



Strategies for Testing Agile Developed Software



&



=



Start Collaborating Today!

Strategies for Testing Agile Developed Software

In the old days with Waterfall development, a company figured out what to build; designed it, wrote code; tested it; and deployed the product. Testing was usually done late and often separate from development activities. This was often unavoidable because the methodology was such that an application was not ready to test until late in the process. This led to a host of problems down the road such as:

1. Locking down the requirements early, even though they would likely change.
2. Changing requirements were expensive to implement after the fact, and subsequently introduced new bugs in the software.
3. Only out too late was it realized that too much was attempted and the team ended up delivering late on the project, or shipped on-time, with defects, to get it out anyway.
4. Limited success or value is realized until final product delivery.

With Agile development, the approach is much different. Instead of one big delivery, a time box of a few weeks or months is established. Inside the time box, a small feature or function-set is implemented, deployed, and feedback collected. Development proceeds through a series of time boxes, with the functions in each time box implemented in order of priority. This way, concrete progress is achieved early and deadlines better maintained.

Impacts on Testing

With short time box (i.e. 2 weeks), conventional test cycle management does not work. Typically, the last few days of the time box are saved for regression testing. That works fine early in the development process, but as iterations build up, there is more and more to test in a short fixed time. This is compounded by the trend for many, Agile development shops to forgo having a QA Team, leaving the developers to test their own code. This in turn can result in more bugs, not from a code point of view, but from the viewpoint of the end user, the customer.

Handling short testing cycles

To adapt the traditional testing model to Agile development and compress testing while continually increasing the quality of the software under test, there are three key recommendations:

- Test Early
- Automate
- “Bang for Buck” testing

Test Early

As mentioned earlier, the conventional test cycle breaks down with Agile development as waiting to test at the end creates problems. Therefore, for each new feature, function, or bug-fix, test as soon as possible, instead of waiting to the end of the time box. When the end of the cycle is reached, do regression. Most of the bugs then are already out or at least known. Regression is truly regression and the next iteration can begin on time.

Automate

Automate smoke tests to cover the essential functions of the software. The test should be able to run very quickly and should be built upon with every time box for the features in that time box that are deemed stable. This way, the manual testing effort is saved for new features and functions. Once the end is reached, the smoke test scripts can be easily augmented to become acceptance test scripts.

Bang for Buck Testing

With newer feature rich applications and proliferating platform support, the number of test case/configuration combinations escalates exponentially. To manage this, testers need to skip platforms and test cases based on the understanding of the risks of that time box. This requires experience and knowledge of past defects and failures.

To do this, leverage the defect-tracking system. It can be configured to analyze defects by risk, feature, and impact. This enables a tester to find out where defects are occurring most and what tests to focus on or even enhance/supplement. Doing repetitive testing on functionality that rarely, or has never failed is a low return effort especially after the first few time boxes. Focus on functions and features where defects are most often found. Adjust the test effort and test plan to match where the defects are, then drill further to go where the worst defects have been found in the past.

Go Agile but Don't Forget an Agile Testing Strategy

Those successfully implementing Agile testing techniques have to greatly reduce the cost of defects as tests are executed daily as soon as code is checked-in, providing immediate feedback to developers to their most recent code changes. End-user functional tests combined with unit and integration tests helps to identify basic user interface and functional issues that could affect the information architecture. Identifying these basic problems and removing them early also reduces testing cycles and defects later in the project.

With continuous iteration regression testing becomes a burden as the iterations build more and more functionality into the testing. It soon can become very costly with only manual testing. Therefore, testing early and continuously requires iterative testing activities to be automated and integrated into the development process.

Many development shops have converted to the new Agile paradigm. However, new risks and pressures have emerged in the associated quality assurance and testing. Just as the development process migrated out of Waterfall, the attendant testing process needs to be modified as well to mitigate risks, maintain quality, and keep agile.