CONFORMIQ

A Conformiq White Paper

Best Practices for Improving the Quality and Speed of Your Agile Testing

Abstract

With today's continually evolving digital business landscape, enterprises are increasingly turning to Agile approaches to speed up development and address growing consumer demands for innovation. Agile approaches aim to build quality assurance into product development from the ground up, by having developers heavily involved with testing from the outset. The idea is that if problems can be found and corrected earlier in the design process, companies can get defect-free products to market more quickly. Unfortunately, Agile is often unable to deliver on its promise of early, aggressive, and continuous testing, because many of the testing approaches being used today are insufficient to get the job done.

Below we explore some of the popular alternatives to traditional manual testing, including those being used to improve the speed of functional test design. However, it is important to note that speeding up test design is not the same as improved test design productivity. Even in Agile programs, the focus must be on improving the overall project testing process, not just faster test design. Conformiq 360° Test Automation solves this issue by automating the entire systems development life cycle (SDLC) process, in addition to automatically generating test designs and executable scripts at the speed of development. This paper is intended to provide more insight into these older developer test design methodologies, what they do and don't deliver, and then to compare and contrast them with the newer Conformiq 360° Test Automation process.

Background

One of the processes promoted for speeding test design is Test-Driven Development (TDD). It is a core part of Extreme Programming (XP) and other "light weight" development practices and, though not a core part of Agile development, is a common partner to Agile.

As originally described by Kent Beck, TDD meant that before a developer could add a feature in the software, he/she first must write a failure test case. Their next objective was to write the minimal code that would pass that test case. Once the test passes, they refactor the code making sure that the test still passes. More broadly, TDD is used to describe any process where tests for a feature are written before the feature is implemented.

In practice, TDD has been considered too unstructured, so Behavior-Driven Development (BDD), as an enhancement over TDD, is currently

receiving significant interest as being the "next" great test improvement process for Agile development projects.

BDD is a software development process that aims to combine the techniques and principles of TDD and Object-Oriented Design by leveraging ideas from domain-specific designs. It has excellent philosophical goals and ambitions, as BDD fundamentally aims to engage all stakeholders in the software development process by enabling Subject Matter Experts (SMEs) and non-technical stakeholders, such as business analysts and customers, to contribute and collaborate in the process by writing user stories.

In principle, BDD is founded on the use of a simple and informal notation, which is very close to common language and based on the main concepts of features and scenarios. Scenarios detail the "desired behavior" for each feature; they are essentially acceptance tests in the form of user stories. Probably the most widely known and used notation is Gherkin, which is used by tools like Cucumber, FitNesse, and JBehave.

Here is a Gherkin example:

- **Feature**: The online shop keeps track of goods in a shopping basket.
- **Scenario:** Put an item into an empty shopping basket.
- **Given** the shopping basket is empty.
- **When** user adds one item to the shopping basket.
- **Then** the shopping basket should contain one item.

It is easy to see what is happening here. But if you take another look at this example, Gherkin is really nothing more than a requirement and an informal test description. This informality is both a strength and a weakness when it comes to test automation. Automation frameworks revolving around Gherkin can only generate simple code stubs, and still require a significant amount of implementation by software developers to get these codes ready for execution. Each Gherkin scenario clause is "just text," or text created with the preferred wording by the Gherkin author. Since automation codes need to be written manually and mapped to that description, dealing with change management is a significant issue. Scenarios make it very difficult to assess the quality and completeness of your software testing.

For example, how much have you actually tested your application? The use of Gherkin does not guarantee systematic coverage of functionality. Have you really tested all the possible data combinations? Which data combinations actually make sense? Do you have scenarios that fully cover all decision points within your functionality to be tested? Have you considered boundary values?

Overall, BDD and TDD are extremely limited in supporting Agile development because they are unable to deliver a complete testing process and actually reduce software developer efficiency. Also, because they were developed prior to automated testing tools, they don't account for improved efficiency from advancing technology.

Challenges

While TDD and BDD testing can be used in an Agile process, the results show that with these methods, improved speed comes at the cost of quality and knowledge, especially the understanding of test coverage. Other issues include:

1. TDD was created to match the need of a software development process with short development cycles. Unfortunately, constant time to market pressure makes it difficult to maintain and update the (regression) test suites.
2. Test cases written by developers were created to cover their own code. They did not fully cover the whole *system operation*, or take into account how the multiple code parts written by different developers would run together. This meant that additional system end-to-end test cases needed be written later in the process,

often causing critical system-level defects to be found much later.

3. Because the developers wrote their own tests, this took away time from writing code and subsequently reduced their design efficiency.
4. Finally, the process of having the developer test his/her own code goes fully against the tenets of the Independent Verification and Validation (IV&V) process. Systemic defects can slip through the developer's "blind spots".

The TDD/BDD testing process is shown below.

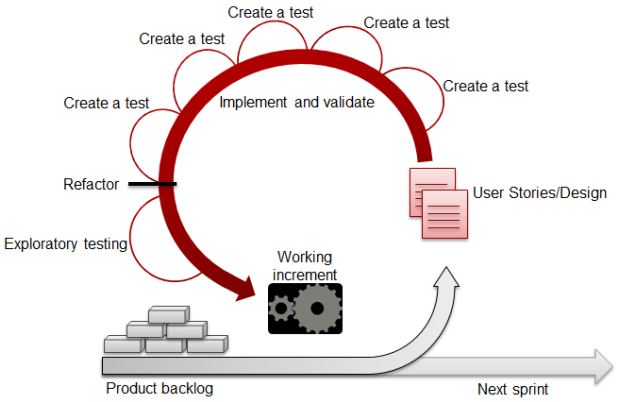

Figure 1: The TDD and BDD testing process

A More Advanced Method is Needed

This is where Conformiq 360° Test Automation comes in, as a transformational approach which includes Model Based Testing (MBT) technology and more. This next-generation solution enables testers and software developers to complement the work done by business analysts, system engineers, and customers by generating the tests they created as both Gherkin scenarios and test automation scripts for the entire test logic as code that includes test data. Tests are automatically derived and generated by Conformiq tools, which provide systematic and repeatable coverage of the functionality to be tested. As requirements change, the model can be quickly changed to match them, and all generated scenarios and test automation codes are automatically updated, eliminating the issue of maintenance. Requirements can be directly downloaded from requirement management tools linked to the Conformiq models; requirements traceability is automatically established.

The Conformiq 360° Test Automation approach to model-based black-box testing starts with simple, high-level formal models of the system under test (SUT) that is being designed, and then automatically generates test cases. The model of the system can be continuously modified in parallel with development of the system itself. With TDD, you write a test case for a feature before you write the feature. When Conformiq 360° Test Automation is applied, you augment your SUT model to express your feature, then Conformiq automatically regenerates the tests (which will include one or more tests relating to your new feature), before you write the feature.

In our experience with industry use cases for Conformiq 360° Test Automation, we have found two key transformations for testing advantage. First, the productivity improvement for actual test case generation versus manual creation of tests is significant, on the level of an order of magnitude. Second, this productivity improvement is even higher in the context of incremental development: Conformiq tools automatically regenerate the full test suite when the model is changed, including determining which prior test cases are no longer applicable. It also automatically generates the test oracle, i.e., the expected correct test execution result.

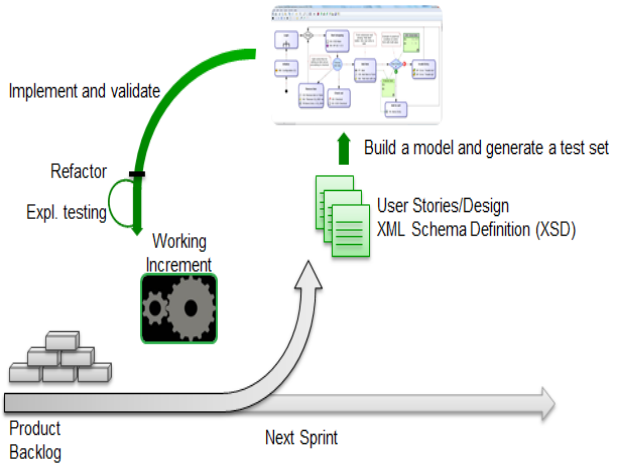

Figure 2: Conformiq 360° Test Automation method

What Are the TDD/BDD Issues that Conformiq 360° Test Automation Solves?

These real-world experiences argue that Conformiq 360° Test Automation is particularly well suited for Agile, especially as an improvement to TDD methods. In addition, it can help in improving some of the inherent issues associated with TDD and BDD, including the following challenges.

Poorly understood requirements.

With TDD, the requirements are not well understood early in the process, and they typically contain ambiguities, omissions and contradictions. One benefit of Conformiq 360° Test Automation is that just the act of modeling the system behavior often improves the quality of the requirements specified. This means many defects can be spotted early, in modeling the specifications and requirements, before a single line of code is even written. As you create a model of the system behavior, you often raise many questions regarding the requirements, so already the modeling process can expose a lot of issues with the requirements. This should not come as much of a surprise. After all, system modeling involves the development of a compact high-level prototype of the real system, and it has been long known that prototyping is a good and efficient way of finding inconsistencies in the requirements.

Varying requirements.

This is especially important within Agile development projects where the requirements are updated during the project. In Conformiq 360° Test Automation, a simple formal model of the SUT explicitly embodies the requirements. The refactoring of those requirements with a model change, and automation, is dramatically less effort than refactoring a set of manual tests. With the Conformiq approach, the effort corresponds only to the number of requirements that change. In contrast, with a manual process, all test cases need to be checked for all requirements that have changed.

Poor/Inadequate Tests.

With TDD, the developers have not implemented the solution yet, so the tests are typically not "good enough" to explain the solution. With Conformiq 360° Test Automation the idea is that the *model represents the actual, desired behavior of the system itself* – not the test cases nor how it should be tested. It improves the quality of the test cases because this automated approach to test design *lowers the risk of having incorrect, missed and redundant tests*. An engineer can, for example, accidentally miss a test case that is dictated by the requirements, such as for an error-handling case, a limit value of a data parameter, or an expiration of a rarely activated timer, but not so with the Conformiq algorithmic approach.

Unit tests instead of system tests.

As mentioned previously, TDD test cases are written by developers and cover only their own code. These unit tests do not cover the system operation, because they do not cover the operation of multiple code parts written by different developers all running together. This means that, with the TDD approach, additional, end-to-end system test cases must be written at a later time, often leading to a delay in

finding the system level defects. With Conformiq 360° Test Automation, these system tests are automatically created as the model grows or as models are combined into the full system.

Over-fitting tests to the code.

A common concern is that if a developer writes the tests first, he may over fit the actual implementation to the tests. With Conformiq 360° Test Automation, the developer doesn't design the test cases, so there is no risk of fitting the implementation to the tests. A key point of model-based black-box testing is that the system is judged against an independent reference. Without this approach there is naturally a possibility that the developer reflects the same fault both in the test and in the implementation code.

Reluctance to do early testing.

Adherents believe that tests should be guided by code as well as expert knowledge of the implementation on where the problems might be. Therefore, with TDD, implementing tests too early is viewed as unnecessarily expensive, because you do not have the details of the code available. This is a very common white-box view to the problem, where we expect to have access to the implementation details for devising test cases. However, with model-based black-box testing, we approach the problem from a different angle. We assume no implementation details of the system, and validate whether the given system conforms to its design and functional specification. The test cases that Conformiq Creator™ generates (using MBT technology) are black-box by nature, which means that they depend on the model and the interfaces of the SUT, but not on the internal structure of the implementation. You do not have to understand how the system has been architected internally to create a model, thus lowering the cost of test creation and allowing tests to be generated prior to incremental code drop.

Developer's lack of interest and/or ability to test.

Testing requires a different mindset from development, and some developers are poor in doing test design. This issue can be resolved by pairing developers with people who know how to test. When the Conformiq 360° Test Automation approach is applied, modeling can be accomplished by a non-developer (an SME, a modeler, etc.) who is an integral part of the development team. This means that developers do not need to also be testers. So rather than spending their time writing test cases, instead developers can spend their time writing code. This is a key benefit in an Agile delivery process with short sprint times. In addition, this approach supports the tenets of the IV & V process, and helps to reduce the errors that naturally result when the same person writes both the testing assets and the code.

Conformiq 360° Test Automation Approach

Conformiq has developed an approach to model-based black-box testing that starts with simple, high-level formal models of the SUT, then automatically generates test cases without further user involvement or direction. The model of the system can then continuously be fleshed out and easily updated in parallel with development of the system itself.

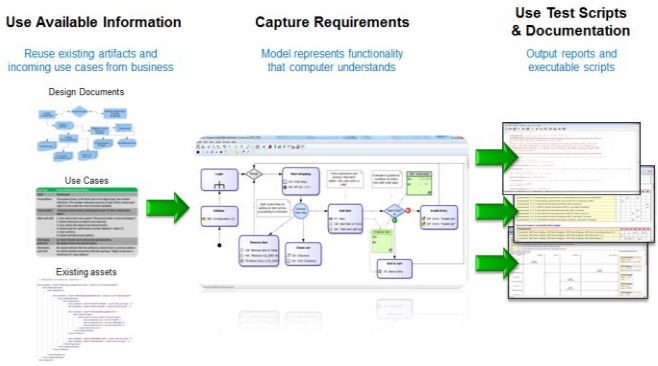

Figure 3: With Conformiq 360° Test Automation, the model is manually created from the requirements and test scripts are automatically generated for execution

Because Conformiq automatically generates test cases, they are consistent in how they are written, whereas manually created BDD textual test cases vary

by author. Beyond test cases, Conformiq ensures that testing is done well, with high coverage, and that all stakeholders have the knowledge they need.

Can these Processes be Used Together?

Instead of seeing BDD and MBT as competing approaches, we can view them as being complementary. It can be beneficial to integrate the two approaches to get the best of both worlds. Instead of requiring and relying on software developers to "connect the dots" through manual implementation of code stubs and maintenance of every clause in every scenario, the Conformiq solution automatically updates a powerful MBT system model. The model is then processed by Conformiq's engine, which generates an optimal collection of tests that can be exported as Gherkin scenarios for business analysts, system engineers, and customers for model review. These can also be executed automatically using frameworks such as JUnit, Cucumber, FitNesse, and output for execution with HP/UFT, Selenium, and other frameworks.

Summary

The promised speed of Agile development is a key to its appeal and broadening use. However, testing methods developed over a decade ago are insufficient for delivering quality at the necessary speed. The innovative Conformiq 360° Test Automation approach for test design eliminates the inherent problems with those previous developer-centric testing methods, and instead introduces a next-generation methodology that delivers productivity, quality and documentation at the speed of development.

Author Kimmo Nupponen is the Chief Scientist at Conformiq. He has been developing automated test design software for over ten years. He understands what is needed for real world use and the differences between tool engines.

Conformiq is transforming software testing with Conformiq 360° Test Automation™, providing the most sophisticated and comprehensive automated test design solution in the industry. The unique Conformiq 360° Test Automation technology enables the next generation of testing: transforming, streamlining and automating even the most complex system-level testing environments. Conformiq 360° Test Automation improves efficiency with a 40% faster test case development cycle; enables delivery of higher quality code with 50% more defects found; increases manageability with 50% better collaboration: and reduces costs with a 400% return on investment. Conformiq serves enterprise IT, communications and embedded software markets worldwide. Privately-held Conformiq is headquartered in San Jose, California, with a worldwide delivery and support organization including offices in Finland, Germany, Sweden, and India.

www.conformiq.com **sales@conformiq.com**

USA
4030 Moorpark Ave
San Jose, CA 95117
Tel: +1 408 898 2140
Fax: +1 408 725 8405

FINLAND
Westendintie 1
02160 Espoo
Tel: +358 10 286 6300
Fax: +358 10 286 6309

SWEDEN
Stureplan 4C
SE-11435 Stockholm
Tel: +46 852 500 222
Fax: +358 10 286 6309

GERMANY
Maximilianstrasse 35
80539 Munich
Tel: +49 89 89 659 275
Fax: +358 10 286 6309

INDIA
29 M.G. Road Ste 504
Bangalore 560 001
Tel: +91 80 4155 0994

 V1115