MICRO FOCUS

ADM Market Insight: How Developers Ensure Software Quality

How Developers Ensure Software Quality

The software development process has evolved considerably over the past couple of decades. Traditionally, it involved three departments: development, quality assurance (QA), and build and configuration.

Under this model, developers wrote code and compiled a version of the software for the QA team who would typically use automated and manual tests, and alert the development team of any notable defects. After making the necessary fixes, the development team would then compile a new version and hand it back to QA for further testing.

This cycle would continue until QA was satisfied with a particular version. At that point, QA would sign off on that version's release and pass it to the builds and configuration team, which would package the software into a shippable form and burn a "gold" CD for manufacturing.

These interdepartmental "handovers" were cumbersome and inefficient. Developers were not only excluded from quality checks, but also risked adopting the mindset where they had no responsibility for quality whatsoever.

Software companies recognized this, and shifted towards the Agile software development model, which breaks the work into separate, goal-specific tasks called sprints. These generally only last a few weeks as shorter cycles improve focus and simplify planning. The Agile model combines development and QA departments, giving developers considerably more quality ownership.

Additionally, cloud-hosted development environments that use services such as Amazon Web Services (AWS) and Microsoft Azure enabled a new set of practices known as continuous integration and continuous delivery (CI/CD). With CI/CD, developers can now drop new code into a repository several times a day, as well as update the production environment as they see fit. Ultimately, these changes allow them to have a more active role in software quality.

Let's look deeper into how developers are embracing their new role as quality assurance partners and what tools they use to catch quality issues before the software reaches the end user.

The Developer's New Role in Software Quality

As modern software development processes become more inclusive, developers have taken more ownership of quality in their work. However, developers should still move away from the "quality equals testing" mindset, since testing only finds quality issues after they occur.

To ensure code quality, checks should also happen on developer machines. Rather than relying on the CI phase for quality checks, developers should monitor for quality as they develop the code so they can fix any issues they come across on the spot.

Today, developers can play an important role in maintaining high code quality by following best practices for code review, properly managing artifacts, and creating strong retention policies.

Code Quality and Code Review

Code quality relies on several components. First, sufficient code reviews are critical for ensuring that the code passes static analysis and dynamic analysis reviews. It's therefore good practice for developers to view the current version of the code as well as the previous versions' branches for the sake of thoroughness.

Collaboration is also an essential component. In an environment where developers are more responsible for quality, working together in teams allows for better, more robust code. Peer reviews can provide a second and even third set of eyes for finding errors and suboptimal code.

In general, developers should limit the time they spend reviewing code, as well as the number of lines they look over during a single review period. That's because they're more likely to miss errors or overlook weaknesses when they spend more time reviewing a significant chunk of code. Typically, 60 minutes and 400 lines are good benchmarks to follow.

Those guidelines aside, there are still several ways of reviewing code. One developer can look over the shoulder of another as the first developer steps through the code. Developers can also email sections of code to each other for review, or even work in pairs at a shared workstation via a technique called pair programming.

However, the best approach to review code is to use a specific tool that optimizes the process. Micro Focus Dimensions CM, for example, is a software change and configuration management system for Agile software development environments.

This software includes PulseUno, a web-based user interface that contains highly useful browse and review features. Its peer review capability, in particular, enables developers to look over project changes in collaboration with other team members. Together, teams can examine the health of changes in streams and branches, as well as vote to approve certain edits.

Artifact Management

Another important aspect of code quality is artifact management. Artifacts are by-products of the software development process that can include anything from supportive scripts to final, compiled binaries and libraries.

To avoid accidentally introducing security holes, or undesirable licenses like the GPL, developers must ensure they only pull code from trusted repositories. They also need to regularly check for common vulnerabilities and exposures (CVEs) to make sure they do not pull in inappropriate licenses. It's easiest for developers to do this at the individual level rather than at the CI level, as this ensures the teams don't waste time on work that will later need to be rolled back.

One approach to artifact management is the utilization of vaults, which are secure storage areas curated by an administrator. This system requires developers to pull their resources from the vaults, ensuring such resources are free of threats and properly licensed. A solid vault system supports approval processes and audit trails while detecting vulnerabilities and licenses.

For a vault to be a valuable resource, it should be highly flexible. Developers should be able to create, configure, and delete vaults on both local and remote systems. They should also use approval checklists, which can be attached to specific vaults or set up as a single default checklist for all vaults. This enables developers to specify certain requirements when an approver reviews a vault (for example, developers can specifically require a package to have a specific Apache license and no security issues).

Another important part of artifact management is binary storage. In most cases, an organization has both internal and external binaries. Internal binaries are built from an organization's own software. These packages typically don't need to be approved when published, and developers can consume them as needed.

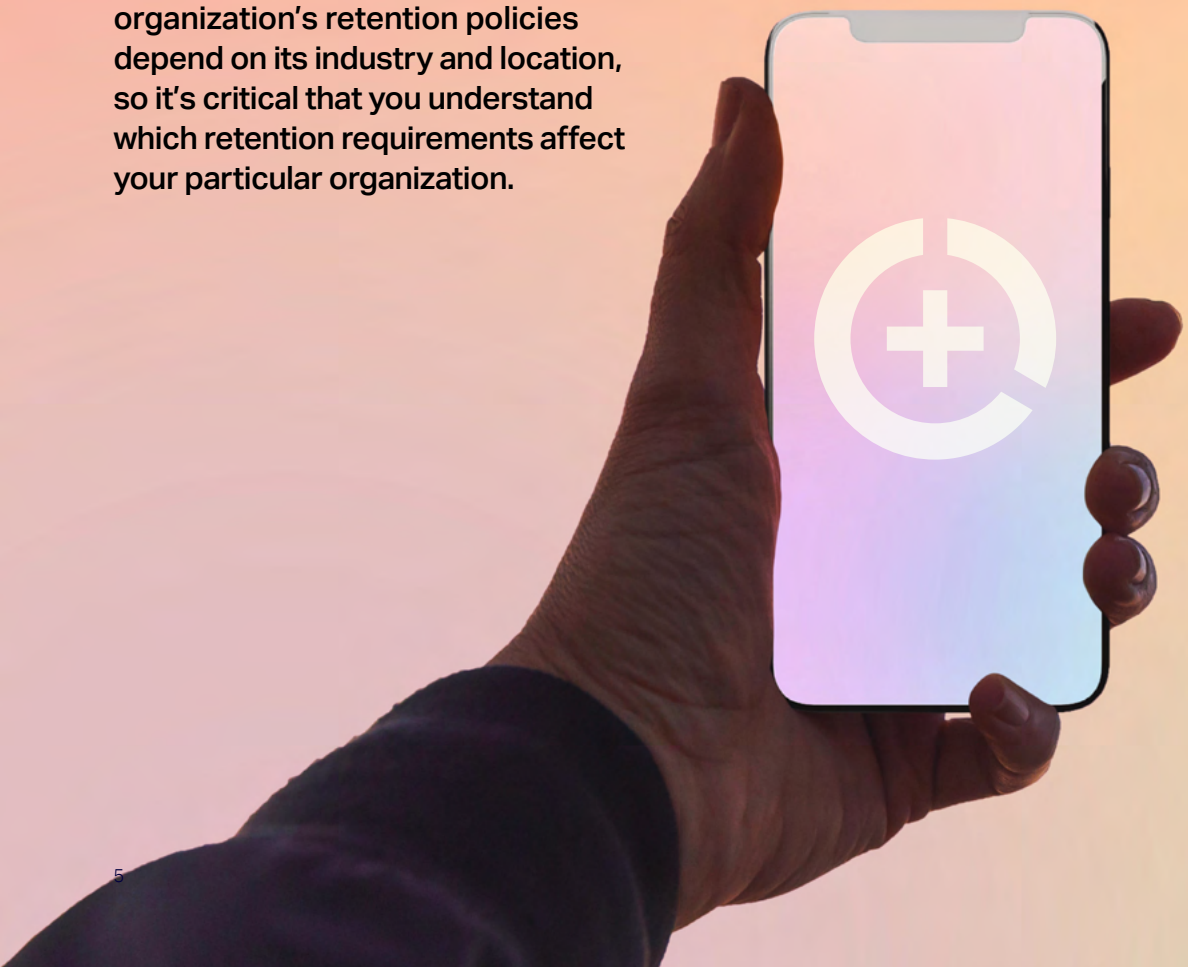
Conversely, external binaries come from third-party vendors. An administrator can set the system so that when someone adds an external binary to a package, developers cannot use it in a build and deployment process until the appropriate authority reviews and approves it.

Retention

All organizations should have strong retention policies and practices in place. Retention is a key component of software artifact management, just as it is for any other data that a business might store.

Retention policies need to reflect both legal requirements and compliance regulations. An organization's retention policies depend on its industry and location, so it's critical that you understand which retention requirements affect your particular organization.

The PulseUno vault supports retention policies to define how long it keeps a vault package. Developers can create multiple retention policies and specify a different retention policy for each vault.



Next Steps

It's time for developers to play a more active role in quality. They can do this by performing earlier reviews of their code and their teammates' code, using vaults to manage artifacts, and implementing a retention policy.

While these approaches may require more work, utilizing the proper tools such as PulseUno can help shoulder most of the load. The net result is that developers can spend more time on new features and less time fixing bugs. To find out how Micro Focus can help you focus on quality in your Agile development initiatives, explore our website.

Learn more