XRAY

Native Quality Management:
Build better software - together

Jira

Content

Introduction: The rise of Quality culture vs testing

Historically, software development and quality assurance were one and the same. If you built it, you also tested it. But then software grew up, and as it got more and more complex, dev and QA needed to split up in order to do their job right.

But instead of these two teams remaining close friends, **they grew far apart**. Each in their own world, operating in different environments, using their own workflows, speaking different languages.

Who paid the price? The software. Now, complex, disconnected workflows weigh us down, **slowing down our releases**. Critical tests are missed, **compromising our coverage**. And more than anything else, we're just **not on the same page** - even simple tasks are frustrating when each team needs to translate into their own language just to get stuff done.

"Leaders who embrace this idea- that companies live and die by quality as perceived by their customers and make it their matra will adapt and grow with them. Those who focus on functionality and internal definitions of quality...won't."

Ronald Cummings-John | From the book Leading Quality: How Great Leaders Deliver High-Quality Software and Accelerate Growth

How can we expect our software to be at the highest quality, when quality and development are so disconnected?

In this eBook, you'll explore the concepts of a "quality centric culture" and how your organization can become one by changing your quality narrative, adopting a "Quality as a team responsibility mindset" and turning testers into Quality Advocates.

You'll also understand the tools, processes and methodologies you can adapt to support your transition into a Native Quality culture that delivers Grade A products, day in and day out.

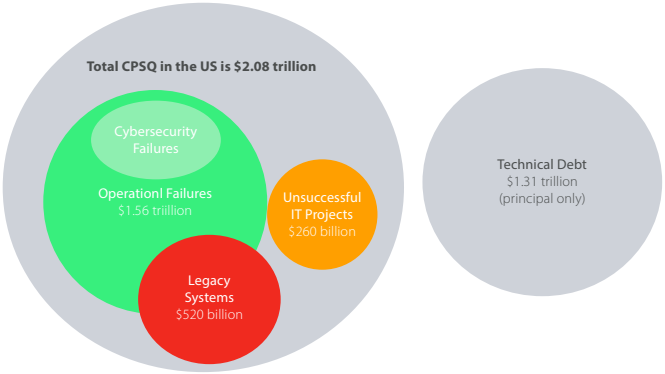
The cost of poor quality

02

A report by Consortium for Information and Software Quality™ (CISQ™) states that the **cost of poor software quality in the U.S. was approximately $2.08 trillion in 2020**. The main factors that attributed to this loss were:

- **Operational software failures:** For 2020, CISQ™ estimated that it is $1.56 trillion, a 22% growth over 2 years
  - The underlying cause is primarily unmitigated flaws in the software
- **Unsuccessful development projects** totaling $260 billion. The project failure rate has been steady at 19% for over a decade.
  - The underlying causes are varied, but primarily **lack of attention to quality**.
- **Legacy system problems** which contributed $520 billion to cost of poor software quality (CPSQ)
  - The underlying cause is mostly **non-value added "waste."**

CPSQ in 2020 in the US



Technical debt is not included in the total CPSQ since it represents a future cost which is increasing (14% rise since 2018).

From the research found, CISQ™ gives the following **best practices to avoid costly quality mistakes.**

- **Find and fix problems and deficiencies as close to the source as possible**, or better yet, prevent them from happening in the first place. This is in line with industry movements such as early work product appraisals and continuous testing.

- **Measure the CPSQ.** With these numbers in hand, you have the basis for a business case to invest smartly in software quality improvement.

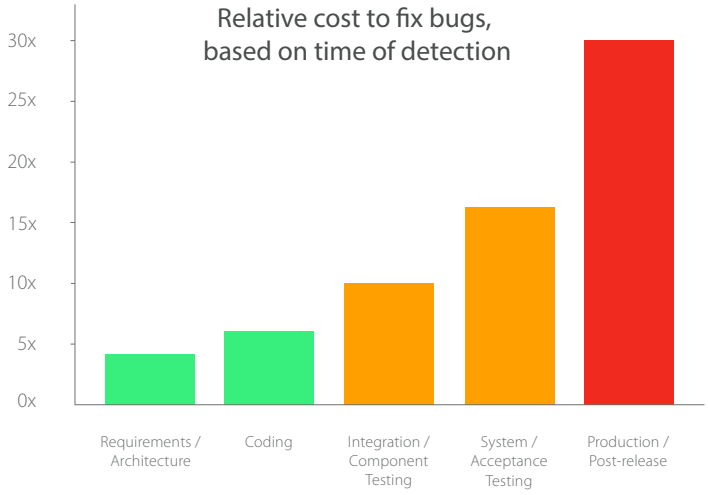
- **Attack the problem by focusing on the different results** of good vs. poor software quality in your shop and relevant benchmark organizations.

- **Economic target areas will likely include**: cost of ownership, profitability, human performance impact, enabling innovation, and effectiveness of mission critical IT systems.

Measure your “Quality maturity” and the time and cost it takes to fix bugs

A method you can use to assess your “Quality maturity” is to measure the time and cost it takes to fix bugs. This will give you a clear indication of how much time you could be wasting on fixing bugs and help you focus on preventing them in the first place.

In the report “The Economic Impacts of Inadequate Infrastructure for Software Testing” by NIST (National Institute of Standards and Technology), the relative cost to fix bugs based on the time of detection significantly increases as you progress through the software development lifecycle. By the time a bug slips through all the way to production and post-release, it can be 30x more expensive to fix than if it was caught during the coding phase.



How to calculate your cost of fixing bugs

In the book, How Google Tests Software – Help me test like Google, authors James Whittaker and Jason Arbon talked about how Google used a process to quantify the cost of poor software quality and how long it took engineers to fix bugs.

They used the following model to calculate the cost of bug fixes:

Source Lines of Code *KSLOC Generated Per Year		200
Average Bugs Per 1000 **SLOC** (Source lines of code)	x	8
Number of Bugs in Code	=	1600
Average Cost to Fix a Bug	x	$1,500
Total Yearly Cost of Bug Fixing	=	$2,400,000
Year Cost of an Engineer	/	$150,000
Number of Engineers Consumed with Bug Fixing	=	16
Engineering Team Size	/	40
Percentage of Staff Used for Bug Fixing		40%

*KSLOC = KSLOC: 1,000 source lines of code

Likewise, you can use this model to plug-in your own numbers and discover the total yearly cost to fix bugs, as well as the percentage of staff used to fix these bugs.

From this research and evidence, we know that software quality can no longer be ignored. Testing needs to evolve to fit the growing demands of rapidly released software.

So, how do you adapt to avoid costly errors?

Evolve into a “Quality-Centric Culture” that prioritizes quality every step of the software development lifecycle.

How to create a “quality culture” in your organization

03

First, we need to understand the meaning of Quality. In the book Leading Quality: How great Leaders Deliver High-Quality Software and Accelerate Growth by Ronald Cummings-John, the author describes "Quality (as) subjective; it's determined by whoever is using the product at the time...Quality is relative; it changes over time."

If Quality is fluid, your team also needs to be collaborative and adaptable to be able to deliver this quality. Shift the focus from being focused on a specific person or team that does testing, to the entire team focused on creating quality.

To begin, understand what is the "Quality Narrative" in your organization.
Leading Quality by Ronald Cummings-John, tells us that "In order to lead quality inside your company, you must become a student of persuasion and influence."

"A company with a highly developed culture of quality spends, on average, $350 million less annually fixing mistakes than a company with a poorly developed one."

Harvard Business Review

He shares 4 practical points you can use to improve your influence:

- Know whom you need to influence and what their motivations, goals, and fears are.

- Create empathy to increase alignment and understanding between teams and individuals.

- Support the narrative with evidence to add weight to your ideas.

- Cultivate internal champions to help create momentum.

Some further questions you can ask include to better understand your quality narrative include:

- Who owns quality in the team?
- What is the perceived role of quality and testing?
- What does quality mean to us? What does quality mean to our customer? Are the answers aligned?
- How important is quality when it comes to releasing a product?
- Are testing and quality exclusive?
- How do we define risk and what is our threshold for risk?

By answering some of these questions, you can better understand how quality is perceived in your organization, and how you can improve it.

Embrace the “Quality as a team responsibility” mentality

04

In modern software development, the tester or QA is no longer solely accountable for quality. **Quality is a team responsibility**.

In a presentation for OmniTestingConf in 2020, Lisa Crispin said that "Quality is a team sport. Quality is something that everyone in the software development lifecycle needs to strive for and contribute towards."

So what happens to testers and the QA roles?

Testers, QA leads and QA teams in general are turning towards a Quality Advocacy or Quality Coaching approach. In the coaching and advocacy role, they're knowledge sharing and building the foundations for a Quality-Centric culture with best practices and tips.

Testers were traditionally seen as the gatekeepers of Quality, however this old-school method has passed. For Quality to be effective, it needs to be a team responsibility. Testers and QA have a responsibility to pass their knowledge and be the quality advocates, because it is not a one team or one person domain.

When the whole team shares the responsibility for the quality of the product, they work faster, smarter and more cost effectively.

Turn testers into Quality Advocates

05

Aristotle once said, “Quality is not an act, it is a habit.” It takes consistent dedication and nurturing to transform into a quality centric culture where quality comes naturally. Testers and QA roles are encouraged to focus their attention on nurturing this quality mindset within the team.

So what does that look like?

SHARING
Knowledge sharing of valuable insights through workshops, blogs, webinars and others

QUALITY MINDSET
Coaching developers to adopt a quality-centric mindset

VISIBILITY
Increasing visibility into everyone’s work with scrum boards

PRACTICAL
Implementing practical exercises, like mob pairing and shoulder-checks

TOGETHER
Pairing testers with developers to bring them into the process of quality

COLLABORATION
Using tools that promote collaboration and visibility like Xray and Jira

As a leader in your organization, you understand the importance that quality plays in your software, but archaic processes, siloes, and old-school mindsets might still linger. You can use tools to support your transformation.

In order for testing to be collaborative, you need one unified toolstack where testers, developers, business analysts and operations all have visibility into the testing progress and contribute towards quality.

Traditionally testing used a separate tool from developers, siloing them from development. This approach no longer cuts it. Testing needs to be an integrated, synchronized part of the software development life cycle.

Enter - Native Quality Management.

From “test management” to “native quality management”

06

In August 2013, Amazon experienced a software glitch that shut down the website for 40 minutes. Amazon lost $4.8M or $120,000 per minute due to this glitch. As a result, Amazon executives prioritized quality management to prevent any similar catastrophes in the future.

With the rise of Agile, DevOps, and Continuous Testing, we understand that QA and development teams need to collaborate and work together. This new, holistic approach naturally embeds the quality management process into the development workflow.

With Native Quality Management, all the tools, tests, and processes used by QA are built natively into your development environment like Jira. That way, every test is accounted for, every task lives in the same workflow, and everyone speaks the same language.

Xray links all requirements to testing, so you never miss a test. It gives all teams the same naming and terminology, so they work seamlessly side by side. And it combines all activity into a single, easy to manage workflow, so teams run with effortless efficiency, speed and control.

Thousands of organizations worldwide trust Xray to infuse quality seamlessly into development, and release consistently quality software - together.

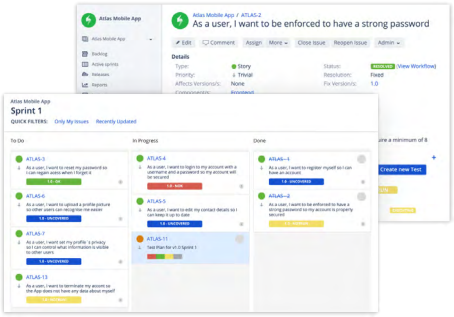
Here's why.

Native Quality Management in action

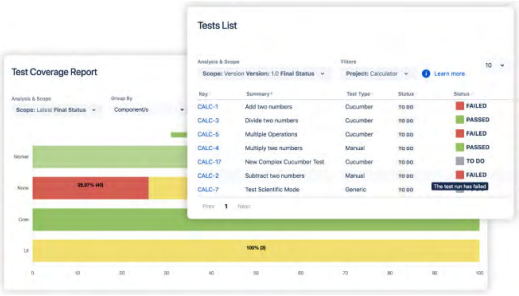
07

With Xray woven into every stage of development, quality now comes naturally.

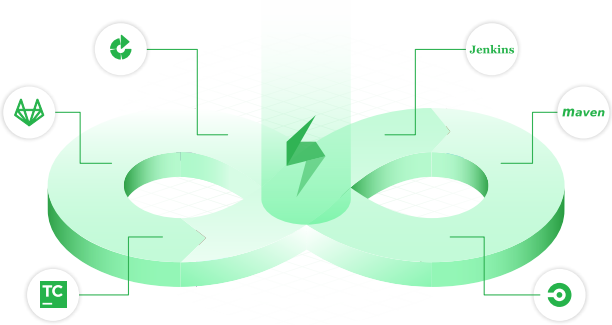
- No bug left behind. Cover everything.
  With all development requirements naturally linked to testing, you'll never overlook another test. Now, quality is built right into everything you develop.



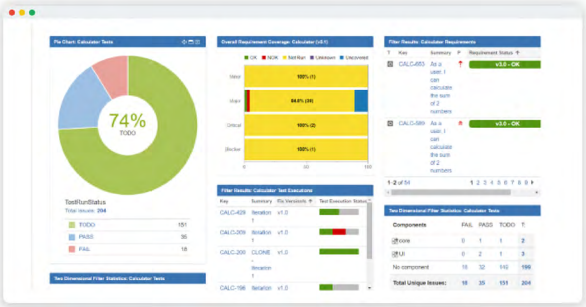
- Learn where tests went right (and wrong)
  With detailed traceability and test coverage reports, you know which tests went wrong and where -- so you pinpoint what to fix, and easily collaborate with developers to fix it.



- **Work in the same flow**
  Streamline development and testing into a single collaborative workflow. With teams perfectly aligned, they run tests with sprinter speed and marathon consistency.

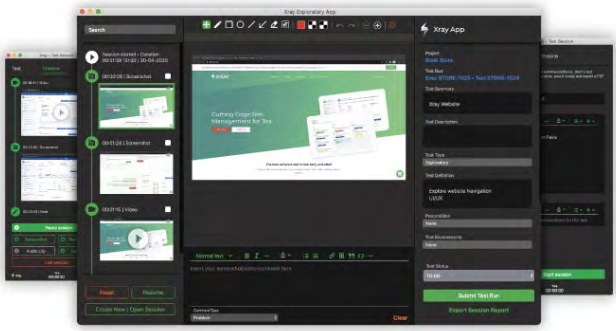


- **Keep tabs on all your tests**
  Xray indexes tests in real time, so you run tests with full control of the entire process. That way, you get total coverage, catch problems fast, and keep releasing quality software with confidence.



- Hook automation into your CI/CD
  Xray integrates with leading automation frameworks like Cucumber, Selenium and JUnit to automate testing, so you can deploy at supersonic speeds. Easy integrations with CI tools like Jenkins, Bamboo and GitLab hook your automation right in your CI/CD pipeline.

- Exploratory Testing
  Use the Xray Exploratory App to uncover hidden bugs by testing your web, mobile and desktop applications in brand new ways -- all from your desktop and with seamless integration into Xray and Jira.



Conclusion: The next wave - the software quality lifecycle

Imagine all development requirements naturally linked to your testing, so you never overlook a test again.

Imagine the same naming and terminology across all dev and QA tasks so both teams work seamlessly side by side.

Imagine all activity managed in one single environment sharing one familiar workflow, giving your teams more control, efficiency, and speed than ever before.

But don't take our word for it. The hundreds of raving reviews from leading organizations worldwide speak for themselves. And they all say the same thing - Native Quality Management is the best way to make sure you deliver the best software. Period.

SEE XRAY IN ACTION

Links

www.getxray.app

/showcase/xrayapp/

/xrayapp/

/xrayapp/

getxray.app