QualityLogic

WHITE PAPER

What Should Quality Cost?

"Testing programs have to consider coverage, depth and repetition. And a quality program is much more than test plans and cases."

Table of Contents

Overview

How much of a company's resources should be devoted to software quality? Too little focus on quality and the product or service suffers. Too much can drag on release windows.

First, let's agree on some definitions. Quality, Quality Assurance, Quality Control... all get used as synonyms for each other, and yet each has a different meaning. To avoid label confusion, this paper will use Quality Assurance to cover both quality processes and code testing. Now, on to the central issue – how to balance quality results versus costs.

A comparison of results versus costs sounds simple, but there are a great many operational aspects of a quality program's effect on product delivery and market reputation that are not so obvious. Testing programs have to consider coverage, depth and repetition. And a quality program is much more than test plans and cases. Development, as well as release and code management processes, enters into the quality discussion, as do defect tracking and quality resource planning. Then there is the need to sort out costs of quality work performed from expenses caused by quality work that is not performed due to various cost and time constraints.

Cost vs. Benefits — the Eternal ROI Question

It is important to understand how Return on Investment (ROI) differs for development versus QA efforts. ROI calculates money spent against the amount made back plus an incremental percentage. This is how expenditures on tools and services related to revenue-generating production are evaluated. Let's say that X dollars were spent on a software development framework, and that framework improved the development process such that an additional Y% of that money came back to the company in terms of increased productivity. That additional money returned to the company coffers, expressed as a percentage of the original amount, was the ROI on the $X investment.

"QA ROI is more difficult to calculate primarily because quality efforts are seen as expenses rather than capital or productive labor investments."

QA ROI is more difficult to calculate primarily because quality efforts are seen as expenses rather than capital or productive labor investments. Where production ROI works on the concept that an investment was required that was going to return a profit, QA is seen as a cost of doing business. There is no line on the balance sheet that defines the value of the benefit bestowed on product sales because of QA efforts. QA results are even more difficult to parse out because many of the most impactful returns come out of development practices that were improved due to inputs from the 'Assurance' aspect of the QA process. There is also the matter of how to quantify the returns from a bug that was caught during testing and didn't get released in the product.

This all begs the question: 'How are QA benefits measured?' The direct answer is to develop a tracking system specifically designed to measure them. While that sounds fairly simple, it requires the cooperation of both development and marketing. The payoffs are quality improvements that can be measured against the efforts and resources required to achieve them.

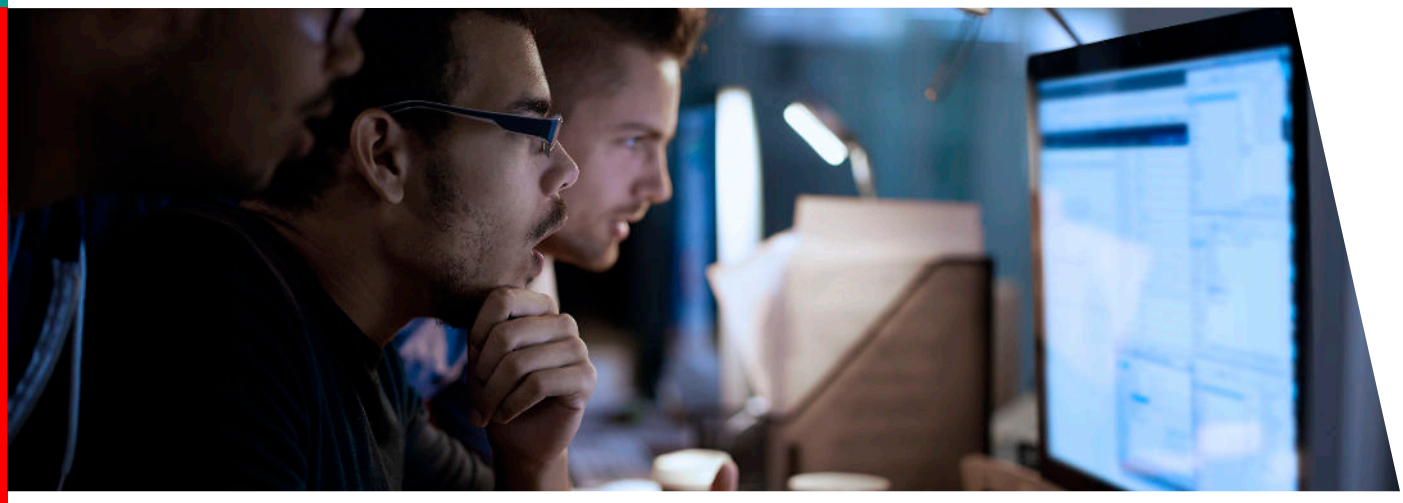
Measuring QA Effectiveness

Software quality revolves around two basic concepts:

- Finding and precisely defining defects in the code
- Verifying that the design features are all in the product and working as product management intended

On one hand there is the search for basically broken code, and on the other making sure that all the code is there and doing what it's supposed to do.

This puts QA management in the curious position of attempting to assess the effectiveness of their department's efforts by measuring something that, if all goes well, doesn't exist. While defects found and fixed can be counted, it is difficult to measure the actual revenue that was preserved by having accomplished QA goals. Indeed, it is intensely difficult to assign an objective value to the preservation of both the product's and the company's market reputations.



Marketing professionals may attempt to measure customer satisfaction with surveys and analysis of tech support call records. Unfortunately, these efforts gather far more negative input that positive responses — complaints are voiced more often and more stridently than praise. A truly effective QA program means that the customer is pleased with the product or service because it 'worked out of the box,' and there was no impulse for further contact between customer and company.

Still difficult to quantify, but more directly measurable, is the time to market for new features and defect fixes. There is intense pressure to take advantage of every new feature insight and get each defect fixed as quickly as possible, and the Internet is thoroughly unforgiving of both real and perceived shortcomings. QA is often all that stands between releasing a feature or fix that delights customers, or conversely aggravates them! An effective, agile QA program is essential to getting product changes verified and released as quickly as possible.

A little-appreciated aspect of such a QA program is its impact on the development process itself. Code has become modular to the point of being accessed through internal system APIs (Application Programming Interfaces). This structure makes a code module accessible to the entire system and allows the creation of intensely complex middleware to handle decisions and data flows between system front and back ends. This also means that new features or fixes can have immediate repercussions throughout the product, causing operational failures that appear to have no connection to the changed code.

"...in order to get the resources to bring them to fruition, their contributions to the bottom line have to be measured."

A QA process that aggressively pushes new code through all its possible parameter values and verifies full system operation at each release point will catch this kind of collateral damage. This heightens awareness in the development team and makes them more conscientious about analyzing each code change for unforeseen consequences. Ultimately, development efficiency increases as these issues are eradicated and coding time is no longer eaten up by fixes.

All of these aspects of QA have one thing in common: in order to get the resources to bring them to fruition, their contributions to the bottom line have to be measured. QA funding is inextricably bound to results-oriented data collection. This means tracking each and every QA action, its results, and especially its contribution to creating competitive products.

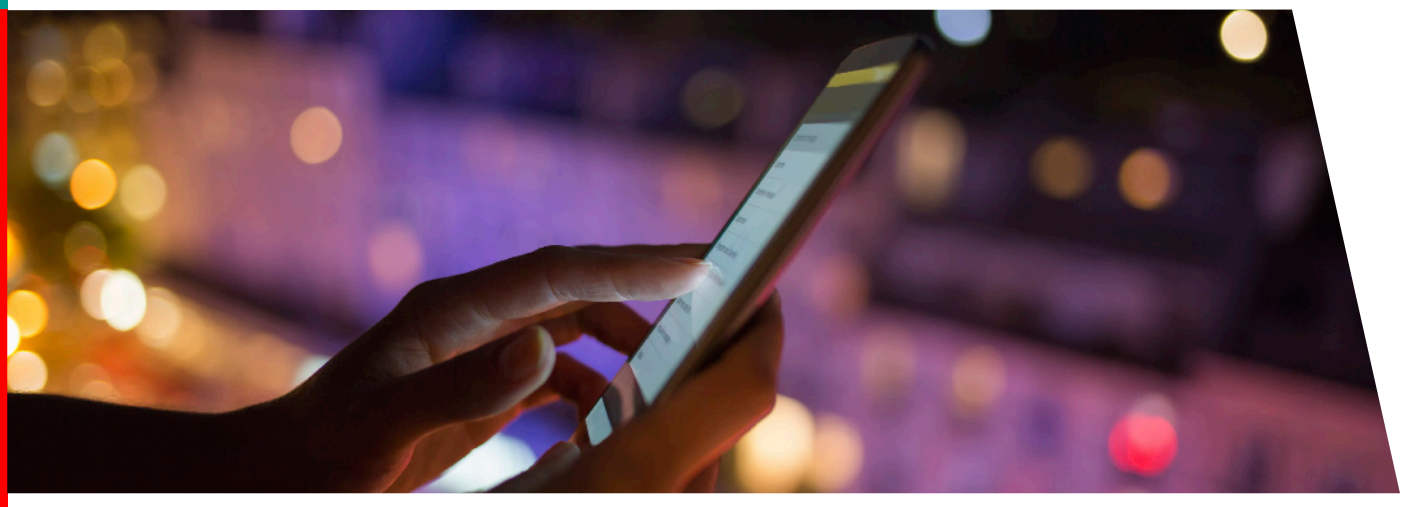
Which QA Processes Are Really Cost Effective?

Anyone involved in QA has opinions, preferences and prejudices concerning what, how and how much should be done in pursuit of QA goals. What follows is a look at some of the more contentious concepts, and a review of each.

We're Over the Waterfall

The over-arching consideration about marketing, development and QA is their mutual relationship. This relationship is commonly called the Software Development Life Cycle (SDLC). For decades, the waterfall model held sway over the interactions between marketing, development and quality. This version of the SDLC held that marketing designed the product, turning a feature list over to the development team who massaged that list into detailed specifications, developed code and then 'threw it over the wall' to their quality colleagues.

In the waterfall development model, marketing, development and quality were isolated from each other, with information passing between them through very formal channels. Separation between development and QA in particular was considered a high priority goal. Development of a new or significantly changed product was expected to take 12 to 18 months, much of which was typically consumed by contention between development and QA over compliance with marketing's product requirements.



Enter the Agile SDLC

The Internet, and eCommerce in particular, reshaped the playing field. With market conditions and the competitive landscape changing at a moment's notice, there just wasn't time to wade through the waterfall slog anymore. Enter the Agile SDLC model.

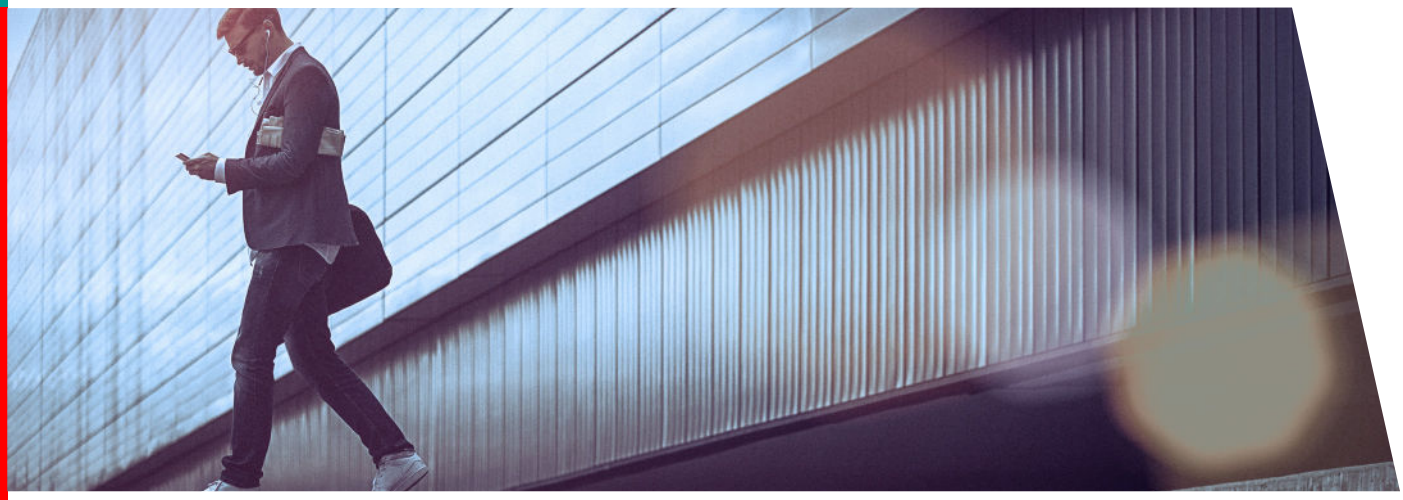
Through Agile, product changes are made on an incremental basis — the more granular the complexity and impact of the changes the better. It postulates many small changes in place of a few large ones. To accomplish this, release cycles are reduced from months to one or two weeks, and the marketing, production, quality triumvirate is effectively merged into an array of action teams. Instead of isolation, each group contributes personnel to work directly in small teams with the others. In its purest form, Agile is the methodology for CI or Continuous Integration — a steady flow of defect fixes and new features into the released product.

Quality in an Agile Environment

Quality's role in an Agile team is to work with:

- Marketing in defining new features to support rigorous testing
- Development to introduce the quality perspective to the coding process, making modules more uniform and easily testable

Quality often maintains the tracking system on which development sprint goals are carefully documented, their achievements recorded and their verification confirmed. Agile methodology has enjoyed wide acceptance in a relatively short period of time. But it seems that the waterfall model dies hard, and most Agile implementations are more like mini-waterfall processes. The team meets, plans the upcoming sprint and the developers start writing code. Part way through the sprint, the quality people get code releases to test, and finally that code is integrated into a production release that is verified and put into the live product. This hybrid system misses a major opportunity for cost savings.



Agile methodology supports a cherished tenant of good QA, which is that a defect caught earlier in the SDLC is less expensive to fix. When a rapid, direct mix of QA and development is implemented, it prompts detection of code and architectural errors early in the development process and actually prompts the design of code to facilitate testing.

In addition to testing the feature/fix code and the release candidate, quality team members help by allocating time and personnel resources for the quality effort during the planning meeting. This avoids overestimating what the sprint can accomplish by aligning the schedule for quality processes, as well as code development. Successful agile development absolutely requires the resources to make quality an integral part of each sprint.

"Agile methodology supports a cherished tenant of good QA, which is that a defect caught earlier in the SDLC is less expensive to fix."

Test Automation

The role of automation is an often-debated area of the QA process. However, asking if automated test cases are a cure or a curse is the wrong question. Test automation isn't a binary choice of do it or don't.

Code that is stable and is changed only occasionally begs for automated testing. Systems that have large, seldom-changed frameworks need test automation to verify that changes didn't break something on which they were not supposed to have any effect. Those test scripts are safety/sanity checks that perform system-wide functional verification.

Then there are product types and code blocks that are not well suited for test automation. Their code changes often, intermittently or both, and the changes tend to be widespread rather than confined to a specific aspect or functional area. Since test scripts have to change along with the code they test, the scripts would cost as much to develop and maintain as the code.

Most systems involve both of these situations — some areas need automated testing, while others would make automation a nightmare. Writing and maintaining test automation scripts is a programming job, and has all the costs and constraints found in product development. The QA staff who perform this work are called SDETs (Software Development Engineers in Test). They are much harder to recruit, and typically more expensive than development engineers.

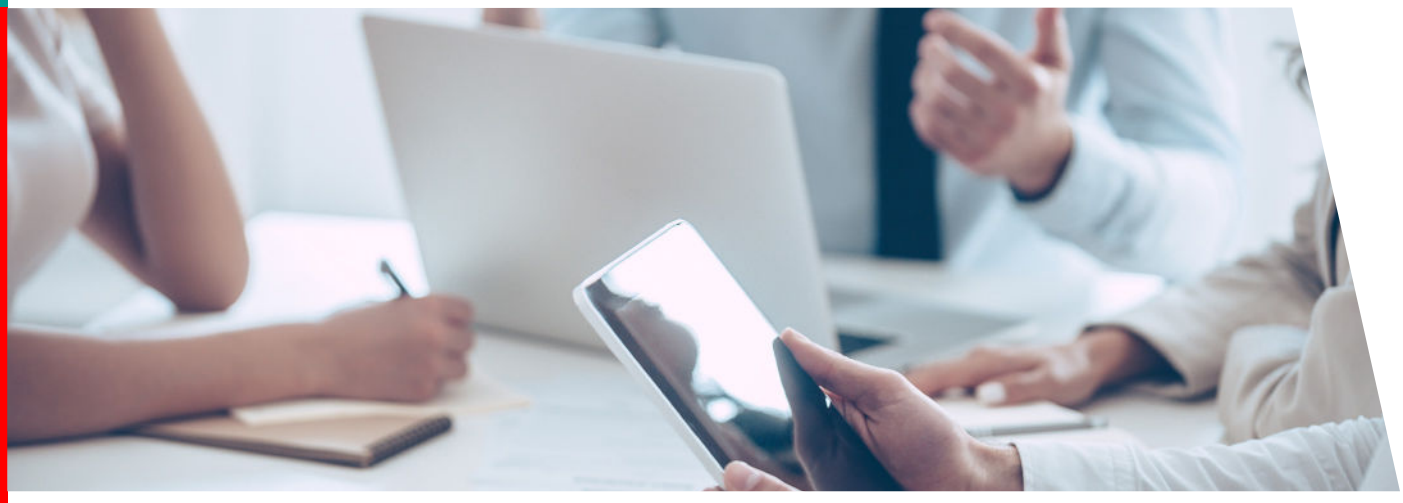
All these considerations mean that implementing test automation is not a one size fits all. It requires an evaluation of each instance of proposed test automation to see if it makes sense or not. Automation has the advantage of speeding regression testing and allowing wide-coverage sanity checks at the push of a button. Take advantage of what automation does best by carefully assessing where it makes the most sense to use it, and writing/architecting code to take advantage of its strengths.

"All these considerations mean that implementing test automation is not a one size fits all. It requires an evaluation of each instance of proposed test automation to see if it makes sense or not."

Communications

A principal and oft-overlooked purpose of software QA is to facilitate communication. From its inception, QA has closed the feedback loop between development and marketing to 'assure' that the product does everything marketing requested, and doesn't have any hidden defects. The Agile methodology strongly supports this by putting all three of the main players in product development in communication with each other. Quality ensures that everyone understands the implications of how management decisions affect releases, and the process of verifying functionality and usability.

While they may fall to production, defect tracking processes and sprint planning are often done by QA. Whether managing them or not, quality should make maximum use of these initiatives to keep release verification on track and to create an audit trail of what happened with each cycle. When considering a test automation project or an upgrade of QA staffing or tools, these systems contain data that make the case for their additional value.



Budgetary Constraints

An effective QA group is an expensive investment, and creating effective test automation adds to the up-front costs. When the time comes to allocate resources and draw up departmental budgets, there is a strong temptation to put QA at the end of the process. It is often listed as an expense on the balance sheet and not a revenue generator. This puts QA management in the awkward position of having to argue for something that is most strongly perceived in its absence. Operational data becomes invaluable at this point.

The cost of deferred or canceled QA budget allocation was illustrated in a survey carried out by ClusterHQ. The survey found that a quarter of the respondents report encountering bugs discovered in production one or more times per week. The most common causes of these bugs were: inability to fully re-create production environments in testing (33 percent); interdependence on external systems that makes integration testing difficult (27 percent); and testing against unrealistic data before moving into production (26 percent). When asked to identify the environment in which bugs are most costly to fix, 62 percent selected production as the most expensive stage of app development to fix errors.

Microsoft contributed the term 'escape rate' to the QA lexicon. It refers to the defects that slip past the QA process and are ultimately released as part of the product. They range from aggravating features that don't work right, to typos in the user interface, to hidden functional breaks that erase both revenue streams and business reputations. Escape rates are typically expressed per release version and per thousand lines of code. Either way they represent expensive losses that are difficult in the extreme to recover.

Escape rates are, however a primary measure of QA effectiveness. Tracking their decrease makes a potent argument when it comes time to discuss budget allocations. This is also the time to emphasize risk management through carefully planned test process coverage.

Look to system verification coverage to see what types of QA efforts need to be applied where. It is key to laying out an automation project plan and allocating human QA resources. This is where the balancing act of resource allocation versus risk assessment takes place. While there are a number of open source and proprietary tools to assist with coverage assessment, the paramount objective is to create a comprehensive overview of the code body and allocate test cases and efforts so that the risk of escaped defects is minimized.

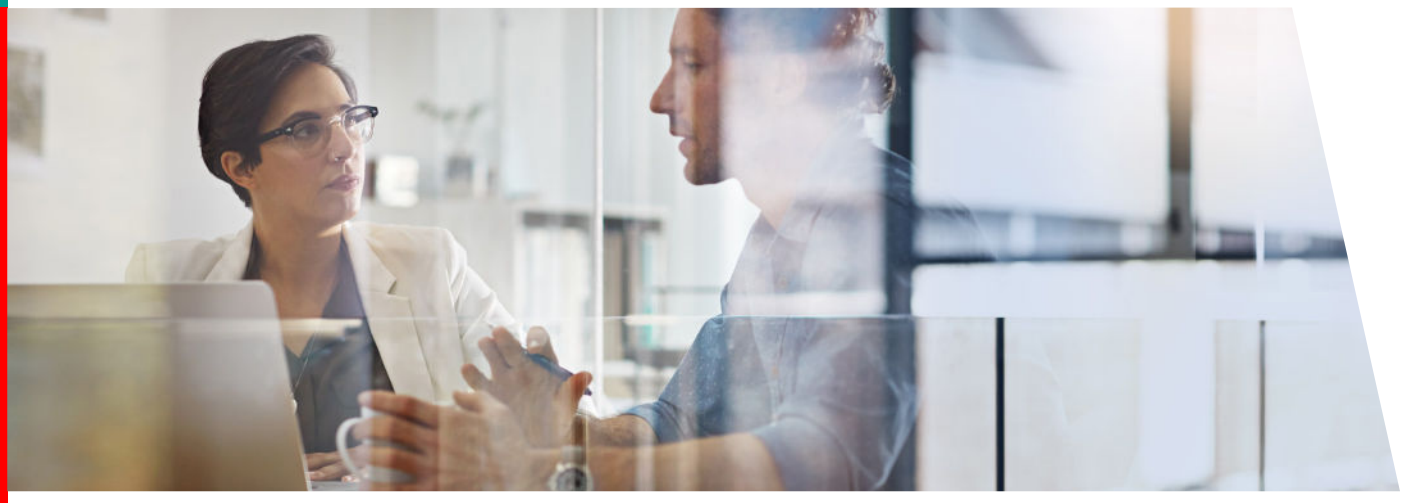
Defect detection/correction history maintenance, escape rate measurement, and coverage verification are on-going efforts to sharpen and refine QA processes. Their records are the milestones that guide QA improvement and supply ammunition for the budget battles. Comprehensive audit trails should be designed into all QA efforts.

Budgets Are One Thing, Resources Are Another

Creating the results that score a winning budget requires a QA organization that can do whatever the quality situation calls for quickly and effectively. This means a team that understands the technical aspects of the system, as well as the people who created it and who have the quality perspective. That perspective means constantly looking for what could go wrong and, if something does, what else it can damage. This mix of knowledge, skill and perspective can make a good QA engineer (or SDET) difficult to find.

Staffing is made more difficult by the clustering of technology businesses. Everyone is trying to hire both developers and QA engineers out of a finite personnel pool.

One obvious solution is to outsource QA to a third party service. This means that staffing and the QA function are performed by someone whose primary business is Quality Assurance. This last point bears repeating, a QA service has already done the heavy lifting necessary to structure and staff winning QA organizations. As third party services, they are scrutinized closely and held to their cost estimates and accomplishment schedules. Failure to perform isn't just an uncomplimentary paragraph in the annual report, it's a lost client and a damaged business reputation.



Offshore

Offshore organizations typically have sales offices near the tech hubs, but their test personnel and labs are located in another country. This allows them to offer their services at competitive prices due to the low wages prevalent in these areas. Many of the countries that host these labs have made significant investments in technology education.

A look at a time zone map will quickly point up a major drawback of using one of these offshore QA labs. There is a half day differential between their lab and a U.S. development facility. Communication is typically via email rather than conversations. While this can help with language differences, it makes pro-active, responsive problem solving, a hallmark of Agile development, nearly impossible. Cultural differences can cause interpretations of directives and specifications to be challenging at best. Offshoring also removes a level of intellectual property protection by subjecting disputes to IP laws outside of the U.S.

Nearshore

Nearshore facilities are typically located in Latin America, which puts them within two hours of the U.S. east coast. While this relieves much of the time zone displacement that occurs with offshore locations, it does little for the other communication issues mentioned above. They also lack some of the extreme cost of living differences that make offshore labor much less expensive.

Crowd-Sourced Testing

In the age of Agile development methodology, Beta testing has evolved into crowd-sourced testing. Release candidates are sent out to large groups of testers affiliated with a test services company. The testers then exercise the putative release in a wide variety of user environments reporting their results to the services company, which renders a combined report back to their customer.

Marketing confidentiality is a singular concern with this type of testing as it gives advanced notice of a product or feature release well before its market window is prepared. Security is also an issue as a pre-release copy of the product is placed where a determined hacker can acquire a copy to examine for exploits.

The main issue, however, is timing. Crowd-sourced testing can only be done on an integrated release. It does nothing for module testing, support services testing, or end-to-end tests that verify updates to databases. These functions must be performed well before a release candidate is issued. Where Beta tests drew input from a relatively small group of test personnel who were usually familiar with the product and its intentions, crowd testers will approach the product as a finished system and report issues only from that perspective. Their input is useful but incomplete.

Onshore

Onshore companies have labs in the continental United States. These are often located close to the tech companies who are their customers. However, some employ 'rural sourcing'. Their labs are located close to universities and qualified personnel but away from the expensive technology hubs to make their services much more affordable.

Onshore QA services, especially those with labs in the Mountain or Central time zones, are on a working schedule compatible with any U.S. based company. They will operate with the same cultural background and language that their action directives and system specifications were generated from. As an added bonus, they are physically close enough for affordable on-site visits.

Time zone compatibility also allows the QA engineers from an onshore service to become integrated into an Agile development process. They can directly participate in Agile sprint planning meetings and stand ups. This last item is important because it allows a third party QA service to fill out the QA staffing requirements that have grown as Agile has become ubiquitous.

Two Pertinent Use Cases

QualityLogic's business model focus is to provide onshore QA services. The use cases below are from actual client engagements and show how an onshore QA service can work closely with a development team to provide the QA components that Agile methodology demands.

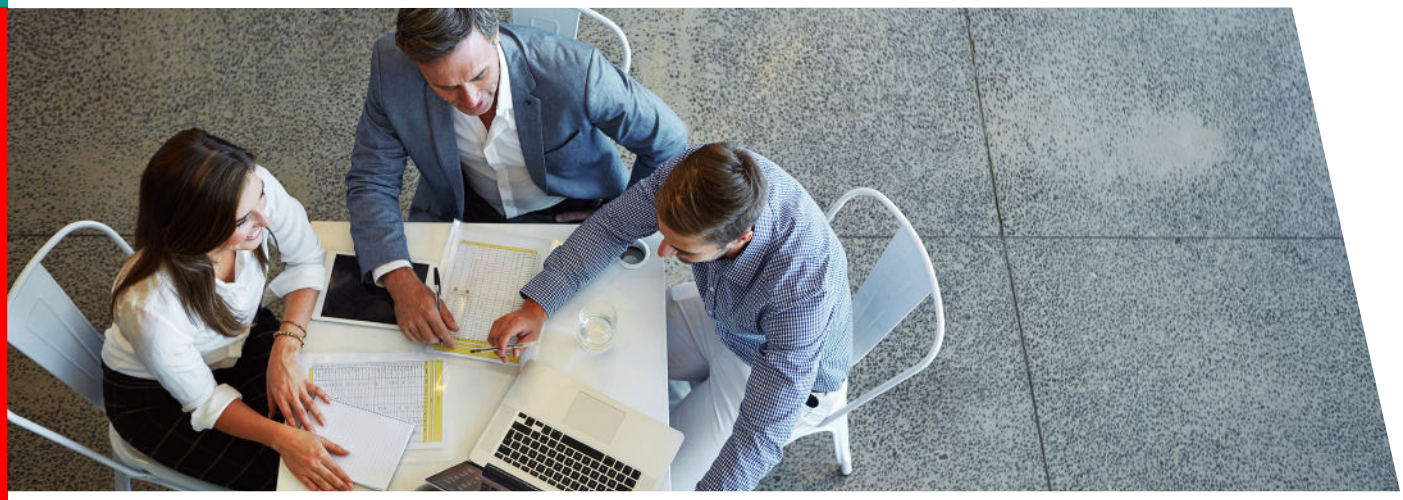
Case One

An emerging eCommerce company was implementing Agile methodology and expanding its array of web offerings at the same time. Its QA director had been unsuccessful in hiring engineering level QA personnel to fill the expanding list of vacancies created by these dual efforts. After consultation with his CTO, he began surveying third party QA services to find engineering talent. His expectation was that this would be a temporary arrangement, lasting no more than four months, while his HR department hired the necessary full time employees.

QualityLogic placed three QA engineers into their Agile teams immediately, followed by three more within weeks. Two of the engineers traveled to the client company to learn their methodology implementation and infrastructure. They brought the additional engineers up to speed as quickly as they joined the project. The QualityLogic engineers were in constant communication and sat in on sprint planning meetings and standups via conference and sprint calls.

The QA workload grew as the customer's expansion continued, and the Agile teams were eventually being assisted by nine QualityLogic engineers. As system features matured from new offerings requiring support of large new code releases and transitioned into maintenance activity, the QualityLogic engineer complement was cycled back down to four engineers. They continued to seamlessly support the QA requirements of the sprints of which they had become integral members.

This symbiotic relationship has worked to the degree that, five and a half years later, QualityLogic is still augmenting this company's QA staff.



Case Two

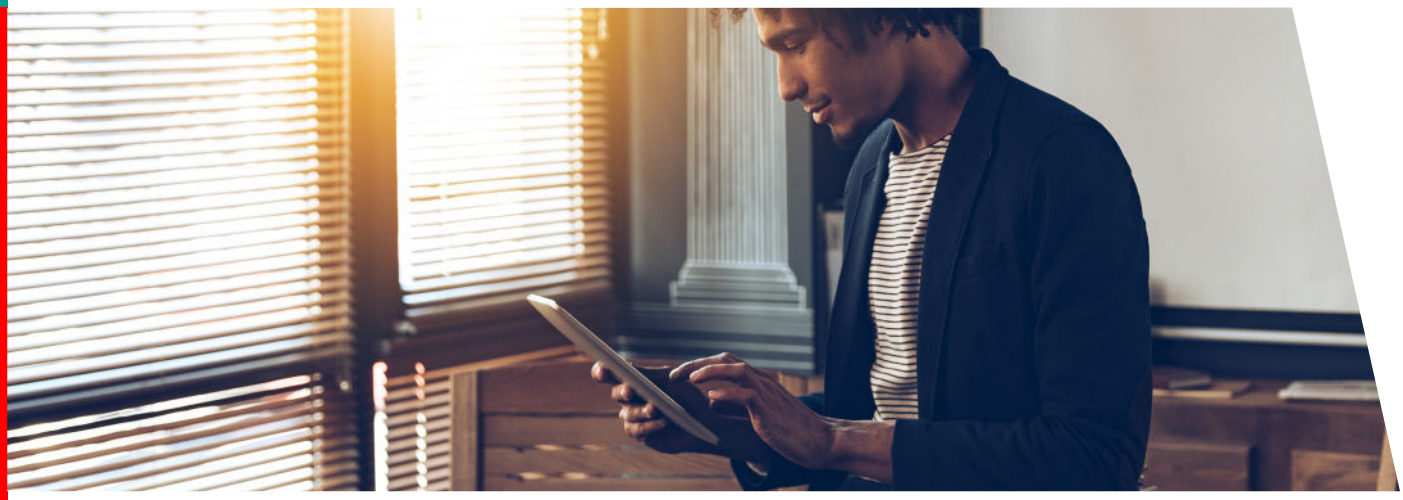
Following a merger, a top three media company required a significant expansion of their QA staffing and its flexibility in handling unanticipated work load fluctuations. Their systems were changing rapidly in both features and content, and they needed a QA service that could ramp up and down quickly with these variations.

QualityLogic performs manual functional testing on their web properties, mobile apps, OTT apps, and game console apps, validating that their players, sites and ads are working correctly across all their web sites and apps. The work requires creation of extensive test cases based on system functional specifications. Defect fixes are assigned and verification reported through a direct interface with their JIRA tracking implementation. It is also the communication channel used to perform validation testing on production releases.

At the other end of the skill spectrum, QualityLogic QA engineers test the analytics generated by their systems to validate beacon firing and content. They further examine server recordings to see that the generated data is both processed and stored correctly. A new and interesting aspect of this project is to scan app store feedback and review comments for their apps. Any issues are reproduced and a defect written up.

QualityLogic's test leads and project managers work directly with the media company's QA management on a day-to-day basis. They participate in Agile planning and standups and perform conferenced updates on test plans and results. Using a third party QA service has allowed this company to cover its testing requirements quickly and absorb unexpected workloads to deliver sites and apps that were ready to go when their hard-wired market windows opened.

"A new and interesting aspect of this project is to scan app store feedback and review comments for their apps. Any issues are reproduced and a defect written up, if possible."



Summary

Successful companies win with effective QA. The cost of releasing a product or service that hasn't been thoroughly exercised and verified is far too high to skimp on the thought processes, systems and teams required. Agile development and test automation were created to serve explosive growth in the online market place. All of this has made the human resources that staff a QA department more critical than ever.

While the costs of a first-class QA capability are daunting, it is not only worth the cost, its costs can be controlled. When the need arises onshore, QA outsourcing can provide cost efficient, experienced, highly-skilled talent that is organized and ready to go.

For More Information

Visit www.QualityLogic.com or call +1 208-424-1905