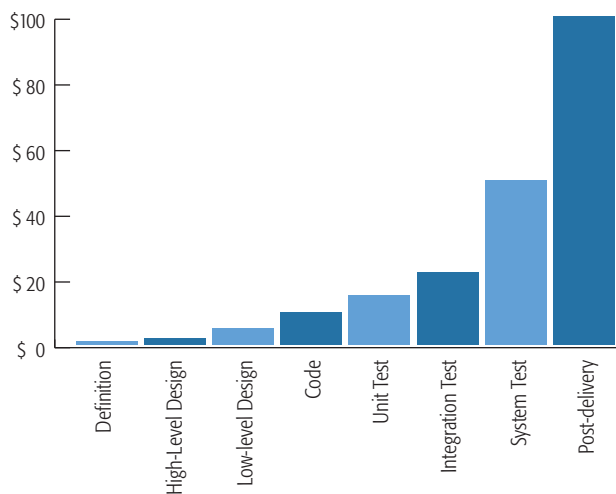


Integrating Quality into Every Stage of Development

by Keith Vanden Eynden

Where does quality assurance start? Is it a stage in the development process? Is it the responsibility of a single department? Unfortunately, in many development organizations, programmers write code until it is as finished as they can get it and then throw it over the wall to the testers, who then have whatever time is left in the release schedule to test as much of the application as they can get to. In these organizations, quality is little more than an afterthought.

While this type of arrangement is common, it is by no means ideal. Bugs are expensive to fix, and the later you find them the more costly they are to fix. If your organization discovers a usability issue during design, it might take a few sentences in a document to avoid the problem. However, if QA finds the issue two weeks before release, you may have thousands of lines of code to sift through, resulting in hours of extra work and rendering some of the time spent in the other stages of development useless. The cost to fix defects increases steeply as the development process continues.



Software Reliability: Achievement and Assessment, B. Littlewood, ed. (Henley-on-Thames, England: Alfred Waller, Ltd., November, 1987)

A much better solution is to consider quality at every stage of development. Whether you are a developer, QA analyst, or development manager, this article will provide you with some simple ways to implement quality assurance throughout your development process.

Requirements

As soon as a requirement is written, you can test it. Begin by asking whether a requirement is valid and should even be included in the software. Many QA analysts spend hours trying to test features that do not belong in an application in the first place.

To introduce quality into the requirements phase, develop a quality measurement for each requirement. Consider using the following criteria.

Necessity

By keeping the goal of the software in mind you can determine if a requirement is necessary. This keeps your organization from doing things like adding a table of contents generator to a spreadsheet application or an image editor to an accounting program.

Correctness

Correctness evaluates the requirement against what the user wants. You can employ use cases or consider regulations and business rules as part of this criteria.

Completeness

Determining if requirements cover everything can be a daunting task at the early stages of development, but here are three ways to help you check if the requirements are complete:

- List problems that the software or enhancement must solve
- Use modeling to uncover problems already solved by the current version of the software
- Brainstorm with developers, product managers, users, and other team members on possible hidden requirements

Coherency/Consistency

The requirements should be clear to everyone who reads them. This means including definitions of terms used throughout the document. It is important to be consistent when referring to components of the software. For example, do not refer to the window that stores user preferences as the Options dialog box in some places and as the Settings dialog box in others.

Testability

Make sure the requirement can actually be tested and verified. If it is testable, you can start creating tests at the requirements stage and use those tests throughout the development process.

Feasibility

Consider whether the requirement can be completed in the allotted time, within the budget, and with the available resources. Too many organizations undertake unrealistic features that overtax their resources and sabotage quality. The best way to avoid this is to schedule ample design time up front and to avoid rushing into coding.

Traceability

Identify the components affected by a requirement throughout the application so that the impact is clear and both developers and QA analysts can identify which areas must be tested or updated if there are changes. While this can be hard to do and time consuming, it is well worth the investment.

Strive for quantifiable criteria when evaluating for requirements. For example, “the query should return the results quickly” is not quantifiable; whereas, “the query will return results in less than ten seconds over an ISDN line” is.

Also, templates can be valuable for ensuring that all elements of the requirements are captured, no matter who is writing them. Your organization will not have to worry that one developer will provide the details of the database structure while another will ignore it in favor of an exhaustive list of new functions. Some organizations have technical writers create or edit requirements to ensure consistency and clarity.

Design

When creating a design document try this trick: Write a document someone will actually read. What good does it do if you write a design document people abandon after three pages? To make the document engaging include use cases, diagrams, charts, examples,

and anecdotes—anything that breaks up the flow of technical jargon. The more people that read the document, the more issues your organization will find early.

In the design phase, you want input from a variety of departments. Sales and marketing can provide input about the users’ needs. Technical writers can help with the clarity of the document. QA can uncover elements that might be un-testable.

Regardless of who reads the design document, consider the following:

- **Standards** – Review the design against any internal, industry, government, or regulatory standards.
- **Specificity** – Make sure the design document does not contain vague terms. Watch for words, such as always, never, sometimes, and etc. Also, beware of passive voice, where it is unclear what is performing the action. For example, “results are calculated” indicates nothing about which component performs the function or how the software presents the results to the user.
- **Usability** – Read the design document as if you are the end user and make sure the software’s components are clear and that nothing essential is missing.

Coding

One of the most effective QA activities at the coding stage is unit testing. Unit testing exercises a portion of the code and determines if it functions properly. While many developers perform ad hoc unit testing, the true value comes from structured, repeatable tests. To achieve this, create unit testing scripts in tandem with the development process. While this adds another development activity, it enables your organization to find defects early and fix them more cheaply. That savings can often justify the extra time required to develop unit tests.

You can also implement test driven development to further integrate unit testing into your process. Begin by writing the tests that prove a feature is working. Then write the minimum amount of code for the portion of functionality to pass a single test. When the component passes, you can begin writing code to satisfy the next test. After you add an element of functionality, return to the previous tests. Ensure that the code still passes those tests while considering ways to simplify it. Through this cyclic process, testing becomes an integral part of writing code.

Another way to use unit testing is to incorporate it into the build operation. You can configure your source code management and automated build tools to run a series of unit test when files are

checked in. The files are only committed if they pass the tests. This ensures that only verified code is added to the codeline.

Yet another way to implement unit testing is by using pair programming. Two developers work together; one writes the code while another reviews it. This can occur in real time with the second programmer looking over the shoulder of the first, or they can review each other's code at specific milestones.

Any of these unit testing procedures increases the number of bugs you find before the code ever goes to a tester.

Testing

The testing phase is traditionally where the QA department receives the code. In many organizations, this is the first time the software is tested. Implementing testing early does not eliminate this phase. Instead the QA team can test more complex features like system integration instead of focusing on interface consistency and other low-level issues.

Support

After the software is released, quality assurance becomes even more important. At this stage, every defect or issue is more urgent and costly because users are affected and more people (support engineers, developers, QA analysts, etc.) are involved in the fix. Defects for released software should be top priority, and developers and testers need to be available to resolve these issues quickly and adequately. You have to test these fixes to the highest level to ensure a positive user experience.

Conclusion

Testing quality into an application takes longer than if you considered it from the start. If you want to deliver on time, you must build quality into both your product and your process. Quality assurance is not just a department or a stage in the development process. It must be woven through the fabric of your enterprise. This is the only way to achieve quality in a cost effective manner.

About the Author

Keith Vanden Eynden is a senior technical writer at Seapine Software, Inc. with 15 years experience in performance improvement, training, and documentation.

