The Dollars and Sense of Software Quality Control

By Thibault Dambrine

Software quality is hard to quantify. How would you justify the value of a separate software Quality Control (QC) team? If your IT manager told you "We have survived for so long without such a staff. Why change now?" What could you say to convince him or her to think differently?

Simply put, the answer to all these questions lies in "economics." Boiling it down to dollars and cents, quality software happens to cost less—by a wide margin. In this article, I will expose the economics of software quality. In effect, I will describe how one can put a dollar value on quality control and its impact on the enterprise. Should your IT department consider investing in a QC team? Read on!

The aim of this article is to promote an enhanced understanding of how improving software quality can boost your status as a software producer and help cut development costs at the same time. It will describe:

- The components of software quality cost
- Using a software quality costing model, to make a case for a formal quality control process
- Through the above case, an understanding of how software quality can help drive down software costs while improving the perception of the IT department – as one that, more often than not, hits the mark the first time.

The Cost of Quality

Imagine that 100 programs go into production.

- 80 programs install without a flaw
- 20 programs install, but subsequently require remedial actions, fixes, or modifications.

One could say that the first 80 programs complied right from the time of their promotion. They were properly designed, checked, and verified before being promoted, and there is a cost to all this. This cost will be named the "cost of compliance."

The last 20 programs were not perfect. They required extra work because they did not comply with the requirements. The cost of fixing these programs subsequently can best be described as the "cost of non-compliance."

Once all of the programs have reached the point where they are deemed "of good quality," meaning needing no more fixes, the total cost of quality can be summed up with the formula in **Figure 1**.

Let's examine the components of these costs.

In the Price of Compliance, the following activities can be found:

Development Activities

- Staff training
- Requirements analysis
- Early prototyping
- Fault-tolerant design
- Defensive programming
- Accurate internal documentation
- Proper Requirements
- Detailed Design Documents

Quality Control Activities

- Design review
- Code inspection
- Unit testing
- End-to-End testing
- Regression Testing
- Beta testing
- Test automation
- Pre-release testing by staff
- User acceptance testing

The activities above are relatively standard and most developers would be expected to be familiar with each one of these line items.

In the Price of Non-Compliance, the following activities can be found:

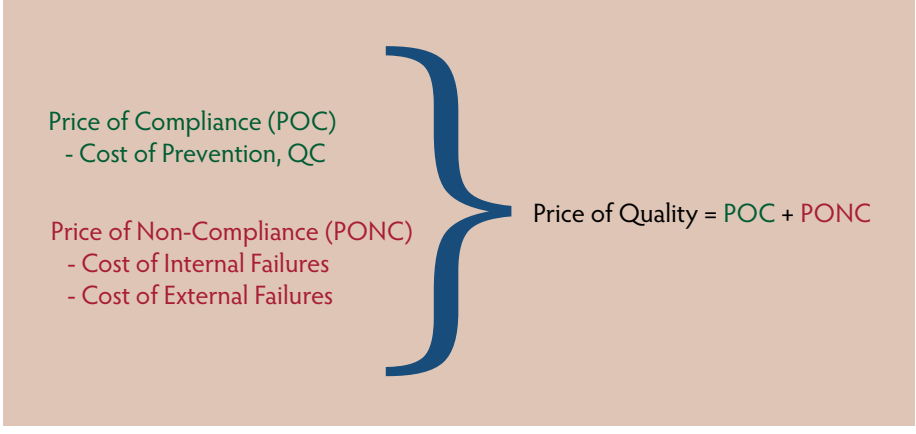

Figure 1.

Internal High Visibility Costs

- Bug fixes
- Wasted in-house user time
- Developer fixing time
- Tester re-testing time
- Cost of late software product shipment
- Receivables potentially affected

External Low Visibility Costs

- Cost of decisions made based on bad data
- Lost Market Share
- Technical support calls
- Investigation of customer complaints
- Refunds and recalls
- Coding / testing of interim bug fix releases
- Shipping of updated product
- Added expense of supporting multiple versions of the product in the field
- PR work to soften drafts of harsh reviews
- Lost sales
- Lost customer goodwill—reputation for producing buggy software
- Discounts to resellers to encourage them to keep selling the product
- Warranty costs
- Liability costs
- Government investigations—if company subject to regulatory rules
- Penalties
- All other costs imposed by law

There are two points to notice here:

1) The list of "internal" or "high visibility" costs is relatively limited and easy to quantify. The costs listed there apply to the IT department only.

2) The list of "external low visibility" costs touches potentially all areas of the company and its impact is far wider than the first list.

What happens here most often is that when evaluating the cost impact of a given bug, the temptation is to only count the "high-visibility" costs. External or low-visibility costs are typically easy to overlook or minimize because they are hard to quantify. They are also commonly known as "soft costs."

The 1-10-100 Rule

How would one quantify the cost impact of these "soft costs"? Many have pondered this question. The accepted measure in most quality circles for this type of cost is the 1-10-100 rule. This rule states that:

- It costs 1 unit of labor to fix a bug at the programmer's workstation.
- It costs 10 units of labor to fix a bug once it has been caught at the Quality Control stage.
- It costs 100 units of labor to the company once a bug has been released to the user.

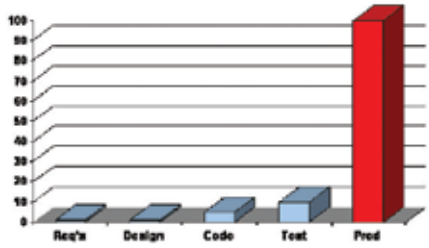

Figure 2.

If you look at the graph in **Figure 2**, you can see how the cost of bugs can grow exponentially if they are not caught

Table A. Cost of Bug Fixes Without Quality Control:

Cost of Resolving Bug	Cost of Resolving a Bug Immediately	Cost of Resolving a Bug at QC	Cost of Resolving a Bug Once it Reached the Users	Total Cost of Bug Fixes
1-10-100 Rule	**1**	**10**	**100**	
Distribution of 125 Bugs	**100 x $100 x 1**	**0 x $100 x 10**	**25 x $100 x 100**	
Distribution of Costs @ $100/bug	**(100x$100 x 1) $10,000**	**(0x $100 x 10) $0**	**(25 x $100 x 100) $250,000**	**$260,000**
NO QC TEAM COSTS				**$0**
Total Cost				**$260,000**

Table B. Cost of Bug Fixes With Quality Control:

Cost of Resolving Bug	Cost of Resolving a Bug Immediately	Cost of Resolving a Bug at QC	Cost of Resolving a Bug Once it Reached the Users	Total Cost of Bug Fixes
1-10-100 Rule	**1**	**10**	**100**	
Distribution of 100 Bugs	**100 x $100 x 1**	**20 x $100 x 10**	**5 x $100 x 100**	
Distribution of Costs @ $100/bug	**(100x$100 x 1) $10,000**	**(20x $100 x 10) $20,000**	**(5 x $100 x 100) $50,000**	**$80,000**
+ Cost of QC Team				**$75,000**
Total Cost				**$155,000**

early. Clearly, it costs less to catch bugs early. To illustrate this concept with a concrete example, I will put forward the following model:

The Case for a Formal Quality Control Process

We now know more about the components of software quality and the fact that bugs cost a lot less if they are caught early than if they are caught late. The following section will illustrate, with a simplified example, the impact of resolving bugs early on software development costs.

Cost of Bug Fixes Without Quality Control (See Table A.)

Let there be:

- An IT department that DOES NOT have any formal Quality Control personnel or process
- 125 bugs a year, 80% caught by developers
- Each bug costs $100 to fix at the developer workstation.
- NO separate QC cost involved, as per point (1).
- 80% of 125 bugs is 100 bugs, resolved at $100 times a factor of 1, totaling $10,000.
- 20% of 125 bugs is 25 bugs, resolved at $100 times a factor of 100, totaling $250,000.
- There are no expenses for quality control.
- The total is $260,000.

Cost of Bug Fixes With Quality Control (See Table B.)

Let there be:

- An IT department that has a formal Quality Control person, 125 bugs a year, 80% caught by developers.
- Each bug costs $100 to fix at the developer workstation.
- Full time equivalent cost for the Quality Control person: $75K/annum. This person has an 80% bug catch rate.
- 80% of 125 bugs is 100 bugs, resolved at $100 times a factor of 1, totaling $10,000.
- 80% of the remaining 25 bugs are 20 bugs, resolved at $100 times a factor of 10, totaling $20,000.
- The remaining 5 bugs reach the customers and are resolved at a cost of $100 times a factor of 100, totaling $50,000, plus $75,000 for a QC resource and the total comes up to $155,000.

The Big Picture

The total return on investment, after spending $75K on a single quality control resource is an astounding 140% return. Even if the cost of this QC resource was 100K, the return would still be over 100%. **Figure 3** illustrates the cost curve of quality without QC vs. the cost curve with QC.

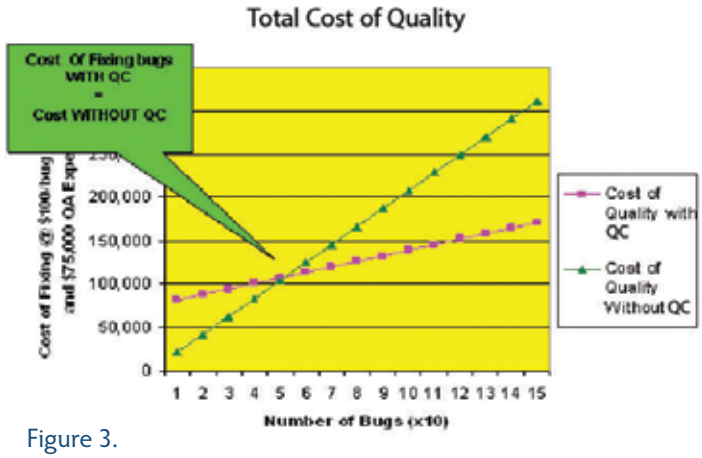

Figure 3.

- The green line shows the cost of quality without QC. In simple terms, every new bug brings an equal amount of cost. (This is a deviation from the 1-10-100 rule, which would make the slope steeper.)
- The pink line shows the same cost curve with QC. Note that it starts at 75K, which is expensive if you have no bugs to start with, but it has a significantly softer slope than the green line.
- The point where these two lines meet is around 55 bugs. At that point, it costs no more, no less to have or not have a QC resource.

The graph in **Figure 4** shows three curves:

- The blue line is an imaginary one: It simply is one that shows, "What if users could see every bug?"
- The green line shows what they really see after the developers have cleaned up 80% of these bugs.
- The pink line below shows what if QC caught 80% of the bugs below the green line.

At the 55 bug point, previously identified as the point where the cost of fixing bugs with QC would be the same as the cost of fixing bugs with QC, a point where there was—on the surface, no advantage to a QC resource.

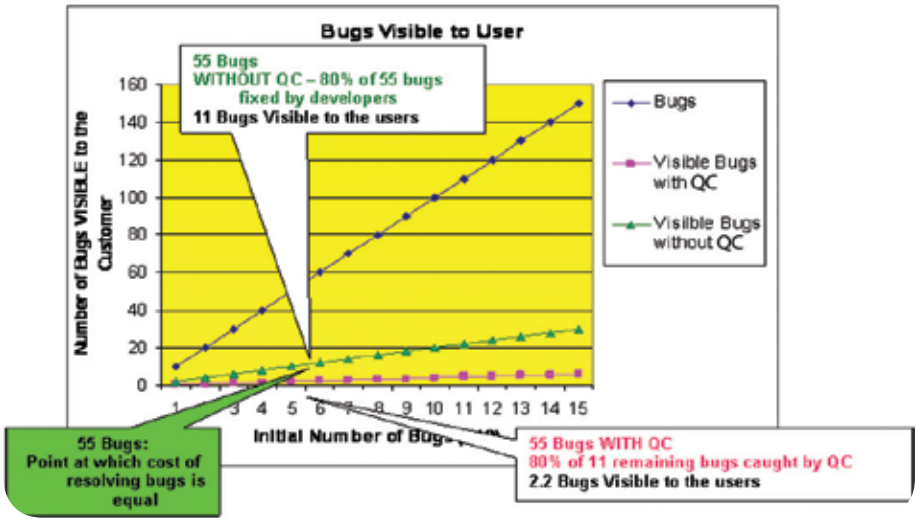

Figure 4.

The visibility curves tell a different story. At the 55-bug data point, without QC, only 80% of these bugs will be resolved—the user will see 11 bugs. With a QC resource added, at that same point, 80% of remaining 11 bugs will be resolved—letting only 2.2 bugs reach the user.

This difference, 11 bugs down to 2.2, can make a phenomenal difference in positive perception from the users towards the IT department.

I used the 80% mark here to clearly make a point. This percentage, for any real life modeling, should be higher. No matter what the percentage is however, the 1-10-100 rule will still show a clear cost and improved user perception for the IT departments that have formal quality processes.

Being consistently reliable when delivering results means an IT department will:

- be trusted to produce reliable content,
- experience fewer service interruptions for emergency fixes,
- command premium budget dollars for salaries and equipment—it will be perceived to be money well spent,
- and get stronger business buy-in when requesting investments in new technology or new staff.

In conclusion

IT is typically a cost center. Understanding that software quality as well as software costs impact the company as a whole is critical to understanding the valuation of quality. Simply put, it goes beyond the boundaries of the IT department.

Bad software quality can adversely affect the entire company in an almost infinite number of ways. Good quality, on the other hand, not only costs less, it also enhances the confidence, trust, and positive perception the corporation can have towards its IT department. Companies boasting such quality IT services will also typically have a bolder attitude towards technology, more open to evaluating new software products ahead of their competitors. These are no small benefits!

If you are wondering if your IT department could benefit from a formal QC process, look at your bug track. Look at your bug fixing costs and apply the 1-10-100 rule using a bug resolution percentage that would be representative of how your developers perform. Use the model provided with this article and see where you could gain both savings and reliability. Ask yourself: What would your QC/QA-driven savings be like? Could your bug visibility curve be pushed down?

About the Author

Thibault Dambrine works for Shell Canada Limited as a senior systems analyst. He holds the ITIL Foundations as well as the Release and Control Practitioner's Certificates. His past articles can be found at www.tylogix.com.

LUG Conferences

Wisconsin Midrange Computer Professional Association (WMCPA)
23 Annual Spring Technical Conference
April 9-10, 2008
Geneva Resort, Lake Geneva, WI

Northeast IBM User Group
18th Annual Conference
www.neugc.org
April 14–16, 2008

Toronto Users Group for System i
TEC 2008
www.tug.ca/tec
April 22–24, 2008
Richmond Hill, ON, Canada

Southeast Michigan iSeries Users Group
MITEC – Michigan iSeries Technical Education Conference
www.semiug.org
June , 2008
Southeast Michigan

OCEAN User Group
OCEAN – 15th annual "Catch the Wave" Technical Conference & Vendor Solution Expo
www.ocean400.org
July, 2008 Irvine, CA