Vector Software

WHITEPAPER

Quantifying The Cost of Fixing vs Preventing Bugs

When you think about improving software quality, your first thoughts might be the cost of new tools and engineering labor to implement them, as well as the impact of “slowing down” the development cycle by adding new processes. But really, you need to first think about the current cost to your organization to find and fix bugs over the product life cycle. Beyond these direct costs, have you considered the opportunity cost associated with missed release dates, or the good will that is lost from unhappy customers, or product recalls?

In the book “How Google Tests Software – Help me test like Google” the authors describe the evolution of the Google™ software testing process from the early days until now[i]. Bugs that resulted from incomplete testing had become one of the biggest barriers to Google’s continued success. Google was able to quantify the cost of their poor quality in terms of the number of engineers involved in fixing bugs. They then put a plan in place to improve software quality through continuous testing, and by testing earlier in the software development lifecycle, thus freeing up those same engineers to innovate new features and beat their competitors to market.

Here is an example of what this calculation might look like for a mid-sized automotive project, on a yearly basis, using some of Google’s numbers:

1.	Source Lines of Code (KSLOC) Generated Per Year		200
2.	Average Bugs Per 1000 SLOC[ii]	x	8
3.	Number of Bugs in Code	=	1,600
4.	Average Cost to Fix a Bug[iii]	x	$1,500
5.	Total Yearly Cost of Bug Fixing	=	$2,400,000
6.	Yearly Cost of an Engineer[iv]	/	$150,000
7.	Number of Engineers Consumed with Bug Fixing	=	16
8.	Engineering Team Size	/	40
9.	Percentage of Staff Used for Bug Fixing	=	40%

Table 1: Metrics from Google

Plug some numbers into the following table to estimate your costs:

1.	Source Lines of Code (KSLOC) Generated Per Year		______________
2.	Average Bugs Per 1000 SLOC	x	8
3.	Number of Bugs in Code	=	______________
4.	Average Cost to Fix a Bug[v]	x	______________
5.	Total Yearly Cost of Bug Fixing	=	______________
6.	Yearly Cost of an Engineer[vi]	/	______________
7.	Number of Engineers Consumed with Bug Fixing	=	______________
8.	Engineering Team Size	/	______________
9.	Percentage of Staff Used for Bug Fixing	=	______________

Table 2: Metrics for your project

It’s pretty likely that when you run through this exercise you’ll find that your team is spending 30-50% of their time fixing bugs, and that’s just the direct cost. Imagine if you could use more of that time for product innovation, and new feature implementation. Imagine if you could reduce your customer support costs because fewer bugs get to customers.

Imagine if your customers were happier and bought more of your products!

Organizations like Google, who have successfully embraced a continuous software quality approach, have determined that implementing automated testing into the development cycle pays huge dividends. A key element in their approach is the test platform that they have created to ensure that bugs are found as early in the development process as possible – when they are cheapest to fix.

Here are some sample costs, from Google, of a bug found in different phases of testing which clearly shows that finding bugs earlier in the development process translates to a lower cost per bug[vii].

Software Testing Phase Where Bug Found	**Estimated Cost per Bug**
System Test	$5000
Integration Test	$500
Full Build	$50
Unit Test / Test Driven Development	$5

Table 3: Costs to Fix Bugs at Google

Now that you have a better idea of what bug fixing is costing your team, what is the best way to reduce bugs, and improve quality? Well the average cost to find and fix a bug is not going to go down over time, but improving and automating your testing process will reduce your overall maintenance cost. The following graph shows how creating a more robust test process will lead to reduced software maintenance costs.

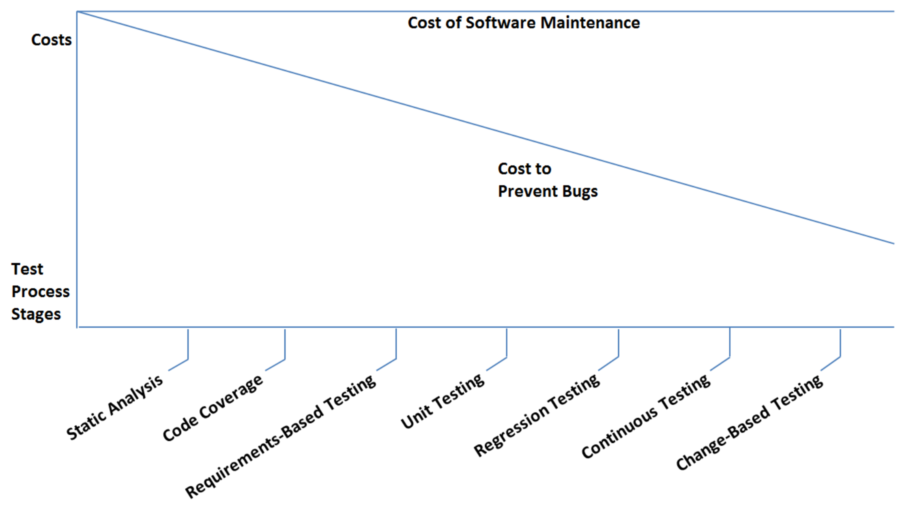

Figure 1: The cost of software maintenance drops as the software test process matures

Organizations find themselves at all different spots in this spectrum; some may have no formalized testing infrastructure, and others, such as Google, may have already solved this problem. Most likely, your team falls somewhere between those two extremes.

Where should you start on this journey to improved quality? As with most engineering tasks, a measured, step-wise process is the most effective way to improve. Each change should build on the foundation of the previous improvements, as shown in the following graphic.

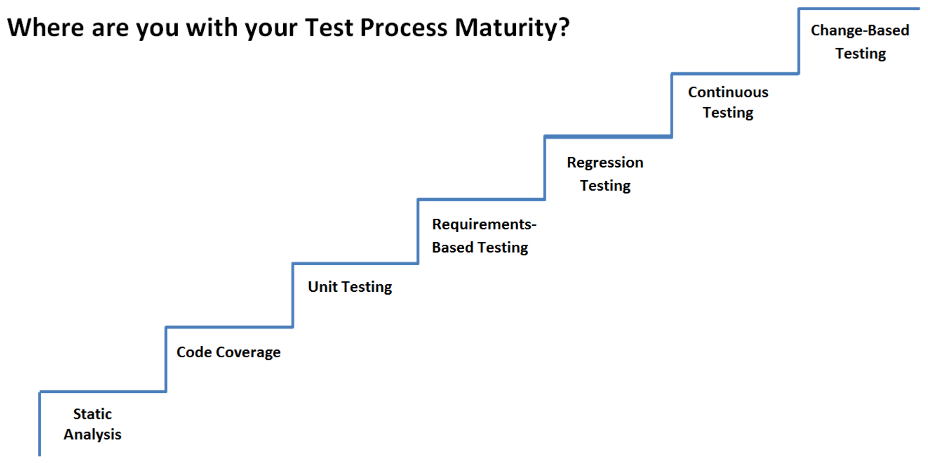

Figure 2: Adopting a software test process can be easily achieved in a stepwise fashion

By adopting an approach that embraces the value of prevention, instead of fixing, your organization can free up valuable engineering resources for innovation and new product development, enabling you to carve out a greater share of the markets you serve.

A great starting point is to introduce static analysis and code coverage tools. These tools can be implemented without changing the workflow of your development team, but will yield actionable metrics to help improve quality. The VectorCAST/Lint static analysis tool reports on coding constructs that have historically proven to be problematic, and hard to maintain. The VectorCAST/Cover code coverage tool reports on which portions of your source code have been tested, and more importantly which portions have not been tested.

Let us help you figure out where you are with your Software Test Process Maturity. We can work with your team to understand the unique challenges you're facing, and help you design a process improvement plan to prevent more bugs, and improve time to market and profitability, backed by a solid Return on Investment.

Google is a trademark of Google Inc. All rights reserved. Google and the Google Logo are registered trademarks of Google Inc.

---

[i] Whittaker, James , Jason Arbon, Jeff Carollo. How Google Tests Software. Westford, MA: Addison-Wesley, 2012.
[ii] Copeland, Patrick (2010). Innovation Factory: Testing, Culture, & Infrastructure (PowerPoint slides). Retrieved from http://googletesting.blogspot.com/2010/04/googles-innovation-factory-and-how.html. Slide 33.
[iii] Ibid, Slide 29.
[iv] Burridge, Brian. When calculating development costs, the hourly rate is only half the equation. August 26, 2010. Brian Burridge blog.
[v] Ibid, Slide 29.
[vi] Burridge, Brian. When calculating development costs, the hourly rate is only half the equation. August 26, 2010. Brian Burridge blog.
[vii] Copeland, Patrick (2010). Innovation Factory: Testing, Culture, & Infrastructure (PowerPoint slides). Retrieved from http://googletesting.blogspot.com/2010/04/googles-innovation-factory-and-how.html. Slide 29.

About Vector Software

Vector Software, Inc., is the leading independent provider of automated software testing tools for developers of safety and business critical embedded applications. Companies worldwide in automotive, aerospace, medical devices, industrial controls, rail, and other industries, rely on Vector Software's VectorCAST™. By automating and managing the complex tasks associated with unit, integration, and system level testing, VectorCAST helps organizations accelerate the development and ensure the reliability of their embedded software applications.

Vector Software, Inc.
1351 South County
Trail, Suite 310
East Greenwich, RI 02818 USA
T: 401 398 7185
F: 401 398 7186
E: info@vectorcast.com

Vector Software
Golden Cross House
8 Duncannon Street
London WC2N4JF, UK
T: +44 203 178 6149
F: +44 20 7022 1651
E: info@vectorcast.com

Vector Software
St. Töniser Str 2a
47906 Kempen Germany
T: +49 2152 8088808
F: +49 2152 8088888
E: info@vectorcast.com

Vector Software
Rm 403, Building 6, No.88
Daerwen Rd, Zhangjiang
Hi-tech Park Pudong New Area
Shanghai 201203 China
T: 21- 3126 8126
F: 21-5132 8526
E: info@apac.vectorcast.com