removal efficiency is 90 days after release.

As an example, if the development and testing teams found 900 defects before release, and customers reported 100 defects in the first three months of usage, it is apparent that the defect removal efficiency would be 90 percent.

Unfortunately, although measurements of defect potentials and defect removal efficiency levels should be carried out by 100 percent of software organizations, the frequency of these measurements circa 2008 is only about five percent of U.S. companies. In fact, more than half of U.S. companies do not have any useful quality metrics at all. More than 80 percent of U.S. companies, including the great majority of commercial software vendors, have only marginal quality control and are much lower than the optimal 95 percent defect removal efficiency level. This fact is one of the reasons why so many software projects fail completely or experience massive cost and schedule overruns. Usually failing projects seem to be ahead of schedule until testing starts, at which point huge volumes of unanticipated defects stop progress almost completely.

As it happens, projects that average about 95 percent in cumulative defect removal efficiency tend to be optimal in several respects. They have the shortest development schedules, the lowest development costs, the highest levels of customer satisfaction, and the highest levels of team morale. This is why measures of defect potentials and defect removal efficiency levels are important to the industry as a whole; these measures have the greatest impact on software performance of any known metrics.

Additionally, as an organization progresses from the U.S. average of 85 percent in defect removal efficiency up to 95 percent, the saved money and shortened development schedules result because most schedule delays and cost overruns are due to excessive defect volumes during testing. However, to climb above 95 percent defect removal efficiency up to 99 percent does require additional costs. It will be necessary to perform 100 percent inspections of every deliverable, and testing will require about 20 percent more test cases than normal.

It is an interesting sociological observation that measurements tend to change human behavior. Therefore, it is important to select measurements that will cause behavioral changes in positive and beneficial directions. Measuring defect potentials and defect removal efficiency levels have been noted to make very beneficial improvements in software development practices.

When these measures were introduced into large corporations such as IBM and ITT, in less than four years the volumes of delivered defects had declined by more than 50 percent, maintenance costs were reduced by more than 40 percent, and development schedules were shortened by more than 15 percent. There are no other measurements that can yield such positive benefits in such a short time span. Both customer satisfaction and employee morale improved, too, as a direct result of the reduction in defect potentials and the increase in defect removal efficiency levels.◆

Reference

1. Jones, Capers. Estimating Software Costs. 2nd edition. McGraw-Hill, New York: 2007.

Notes

1. The averages for defect potentials are derived from studies of about 600 companies and 13,000 projects. Nondisclosure agreements prevent the identification of most companies. However some companies such as IBM and ITT have provided data on defect potentials and removal efficiency levels.
2. The normal period for measuring defect removal efficiency starts with requirements inspections and ends 90 days after delivery of the software to its users or customers. Of course, there

are still latent defects in the software that will not be found in 90 days, but having a 90-day interval provides a standard benchmark for defect removal efficiency. It might be thought that extending the period from 90 days to six months or 12 months would provide more accurate results; however, updates and new releases usually come out after 90 days, so these would dilute the original defect counts. Latent defects found after the 90-day period can exist for years, but on average about 50 percent of residual latent defects are found each year. The results vary with number of users of the applications. The more users, the faster residual latent defects are discovered.

3. Formal design and code inspections are the most effective defect removal activity in the history of software, and are also very good in terms of defect prevention. Once participants in inspections observe various kinds of defects in the materials being inspected, they tend to avoid those defects in their own work. All software projects larger than 1,000 function points should use formal design and code inspections.

Additional Reading

1. Boehm, Barry W. Software Engineering Economics. Prentice Hall, Englewood Cliffs, NJ; 1981.
2. Crosby, Philip B. Quality Is Free. New American Library, Mentor Books. New York: 1979.
3. Garmus, David, and David Herron. Function Point Analysis. Addison Wesley Longman, Boston: 2001.
4. Garmus, David, and David Herron. Measuring the Software Process: A Practical Guide to Functional Measurement. Prentice Hall, Englewood Cliffs, NJ: 1995.
5. Grady, Robert B., and Deborah L. Caswell. Software Metrics: Establishing a Company-Wide Program. Prentice-Hall: 1987.
6. International Function Point Users Group. IT Measurement. Addison Wesley Longman, Boston: 2002.
7. Jones, Capers. Applied Software Measurement. 3rd edition; McGraw-Hill, New York: 2008.
8. Jones, Capers. "Sizing Up Software." Scientific American New York: Dec. 1998.
9. Jones, Capers. Software Assessments, Benchmarks, and Best Practices. Addison Wesley Longman. Boston: 2000.
10. Jones, Capers. Conflict and Litigation Between Software Clients and Developers. Software Productivity Research, Burlington, MA: 2003.
11. Kan, Stephen H. Metrics and Models in Software Quality Engineering. 2nd edition. Addison Wesley Longman, Boston: 2003.

About the Author

Capers Jones is currently the president of Capers Jones and Associates, LLC. He is also the founder and former chairman of Software Productivity Research (SPR) where he holds the title of Chief Scientist Emeritus. He is a well-known author and international public speaker, and has authored the books "Patterns of Software Systems Failure and Success," "Applied Software Measurement," "Software Quality: Analysis and Guidelines for Success," "Estimating Software Costs," and "Software Assessments, Benchmarks, and Best Practices." Jones and his colleagues from SPR have collected historical data from more than 600 corporations and more than 30 government organizations. This historical data is a key resource for judging the effectiveness of software process improvement methods.

Software Productivity Research, LLC
Phone: (877) 570-5459
Fax: (877) 570-5459
E-mail: cjonesiii@cs.com