Improving the quality of Agile-developed applications through model-based testing

A Point of View

Abstract:

This paper provides a point of view on model-based testing and its impact on Agile delivery. Organizations that implement model-based testing see a substantial improvement in functional coverage and improved levels of quality. The paper assumes a knowledge of Agile practices and the roles defined for Agile development teams.

Quality has increasingly become a key focus area for enterprises over the last decade. Significant outages have landed companies in the news and alienated clients and users. At the same time, businesses have found it necessary to drive change faster to provide competitive differentiation. Application portfolios have become increasingly complex utilizing multiple channels, platforms, and integrations. Based on all these contributing factors, application testing and overall quality have become critical business needs. It has become important for enterprises to consider new methods to drive this quality.

In IBM's point of view, the focus of testing is overall quality. Testing needs to mature past the intent of just identifying defects for remediation to defect prevention. In our view, while testing is necessary regardless of the development approach taken (Waterfall, Agile, or some combination), it is Agile development that incorporates testing as a key element of the developing sprint and, therefore, leads naturally to driving quality.

This point of view is a result of the following observations of both successful and unsuccessful teams:

Observation 1: Comprehensive testing can be perceived as being incompatible with Agile development

Enterprises normally adhere to a two-week sprint model, relegating other business factors as secondary to the orthodoxy of the Agile development approach. This is especially common in the early stages of the introduction of Agile[1]. As a result, it is perceived that there is insufficient time to repeatedly analyze, design and build the appropriate tests for each sprint. There are multiple examples where development teams choose to focus only on a subset of tests that lack complete user story coverage to maintain a sprint schedule. This can lead to a release that has a catastrophic defect requiring immediate remediation, lost development time, and slipping release schedules. Thus, the challenge that enterprises face is how to effectively conduct the full stack of

testing—unit, system, user acceptance testing (UAT), regression, etc.—within an Agile development paradigm that will not hinder the Agile process.

In IBM's experience, it is very difficult to construct the necessary test cases from scratch during a two-week sprint. IBM recommends developing and refining a test model in parallel with story development and backlog grooming. This will provide the foundation to generate sets of automated tests with 100 percent functional coverage.

Observation 2: Well-executed Agile development begins with a well-defined user story

Agile works well when there is a common definition, understanding, and vision of the desired end state, and the minimum viable product (MVP) is well-defined. This results in clearly articulated sprints with appropriately decomposed stories that lead to specific outcomes. As a result, the working software that is developed and tested can be assessed against tangible requirements; thus, changes can be decided rapidly and in near-real time. We observe, however, that even in well-executed development, there is less upfront focus on testing (including test automation) than in other elements of the development cycle.

In IBM's experience, as Agile Development begins with a well-defined user story, Agile testing should begin with a well-defined test model. The most successful Agile development teams are those where testing is central to the design and definition of the application, and where the test automation is kept current with the user stories in the sprint. Where development focuses on the business perspective, our experience leads to higher quality applications, reduced development time, and less defect remediation. All these factors result in lower overall costs.

Observation 3: If comprehensive testing is going to be adopted, it needs to be rapid and flexible

By its nature, Agile development provides a user-centered view of product outcomes. This view determines if the overall effort is on track or requires a mid-stream course correction to deliver on the desired outcome. Therefore, any process element inhibiting speed is unlikely to be included. We believe model-based testing accelerates the development process by generating test cases and test automation while reducing the number of test cases needed to achieve the desired coverage. The ability of model-based testing to focus on the right tests reduces the time needed to test, as compared with brute-force testing. For this reason, model-based testing supports Agile development initiatives.

In IBM's experience, model-based testing can reduce the test execution cycle by 40 percent, reduce the number of test scripts needed by over 50 percent, and reduce the number of defects entering production by 35 percent [2].

Observation 4: Test automation is critical to successful comprehensive testing in Agile environments

IBM observes that automation is difficult to create and maintain in an Agile paradigm due to the short nature of an Agile sprint. Many companies create automation one sprint behind or have a separate automation team work to develop the automation in parallel. To drive ongoing successful Agile testing, IBM believes that it is very important to create automation within each sprint itself.

In IBM's experience, comprehensive automation can be successfully created in the sprint by reviewing and updating the test models in parallel with backlog grooming and story development, and having the capability to generate automation from the models. The automation framework needs to integrate with test data to facilitate the execution of automation. It is also critical for the framework to be able to work on a wide variety of application environments.

Observation 5: Agile typically focuses on integration

The focus of successful Agile sprint teams is always to develop robust, defect-free code to deliver the desired feature. It is not to ensure that all the various features work together. There is trust that interfaces between one code component and another will work if every team follows the design criteria established in Sprint 0 (or at least that is the theory). In reality

of course, life is not as simple, which is why unit testing alone can lead to failure as discussed in Observation 1. For this reason, we see the following as key to ensuring the delivery of a quality application: integration testing, regression testing, and UAT, among others. However, these test functions are incompatible with the activities of a sprint.

In IBM's experience, it is critical to have a mature, automated integration testing capability that can be continuously executed to validate the integration with the application portfolio. This capability can only be created through ongoing generation of automation from the sprints.

Observation 6: Testing needs to transform from defect detection to defect prevention

No matter how effective and efficient testing is, it is critical to begin to change the focus of testing from detection to prevention. This will improve overall quality and drive the cost of testing down. Model-based testing contributes to this effort by starting the development and updating of models early in the sprint. It is also important to utilize defect patterns and trends to change the issues and problems which caused these defects in the software development lifecycle.

In IBM's experience, IBM's Cognitive Defect Analytics can significantly improve both the testing process and to help drive overall quality across the software development lifecycle (SDLC). Defect classification enables the team to assign tasks to the optimum developer's backlog to resolve. Analytics and prediction can be used to assess the defect against history to identify how best to resolve the issue. Analytics can further be used to classify the defects and prioritize remediation. Proactive defect prevention can be delivered with pattern identification and fast investigative analytics can drive changes to the way that the software is being developed and tested.

So where does this leave testing leadership?

In order to address these observations, it is best to separate unit testing from the other testing stack elements with the result that:

- Unit testing should be performed by the developers aligned with Agile development approaches and gain expertise with the specific application being developed.
- Model-based testers are assigned, when needed, working with the sprint teams. In this model, they can support more than one application at a time as experts in the test methodology. In IBM's point of view, organizationally, a central resource pool of model-based testers is appropriate. Leadership needs to create a distinction that recognizes that developers utilize unit-testing techniques to focus on identifying, minimizing and remediating code defects within the sprint and model-based testers focus on the business outcomes desired from the application. It is through this combination that application organizations can deliver high-quality products, rapidly and with confidence that they will meet the needs of their customers.

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
February 2018

Please Recycle

[1] *Please see IBM's whitepaper entitled "Why Agile works … and Why it doesn't"*

[2] *These are results achieved with existing clients.*

19013419-USEN-00