The Proven ROI of Development Testing: An in-depth analysis of Coverity customer experiences
Introduction
Any development organization that is serious about software quality immediately sees the benefits of static analysis solutions for development testing. These have been shown to find more bugs and identify problems more quickly than any other method, while at the same time delivering increased developer productivity, faster time to market, and decreased risk of brand damage. Only one question remains: Do these benefits justify the cost and effort of adopting a new technology?

This paper, based on in-depth interviews with 12 Coverity customers from industries including biotechnology, financial services, medical devices, telecommunications, manufacturing, SaaS applications, energy, communications, independent software vendors (ISVs), technology service providers, and safety critical systems, investigates that question by looking at the actual value achieved in real-life implementations.

These Coverity customers consistently reported that the return on their development testing investment is real and substantial. This paper describes the various types of benefits reported by Coverity customers and takes an in-depth look at the most frequently reported areas of value including:

- Preventing defects
- Increasing developer productivity
- Reducing headcount costs
- Accelerating time-to-market

Participant Profile
The 12 participants represented a mix of individuals with responsibility for managing development teams, hands-on code development, quality assurance, and security. Participants worked with a wide range of languages including C/C++, C#, and Java. Anywhere from as few as four to as many as hundreds of developers regularly used Coverity in these organizations to streamline and optimize development testing of millions of lines of code produced either internally or by outsourced teams.

All participants were employed by companies that write software for a wide range of industries and products. Both enterprise application and embedded software development projects were represented, with the companies in a wide range of industries.

Quantifying the Hard and Soft ROI of Coverity
In today’s economy, every dollar spent must be justified. The development organizations that build the software used for devices and applications are no exception. All development and quality tool investments must deliver significant value to the organization. Static analysis solutions are subject to the same scrutiny, which has become even more intense in recent years. Participants reported they had ongoing discussions with their finance teams about the benefits provided by Coverity and needed to demonstrate the business value of the investment.

“It’s hard to convince financial guys to spend money. The in-house success we’ve had with Coverity simplifies that process tremendously as we’ve increased our investment.”

The participants in this study consistently reported that Coverity’s solutions deliver real and substantial benefits, although approaches to evaluating ROI varied depending on the specific business needs of each company. Coverity customers saw clear, hard ROI in the following areas:

- Preventing defects that would be costly if they reached customers
- Improving quality to protect brand reputation
- Increasing developer productivity
- Replacing expensive and incomplete manual code reviews
- Accelerating time-to-market

In addition, participants saw many soft benefits including increased development transparency, improved coding skills, and better team management.

The ROI of Defect Prevention
The cost savings of finding a bug early in the development cycle are well understood. The longer it takes to find a problem, the more expensive it is to fix, with the most costly bugs being the ones
that reach customers. The savings of preventing field-found defects is huge, both in terms of the cost required to fix a problem once it reached customers as well as the potential for serious reputational damage. Since Coverity highlights problems very quickly, the cost savings identified were very substantial.

“We brought in Coverity for a demo and they did a quick analysis of some code we had out in the field that we thought was pretty good. The developers in the meeting turned a funny shade of white when they saw the list of bugs just waiting to trip at any time. That was our business case right there.”

Calculating the exact cost savings varied depending on the maturity of the software being tested, although the basic formula was the same:

\[
\text{Number of bugs found by Coverity} \times \text{Cost if that bug reached the field}
\]

Participants who had an existing release of their software in the field found it easiest to accurately calculate the ROI of preventing defects. Since they had a history, these customers were able to base their analysis on real problems that had occurred prior to adopting the Coverity solutions, including a calculation of the actual costs to fix bugs that reached their end users and the impact on future business and reputation.

These participants started by evaluating all significant bugs over the one-year period prior to the Coverity purchase. Of those bugs, they identified the ones that Coverity would have found if the solution had been in place. Finally, they calculated the actual cost to fix those specific bugs.

“We looked at last year and pointed to particular problems that Coverity would have found and what they cost. We were confident of the figures because the problems had actually happened. The savings Coverity would bring by preventing just a few of those defects in the field made for a very simple cost justification for us.”

For participants working on a new development project, the ROI calculation involved a few more steps. These customers started by taking any numbers they knew—typically their average number of bugs per lines of code, number of lines of code, etc. For the numbers they did not know, these customers use a range and calculated ROI using the most conservative end of the range.

“We know the number of bugs that Coverity usually finds in a certain number of lines of code. We know the number of lines of code on this project so we can calculate the number of bugs found that way. Then we make a very conservative estimate on the cost of preventing a bug and it’s obvious that the value is there.”

Some participants looked back after using Coverity for a significant period of time to establish past ROI. That calculation was particularly important in justifying increased solution investment or ongoing use. These participants had tracked the number of bugs reported in the field over the entire life of the product and were able to see that the trend started to decrease substantially at the time that Coverity was adopted.

Other participants did not specifically look at the ROI of increased quality using a detailed calculation, citing that the value in avoiding preventing potentially embarrassing issues was enough of a business justification. Reputation was very important among these participants.

“Our main driver in using Coverity is our reputation. We market performance, so being able to achieve that at the same time we increase functionality is important from a customer perspective. Coverity has found things that were so deeply buried we wouldn’t have known about them otherwise, no matter how much effort we put into it. It’s been a huge improvement to our code efficiency.”
The ROI of Developer Productivity

Increased developer productivity was another key metric used by participants to establish ROI. The formula used was straightforward:

\[
\text{Number of bugs found by Coverity} \times \text{Number of days to fix the bugs} \times \text{Daily developer cost}
\]

All participants knew their daily developer cost. Those who had projects in process were able to use the number of bugs they had historically. Those who were starting a new project used estimates based on similar experiences.

“Sometimes you have a bug that comes from a customer that is very hard to find. If we clean up even a few of those with Coverity, it's a good return.”

The least straightforward number in this calculation was the number of days to fix the bugs. Each bug is different with a widely varying range of time required to fix. For this number, participants established a range, and then used a conservative estimate in their ROI calculations. In general, customers found that even with using the low end of the likely range it was pretty easy to show that it didn’t take long to pay off the investment in Coverity.

“The actual development effort to fix a bug is somewhere between one minute and a few weeks. We made a very low assumption that each bug will take a few days to fix. We know that is low but we want to be cautious so the financial guys will follow the calculations. We don’t need the exact number because we know that even at the conservative number we’re getting a lot of value.”

Participants who had been using Coverity over a period of time were able to calculate very precise ROI numbers. Using the exact number of issues detected by Coverity - information directly available from Coverity reports - gave an exact number.
# Building the Financial Business Case

The participants in this study did point out one difficulty in calculating a hard ROI for static analysis tools: the need to put a dollar value on the real cost of a defect that never happens. The Coverity investment is justified by the bug that doesn’t occur, is never experienced by a customer, and doesn’t incur costs that can be measured.

That said, participants were required to come up with a defendable dollar value to establish a business case for purchasing and renewing their Coverity licenses. This is how participants approached the calculation, including actual numbers reported by participants:

> “I can’t measure the exact cost of bugs that the tool prevented and thus never happened. But we put a very high price tag on defects that get out to a customer. If we lose confidence, any bug is easily tens of thousands of dollars in business impact.”

---

### DEFECT PREVENTION SAVINGS

<table>
<thead>
<tr>
<th><strong>Average number of defects found per 10,000 lines of code</strong></th>
<th>Participants reported Coverity identified 3 to 10 defects per 10,000 lines of code</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of lines of code</strong></td>
<td>Per project number</td>
<td>1 million</td>
</tr>
<tr>
<td><strong>Total number of defects</strong></td>
<td>Divide the number of lines of code by 10,000 and multiply by average # of defects</td>
<td>300</td>
</tr>
<tr>
<td><strong>Average cost of a defect that reaches the field</strong></td>
<td>Participants reported from $2,000 to $30,000 to fix a field-found bug</td>
<td>$2,000</td>
</tr>
<tr>
<td><strong>Total Annual Savings from Preventing Defects</strong></td>
<td>Multiply above two lines</td>
<td>$600,000</td>
</tr>
</tbody>
</table>

### DEVELOPER PRODUCTIVITY SAVINGS

<table>
<thead>
<tr>
<th><strong>Number of critical defects detected by Coverity</strong></th>
<th>For new Coverity purchase, pull reports from bug tracking tool; for renewal pull number from Coverity reports. Participants typically reported hundreds of bugs</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average time to fix a defect before Coverity</strong></td>
<td>Time to fix depends on the bug, so use a conservative average based on past experiences in your business</td>
<td>16 man-hours/bug</td>
</tr>
<tr>
<td><strong>Average time to fix a defect after Coverity</strong></td>
<td>Identifying a bug when it is still fresh in a developer’s mind makes it much quicker to fix problems</td>
<td>2 man-hours/bug</td>
</tr>
<tr>
<td><strong>Total man-hour savings</strong></td>
<td>Multiply number of bugs by the difference in the time to fix a defect before and after Coverity</td>
<td>2800 man-hours</td>
</tr>
<tr>
<td><strong>Fully loaded cost of a developer</strong></td>
<td>Get number from HR or development manager</td>
<td>$100/hr</td>
</tr>
<tr>
<td><strong>Total Annual Savings from Increased Developer Productivity</strong></td>
<td>Multiply above two lines</td>
<td>$280,000</td>
</tr>
</tbody>
</table>

### AUTOMATION SAVINGS

<table>
<thead>
<tr>
<th><strong>Number of staff required for manual code reviews</strong></th>
<th>Participants reported 10-15 FTEs to do what Coverity can do</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Annual fully loaded cost headcount for code review</strong></td>
<td>Get number from HR or development manager</td>
<td>$120,000</td>
</tr>
<tr>
<td><strong>Total Annual Savings from Automation</strong></td>
<td>Multiply above two lines</td>
<td>$1,200,000</td>
</tr>
</tbody>
</table>

### Total Savings

Add three areas of savings above                                                                 $2,080,000
“In the first nine months we had Coverity, it identified 167 issues that were either showstoppers or could have led to serious issues in the field. We used the very defensible number of two days saved fixing each of these bugs without Coverity, to show we saved over 2,500 man-hours of development time.”

Many participants emphasized that developer productivity covered more than just preventing the effort to fix bugs that didn’t occur. Coverity provided an extremely important benefit by giving developers immediate feedback on their code. Coverity significantly simplified fixing problems by identifying errors at a time when the code was still fresh in the developer’s mind and they had the greatest understanding of what they had just written. Coverity was described as giving almost real-time feedback. Participants described their developers as a very costly resource, and being more effective with their time delivered significant business benefits.

“Rather than waiting to find a problem and only then wading through thousands of lines of code, Coverity lets you see the problem right away and fix it while it’s still in your mind. It easily saves a developer several weeks of bug-fix time over the course of a year.”

Participants also emphasized that code reviews were streamlined once they began using Coverity. The teams had high confidence that certain kinds of errors would always be found by Coverity—for example, null pointer defects—so they had the confidence to put their efforts in other areas. For the same reason, knowing that certain types of errors did not exist made the process of fixing problems more focused.

The ROI of Automation

Some participants calculated the value of Coverity by comparing it to the cost of getting the same results using manual code inspection. This calculation was reasonably straightforward and easily demonstrated the value of an automated approach to code review. They simply took the additional headcount required to perform code inspection, calculated the fully loaded cost, and compared it to the Coverity investment. This calculation often happened at the prompting of a finance team who suggested that hiring might be a better option. However, participants described this conversation as very short because it was straightforward to show that Coverity excelled in each of the three key areas of evaluation: it could find more defects, at a faster rate, for less money than manual code inspection.

“Our current rule of thumb is that the price of a system test person is about the same as Coverity. But a system test person finds a serious problem at the rate of 2 or 3 per week and Coverity finds serious problems at the rate of 10 per week on the same code.”

The discussion of Coverity compared to people resources always mentioned that automated tools will always do a better job at coverage and catching defects than is possible for even the best people. Each participant had their own anecdote of something simple that Coverity found immediately that the development teams had missed, even in large review sessions. These examples ranged from a missing semi-colon to a logic problem that mistakenly used AND instead of OR.

One of the key areas where Coverity was able to excel over manual approaches was in analyzing code paths. Coverity automatically exercises all possible paths in the code, including error paths and other areas of the software that were not used as frequently and therefore not tested as extensively using other methods.

“It was definitely cheaper to buy Coverity than it was to hire 10 to 15 people to do the same thing. More than the cost, there are better things to do with people. Even if I had a staff of 15 consistently look at code they wouldn’t find the same number of things as Coverity because they’re not robots.”
Several participants described a typical problem where the human mind will fill-in information that it expects to see and has difficulty even seeing small errors in code. For example, a person will frequently miss incorrect punctuation, a very simple and basic error that an automated tool will catch consistently.

“The way Coverity looks at software can be mind blowing. I’ve seen a team of 3 top developers look at a defect for 2 hours before they understood it, where Coverity finds it right away. You’ll never get that out of one person, no matter how bright they are.”

The ROI of Improved Time-to-Market
Certain participants quantified the value of Coverity in the ability to accelerate their time-to-market. Coverity was able to analyze, find problems with code, and suggest fixes much faster than any other method. This speed helped these participants ship products sooner in order to meet market need and get ahead of competitors. For these participants, the time-to-market savings alone more than justified the Coverity purchase.

“We had a real business need because the market was waiting for our solution and we were not able to ship because of quality issues. Coverity helped us to ship faster and gain an important new revenue stream.”

Participants who worked with outsourcing providers to develop code were particularly impressed by the impact of Coverity. While the partners did have SLAs for code quality, it was the reputation of their own company that was ultimately on the hook if any defects reached the field. The ability to quickly check code using Coverity had enormous value for increased quality, preventing field-found defects, all while maintaining tight development schedules.

“We were given ½ million lines of code and three days to check them. We found multiple show stoppers that we were able to fix and still ship on schedule. There is simply no way we could have done this before Coverity.”

The “Single Bug” ROI
There were some cases where participants did not go through an extensive ROI calculation because they could easily articulate the value they received from Coverity by citing a single, extreme example. These examples were typically highly specific to a customer’s business, and in general happened among companies that had tight service-level agreements (SLAs) with their customers, businesses where software defects could result in safety issues, or vendors whose reputation was strongly based on delivering quality.

“We were literally in the situation where one bug that Coverity found paid the yearly fee.”

“Field defects are expensive for us. As soon as there is a failure, the clock starts ticking on our SLAs. We do have $1 million defects in our business so poor quality is a real cost.”

“If a defect slips through to a customer, then any company has loss of confidence which is bad. But because we build safety systems, in our case it can mean death. That is just not an option.”
The ROI of Compliance

Several participants cited that one of the benefits they achieved from using Coverity was improvements in adhering to compliance requirements including ISO, SEC, local safety standards, or any compliance requirement that included software development. The ability to demonstrate the use of advanced tools to find complicated bugs was described as creating significant confidence with auditors, particularly among participants in financial services, communications, and safety critical industries. Other participants described that a key step in meeting compliance requirements was having software development lifecycle processes that verified the code they generated was reliable. Using Coverity became a documented way of demonstrating those efforts and meeting that compliance check, although none of the participants in this study were required to use Coverity to directly meet their compliance requirements.

“Having the ability to show reports and trends is a powerful thing. These Coverity reports are what we’ll show the auditors.”

ROI Bonus: The Soft Benefits of Coverity

Participants reported they were expected to show hard ROI for their static analysis purchases and they did, as shown in the specific examples above. However, participants were consistently unwilling to limit the ROI discussion to a conversation just about the numbers. While our original line of questioning focused very specifically on hard ROI, the customers in this study went out of their way to emphasize the softer benefits of using Coverity that were more difficult to quantify.

Reporting and Transparency

Coverity users were very enthusiastic about the benefits of reporting and the transparency the reports gave to their organization. Most participants had Coverity integrated into their build systems. Each morning, the latest report data created an immediate feedback loop of information on quality and productivity metrics for the entire team.

“Making metrics visible is a good thing.”
Improved Coding Skills
Participants repeatedly emphasized the value that Coverity brought to teaching developers to code more cleanly. There were clear improvements in the development abilities of the teams using Coverity that happened in an implicit way rather than explicitly forcing them to do things a certain way or sending them to classes in order to improve their code.

“The effect at the developer’s desk is that the coding style will change. If you introduce the same error and Coverity tells you about it twice, the third time you’re more cautious.”

Better Team Management
An interesting benefit of using Coverity that was cited by the managers we spoke to in this project was the increased insight into what was going on with individuals and teams. Coverity was an interesting metric to uncover behaviors that could have otherwise been missed by management. Any differences in the number of defects found between developers who otherwise had comparable projects could indicate a need for additional training or mentoring, or a possible promotion for the more skilled developers.

“If one of our teams has a much higher defect count than the other teams, we know we need to look and see if there is a staff development problem or maybe we’re pushing that team too hard. On the flip side, if a team produces Coverity-perfect code regularly, then you can consider pushing a bit more.”

Increased Developer Job Satisfaction
An interesting take on developer productivity experienced by Coverity users was related to job satisfaction. Multiple participants described scenarios before using Coverity where there were certain nasty bugs that were extremely difficult to find and caused a very high frustration level among the developers attempting to solve them. A particularly nasty problem could keep an entire team distracted for a week or more. Since finding the source of these bugs was so much easier with Coverity, the frustration level among the developers was reduced, productivity increased, and work satisfaction was higher.

“The developers who create the code know that Coverity makes their jobs easier. If our Coverity server isn’t available for some reason, we hear from the developers asking when they’ll get the next analysis.”

Conclusion
All twelve participants in this study found significant ROI in their Coverity investment. Value came from improved quality and increased developer productivity, as well as a range of other benefits. Each person we talked to had a different way of calculating their business’s specific value, but all were able to articulate a strong case for using Coverity and plan to continue using it for future development projects.
Zetta.net’s Commitment to Development “Hygiene”

At Zetta.net, the impact of errors is very large. As one of the fastest growing providers of enterprise cloud backup and disaster recovery solutions, they are deeply committed to not losing data ever.

When he started Zetta.net, Co-Founder and Chief Scientist Lou Montulli went looking for the best methodologies he could find for reducing any kind of errors possible. Static analysis represented something the team couldn’t get any other way—an opportunity to have 100% coverage of the code base.

Lou shared many of the experiences of other Coverity customers in describing the value of the tool, but in addition pointed out that static analysis brings an important “hygiene” to Zetta.net’s development. Code that passes through Coverity has a completeness and consistency that is almost impossible to attain when a development team is trying to move as quickly as possible. Over time, the result is a real mess if there is not something in place to prevent that.

"Coverity helps us keep our code in a pristine, clean state. It prevents future errors and improves the readability of the code because of the standards it enforces. It makes it more consistent. When everything is more consistent your development is just better. It especially helps in a team because one programmer can understand and view the code of another one."

- Lou Montulli, Co-Founder and Chief Scientist, Zetta.net

For Lou and the Zetta.net development team, this is a hygiene commitment. Like brushing your teeth, he says, it’s something that isn’t top of mind in any particular moment, but it doesn’t take much time and yields really great results.

Project Methodology

Coverity commissioned Dimensional Research to interview customers who have deployed static analysis solutions and obtain in-depth feedback on the actual value received. Dimensional Research conducted 12 telephone interviews with Coverity customers and this report is a summary of those conversations. All content in this report was taken from interview transcripts, although some quotes have been edited for grammar and readability.

Coverity secured the participation of customers for the interviews, then passed their contact information to Dimensional Research. Coverity did not participate in the interviews.

Participants were not compensated for participating in this research project, although as a token of appreciation for their time a donation was made to the charity of each participant’s choice. Because of the challenges in getting corporate approvals to discuss sensitive development processes publicly, participants were ensured their feedback would be presented as part of a summarized report with no attribution unless they requested otherwise.

About Dimensional Research

Dimensional Research provides practical market research services that help technology companies make smarter business decisions. Our researchers are experts in technology and understand how corporate IT organizations operate. We partner with every client to deliver actionable information that reduces risk and increases customer satisfaction. Our research services deliver a clear understanding of customer and market dynamics. For more information visit www.dimensionalresearch.com.

About Coverity

This paper was commissioned by Coverity, Inc., the development testing leader. Coverity is the trusted standard for companies that need to protect their brands and bottom lines from software failures. More than 1,100 Coverity customers use Coverity’s development testing suite of products to automatically test source code for software defects that could lead to product crashes, unexpected behavior, security breaches, or catastrophic failure. For more information visit www.coverity.com.