28TH ANNUAL
PACIFIC NW
SOFTWARE
QUALITY
CONFERENCE

OCTOBER 18TH – 19TH, 2010

ACHIEVING QUALITY IN A COMPLEX ENVIRONMENT

Conference Paper Excerpt from the

CONFERENCE PROCEEDINGS

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commecial use.

Acceptance Tests Driving Development in Scrum

Sari Kroizer – Validation Team Leader

Sari.Kroizer@intel.com

Abstract

This paper demonstrates the challenges faced by the validation team when implementing Scrum, and the benefits gained by adding an Acceptance Tests Driven Development (ATDD) methodology. The paper will share with you the experience of our Scrum team on a large software project (23 sprints). The paper includes actual data on test coverage and lessons learned.

Author Biography

Sari Kroizer, Validation Team Leader and Scrum Master in Intel Israel, works with the team and the managers leading process improvements using Scrum and validation methodologies.

She has formal education in software engineering and has experience in programming, web development, and firmware/software validation.

1. Introduction

Validation testing as an integral part of the Scrum process is a big challenge. The cycles are short, regression testing is more necessary than ever, and quality is a requirement for story completion. Under these conditions, the testers need maximum flexibility and good tools.

The ATDD methodology provides a quick solution for integrating validation testing in Scrum. Automated acceptance test cases are designed and developed in parallel with or even before the features.

From our experience, using ATDD with Scrum significantly improves both the development and testing processes. The ATDD methodology helps the whole team understand and focus on the customer's requirements. This is a major goal of Scrum. ATDD helped us to combine the validation into the Sprints, to improve the quality, and to raise the team satisfaction. ATDD also improves the use of testing resource time, including optimization of tester personnel ramp up. And last but not least – it's really fun!

This paper includes analysis (with actual data) of the key problems we faced when implementing Scrum and how we solved them by adding ATDD.

I suggest implementing ATDD on long projects, where the automation is most valuable, and to run manual tests in addition to the automated acceptance tests.

2. QA Environment in Scrum

As the validation team leader, it was my challenge to adapt our validation process to the complex Scrum environment. This required a new mind set: No more long test plans, fully-coded features, or project-long tests cycles. In the Scrum model, the validation process must be as agile as the development process.

We decided to include the software testers in the Scrum team, and to perform validation as an integral part of the sprint. We also defined that the quality of a story is a prerequisite for story completion. This means that a full test cycle is done on each story as part of the sprint, and all the critical bugs and most non-critical bugs are fixed. The number of bugs a story can have at the end of the sprint is set according to the story size.

We also needed to take responsibility for the complete project validation, and to make time for integration and regression tests. In Scrum, the project is incremental, and the deliverable for each sprint is stable and validated software **with** the additional new stories of the sprint.

From the beginning, we recognized that we would need an automated regression test suite for our project. We hoped that we would be able to use this automation for future projects as well. We started to work on the new stories, and tried to build a set of automated tests to help us with the repeated regression cycles, in parallel to the full cycles. The Scrum team included five developers and five testers, and the tests cycles were part of the sprint. The duration of each sprint was three weeks.

Progress of the stories is recorded using a "board" (see Figure 1). In the two first days, the high level design documents and the test cases for the new stories are prepared and reviewed. When the developers finish implementing the new story, the story is moved to "Under Test". The test cycle starts and has to be completed before the story can be moved to "In Debug". After most bugs are fixed, no critical bugs exist, and the tester agrees, the story is moved to "Done". We also had stories for regression tests. Bugs found in the regression test cycles, were scheduled to be fixed in the sprint. A quality criterion was defined for each sprint release.

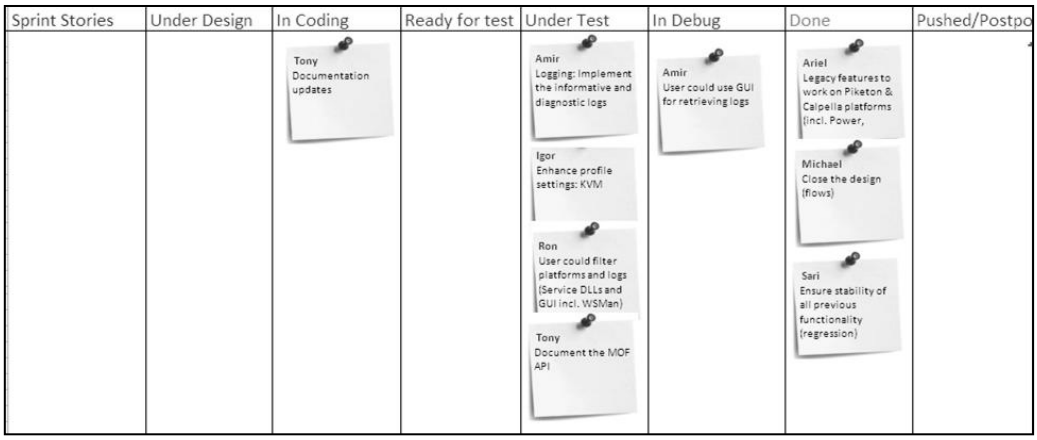

Figure 1: Sprint 13 Board – At the end of the Sprint

3. QA challenges in Scrum

This section describes the main challenges we found when implementing Scrum without ATDD.

3.1 Idle Time

During the sprints, team members often cannot start their own sprint tasks until other team members complete their tasks. This causes 2 main periods of idle time for team members:

- **Testers** – After the first 2 days of design and writing test cases, the testers must wait for developers to deliver new features.
- **Developers** – After delivering features to the testers, the developers must wait until the testers complete the first test cycles.

3.2 Wasted Time on Unstable Features

It can take a long time to stabilize new features. The developers complete code and deliver a feature and the testers start the test cycle. After some manual tests, the testers often realize that a main flow of the feature is not behaving as expected and is blocking other tests. The developers fix the issue and the testers repeat the test. If the feature still does not work correctly it must be returned to the developers again and the "ping pong" game continues.

Sometimes the code of a feature is completely changed during these iterations. If the testers try to continue with tests that are not blocked, they will need to repeat them again because the code has changed. In addition to wasting time , which is valuable since it's constrained to the sprint cycle, the process can be frustrating for the testers.

This situation occurs because developers, unlike testers, do not usually focus on how the customers expect features to work. Also, the testers sometimes do not focus on the high priority tests first.

3.3 "Mini-Waterfall"

One of the main problems is that the Scrum process can unintentionally become almost the same as this description by Hector J. Correa in "Introduction to Scrum":

"A common misunderstanding is to see the Scrum process as a mini-waterfall. Just having short iterations is not enough to make Scrum work. For Scrum to shine it is vital that all team members work together on the same goal during the Sprint. If developers are coding a set of features but QA is testing a different set of features they are not working together. You are not going to have potentially shippable code at the end of this Sprint"

If the team does complete a story in a sprint, the test cycle cannot complete on time. This can leave several stories in the "Under Test" status at the end of the sprint (see Figure 1).
Thus validation completion is postponed to the beginning of the next sprint and the sprint ends with code that cannot be released.

While the cycles are running, the developers are already focused on coding the new stories. Bugs from stories in previous sprints get second priority and the technical debt of unresolved bugs and uncompleted stories starts to increase.

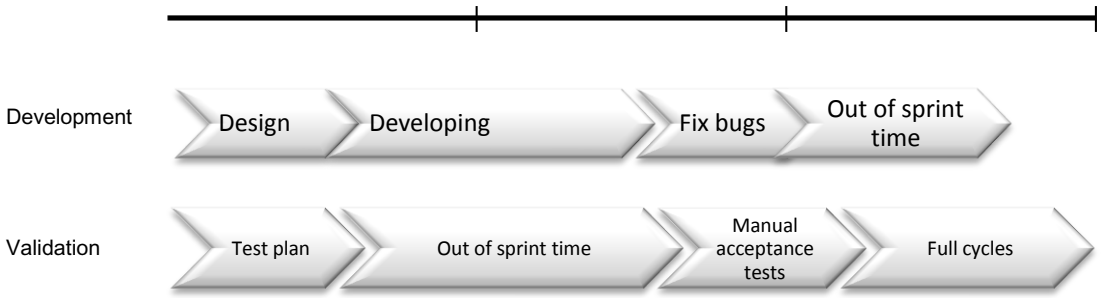

Figure 2: Gaps and Overflows in a Three Week Sprint

3.4 Missing Automation

The first priority and focus of a sprint is validation of new stories. Automation tasks are assigned to "free time", and the automation progress follow up is not part of the sprint tasks follow up. The validation team tries to make progress coding the automated tests in parallel to the sprint tasks. This mode is problematic.

To solve this problem we tried to add “Automation Enhancement” stories to the sprint backlog. Because they were not “real” customer stories they received low priority and were usually the first to be pushed out of the sprint. This caused slow progress in automating the test cycles and therefore we had to run the regression manually.

4. Adding the ATDD Methodology to Scrum

This section describes ATDD and how we implemented it in our Scrum process.

4.1 What is ATDD?

The Acceptance Tests Driven Development (ATDD) methodology extends the Test Driven Development concept to a higher level. In ATDD you start implementation only after you have defined the specific customer valued functionality you want the system to exhibit.

ATDD includes the following:

- **Acceptance Tests** – Specifications for the desired behavior and functionality of a system. For each story, an acceptance test defines how the system must handle certain conditions and inputs and the expected outcome. The tests are for expected results (behavior), not internal structure (methods).
- **ATDD Process** – Each story must include a clear acceptance criteria defined by the customer. The testers and developers define together the acceptance tests cases for the story. The next step is to automate the acceptance tests. These tests will fail until the story is implemented. When the story is ready, the acceptance tests must run and pass. The functionality cannot be considered as “implemented” until the acceptance tests pass.

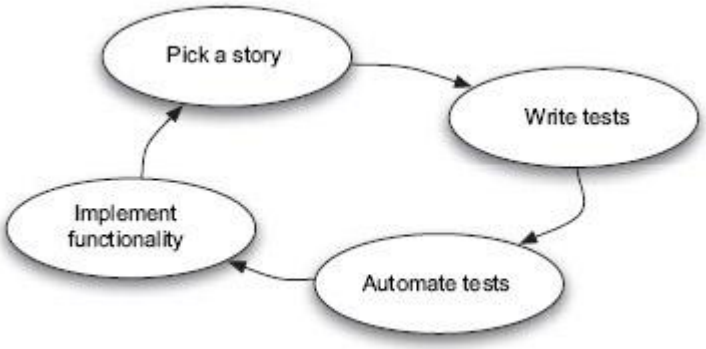

Figure 3: The ATDD Cycle

4.2 Our Implementation of ATDD

For internal reasons, the original Test Driven Development (TDD) methodology was not adopted by the developers. The validation team pushed for the adoption of ATDD in our process. We included the acceptance tests automation tasks as part of the validation tasks, and executed them in parallel with features development.

Preparation

The first effort was to set acceptance criteria for all the stories in the product back log. This task was done by the product owner, with help from the architect and the validation team leader. The product owner worked with the customers' representatives to validate the acceptance criteria, and to make sure that they fit the customers' needs. The completed acceptance criteria for the stories then served as the basis for our test plans.

The Sprint Process

In the first two days of the sprint the high level design of the stories and the test cases are written. During this time the acceptance test cases are designed as well, according to the acceptance criteria. These test cases are reviewed by the developers, and the entire Scrum team must commit to them. While the developers start to implement the stories, the testers start to automate the acceptance tests cases. The interfaces are implemented first. This allows the automated tests to use the interfaces even before the functionality exists.

We added a new column to our board: "code completed". When a story is fully developed we move it to "code completed". The automated tests are now ready to run and the testers start running them. Then the refactoring process begins. The full validation cycle is not started, until the story passes the automated acceptance tests. Then we move the story to the "ready for test" column. When the new features pass the acceptance tests the validation cycle continues with a full manual tests cycle, including less important flows, negative tests, documentation tests, etc.

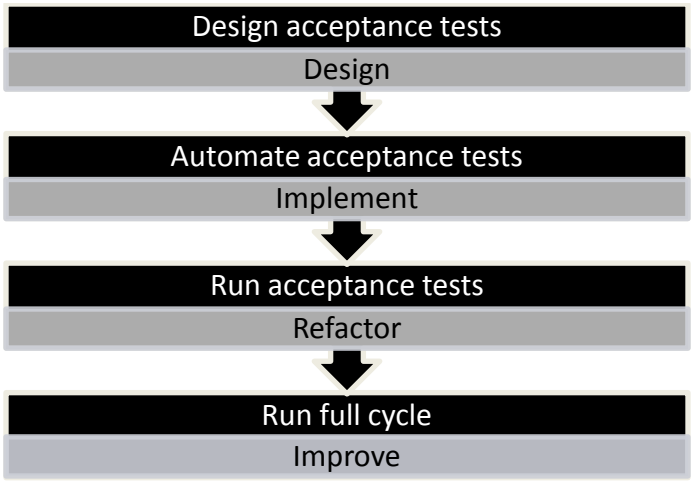

Figure 4: The Sprint Process

4.3 Example

This example demonstrates how we implemented ATDD in a story called "User could configure remote access settings to the firmware".

The acceptance criteria for this story were defined by the product owner:

1. *The user can set remote access settings including: address, port, trigger, certificate, and a strong password.*

2. *The user can configure and reconfigure the firmware using the remote access settings.*

The sprint started and this story was picked up. During planning, the high level design and the acceptance tests cases were defined for the feature. This table shows the acceptance test cases, according to the acceptance criteria (committed by the team):

Test	Description
1	Configure the firmware with remote access settings: a. Set the remote access settings in the service. b. Perform the service "configuration" command with the created settings. c. Check the settings in the firmware.
2	Change the remote access settings in the firmware: a. Change the remote access settings in the service. b. Perform the service "re-configuration" command. c. Check the settings in the firmware.
3	Delete the remote access settings from the firmware: a. Disable the remote access settings in the service. b. Perform the service "re-configuration" command. c. Check that the settings were deleted from the firmware.

We had more tests in the test plan: Advance functionality tests, negative tests on the settings fields, usability tests, system tests, checking how the firmware implements the settings, and documentation tests. (Note that the additional tests are not acceptance tests.)

While the developers were developing the feature, the testers implemented the automated acceptance tests. After one week the code was completed and the automated tests started to run. During the first test run, Test 1 passed but Test 2 failed. It took two days for the bug to be fixed, and some implementation changes were required. During this time, the testers continued to work on other stories, and didn't waste their time on the full cycle of this uncompleted story. The automated test was run again and again, until the story passed.

After the acceptance tests passed, the testers started the full cycle and completed it in 3 days. Some bugs were found during the cycle, but no additional huge implementation\design changes were needed. The bugs were fixed before the sprint ended and at the end of the sprint the story was "Done".

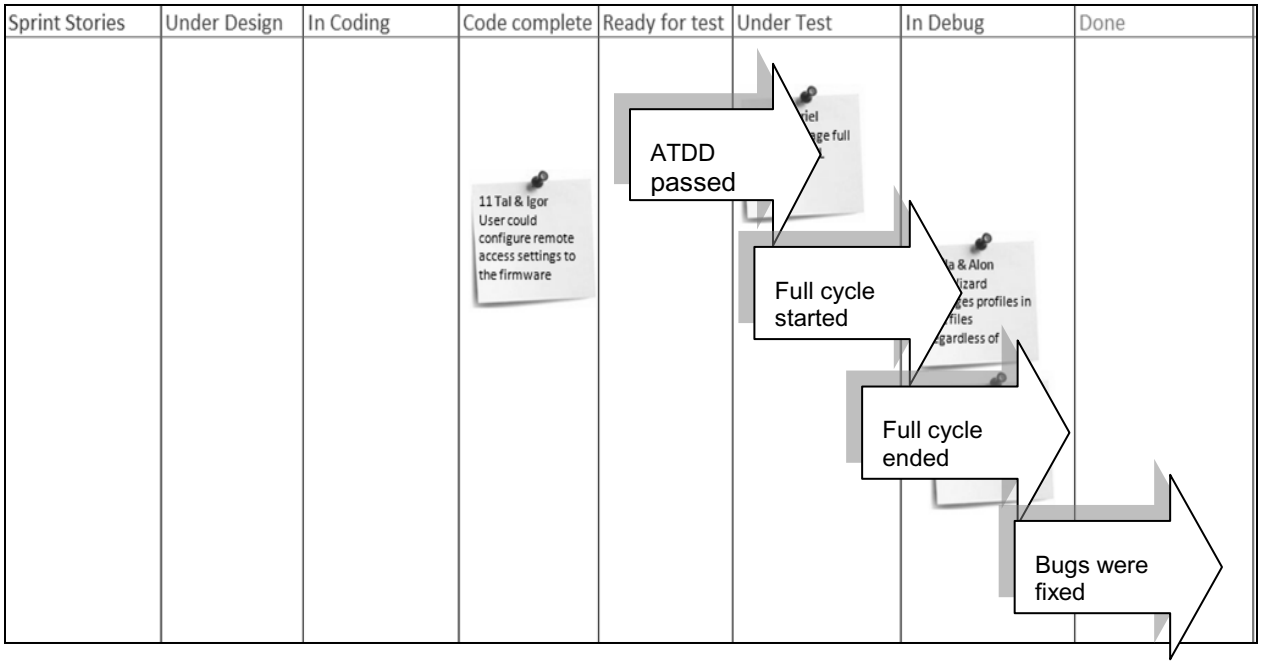

Figure 5: The Sprint Board with ATDD

5. Results

This section describes how challenges encountered in implementing the Scrum framework are solved by adding the ATDD methodology.

5.1 Less Idle Time

The validation team members use the first part of the sprint to automate the acceptance tests. While developing the automated tests they gain experience of the feature. This experience and the fact that the stories must first pass the acceptance tests also greatly reduce the duration of the manual tests cycle. While the testers do the manual tests, the developers fix the bugs and improve the code.

For example: In a sprint with 7 stories, before implementing ATDD the testers would have wasted time (at least 4 days) waiting for the features. With the ATDD, this time was exploited learning the new features and writing the automated acceptance test cases for each of the stories. 16 test cases were implemented and could be run automatically.

5.2 Less Wasted Time

Because the acceptance tests are focused on the customer's requirements and include the most important main flows, they give quick answers about the status of a story.

The testers that run the tests-already know the interfaces and are aware of the acceptance criteria. This means they don't waste time learning the feature during the tests. In addition, the acceptance tests are automatic, and are easy to re-run (and to improve them while running).

The focus of the developers on the customer's requirements is also improved. They know that if the acceptance tests fail, the story cannot progress to the ready for test status. Their code must address the acceptance criteria which are based on the customer's requirements. Also they generally run better sanity tests to make sure that the story will pass the automatic tests.

For example: Using ATDD, a story was tested in 6 days (2 days running ATDD and 4 days for the rest of the cycle). Without ATDD, this story would have taken at least 10 days to run.

5.3 Back to Scrum

Including ATDD prevents the team from slipping into a "Mini-Waterfall." All the team members work together on the same goal during the Sprint. The developers code a set of features and the testers test the same set of features.

The duration of the acceptance tests cycle decreases because the features are more stable and fit the requirements. The full cycle time also decreases because the testers have the knowledge and the experience, and the main flows already work. Most of the critical bugs are found when running the acceptance tests, and are fixed immediately. The rest of the bugs are found and fixed during the cycle.

This means that the stories can be "done" in one sprint, and the sprint ends with code that can be released.

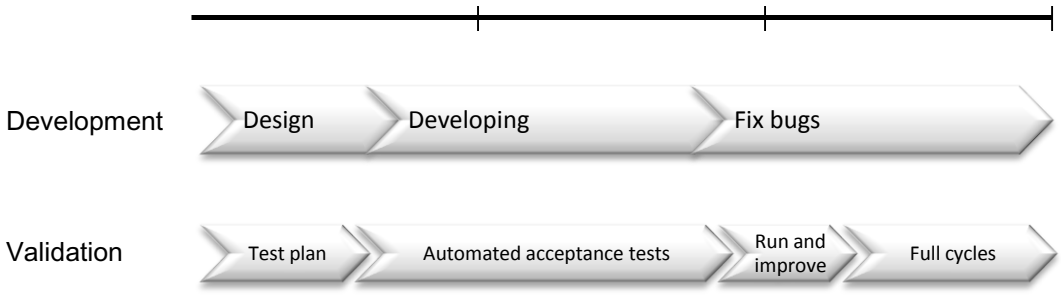

Figure 6: Three Week Sprint with ATDD

5.4 Automatic Regression

The automatic tests written as part of a sprint are re-used in the next sprints for the regression test cycles. This package of tests increases with the sprints, and makes sure that all the main flows are covered by automated tests. This lets you run many more tests in the next sprints without wasting time on manual regression testing. This greatly improves the regression test cycles. The automated regression tests can also be used for future projects.

After introducing ATDD to our process, the average number of tests run per sprint increased from 64 to 150. This increase in tests helped us find many more bugs and improve the quality (see Figure 7 and Figure 8).

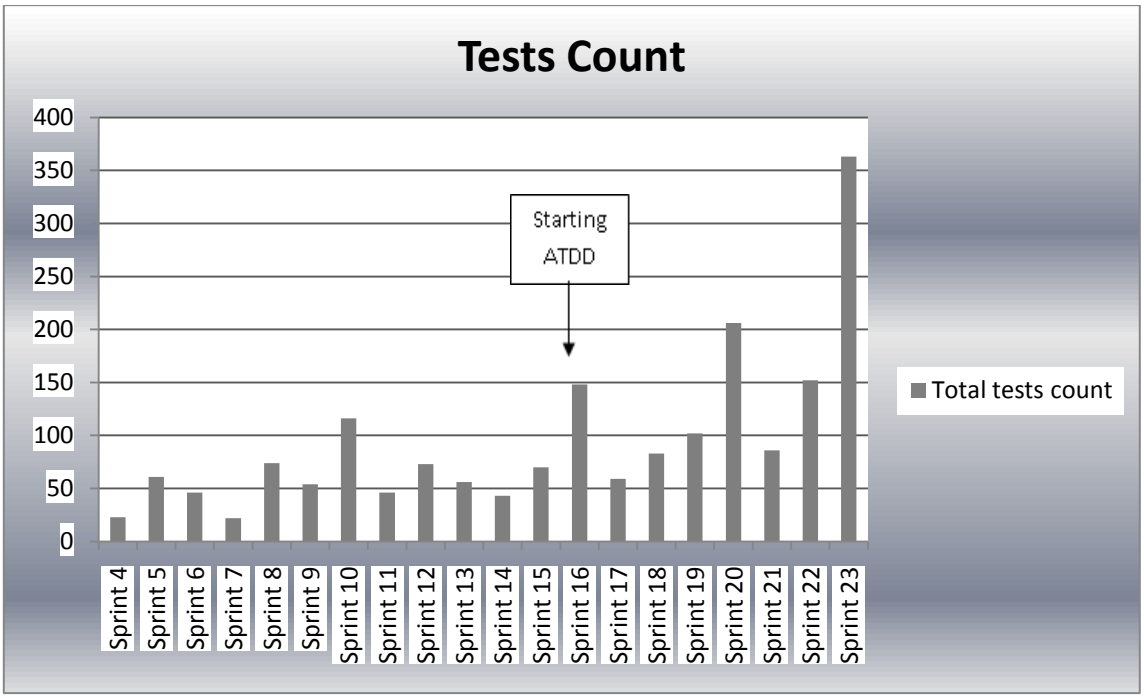

Figure 7: Tests Count

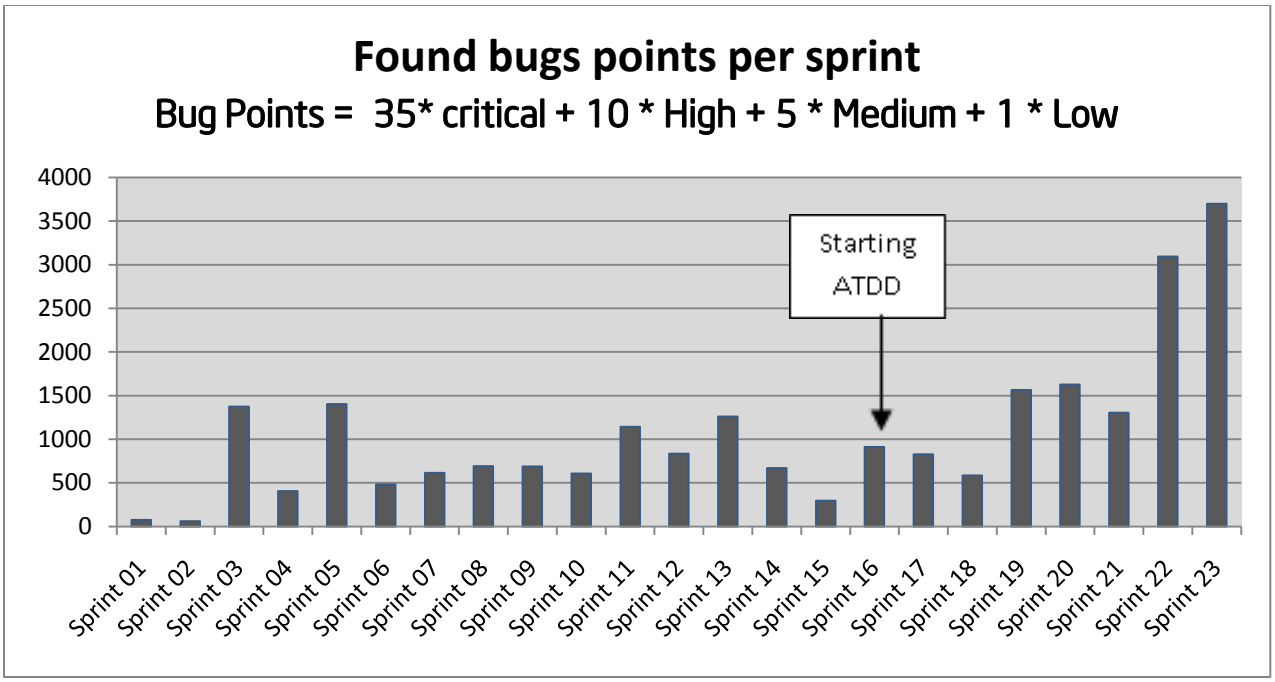

Figure 8: Bug Points Per Sprint

5.5 Quality Improvement

As a result of the improvements gained by including ATDD, and the problems that it solves, the quality of new features is improved. The testers are involved from the design phase, and the whole team is focused on the acceptance criteria.

The ease of running the regression ensures that the project is stable. Stories are completed with validation and bug fixing is all done in one sprint. Bugs are fixed very close to when they were opened. This avoids a technical debt of unresolved bugs and incomplete stories. The pass rate of the full cycles that were run after the acceptance tests passed increased from an average of 71% to 87% (see Figure 9).

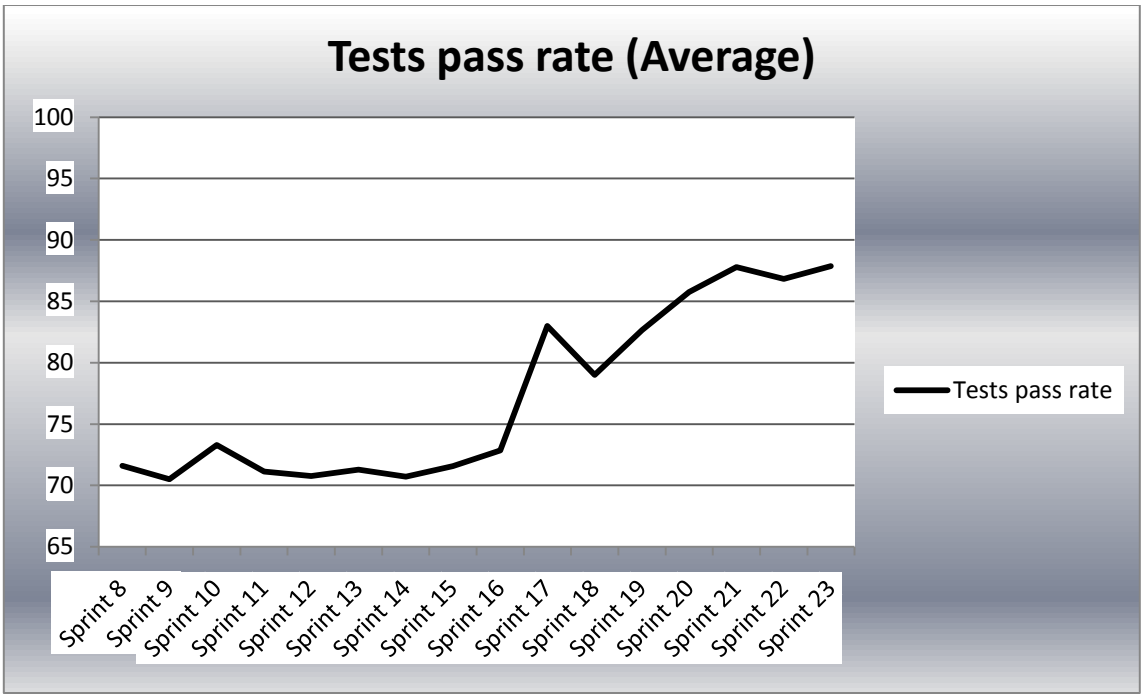

Figure 9: Tests Pass Rate Average

5.6 Team Satisfaction

The feedback that we got from the Scrum team was very good. The validation team members said it was much more fun to work this way. They enjoyed the test development process, and the test cycles became much less frustrating. Being an integral part of the sprint and the fact that bugs were fixed very close to when they were opened also improved their positive feeling. The pressure on validation at the end of the sprint was also reduced.

The main item that returns in each sprint retrospective is “Great communication between developers and validation”. The Scrum team dynamics and satisfaction improved, the information was shared quickly and efficiently, and the whole team was focused on the customer’s requirements.

6. Recommendations

6.1 Use Automation Tools

If you cannot write automated acceptance tests for your product, you cannot use ATDD. Writing automated acceptance tests requires a suitable tool for testing your particular type of product. There are ready made tools for acceptance tests, but we used our own automation tool that was built for our project.

6.2 Use Testers with Developer Skills

The testers must be able to write automated test scenarios for the acceptance criteria.
Use testers with developing skills, or teach them. They need to know at least how to use your automation tool to add the test scenarios. If the testers do not have the necessary developer skills, it will not work.

6.3 Plan for Maintenance

Automated tests need maintenance. Test automation always breaks down at some point. Bugs are found in the automation, and you will need resources to fix the bugs and to improve the structure. Make sure that you include this overhead in your plans.

6.4 Do Not use ATDD for Short Projects

For short projects, if you will not re-use the automation, do not use ATDD. It will take too much time and effort to learn the tool and to maintain the automation.

In our second short project (4 sprints), which was a subset of the first project with some changes in the usage, ATDD did not give us added value. The automation process was not stable until the end of the project. We had planned to use this automation for future projects, so it was worthwhile, but the ATDD process did not work well.

6.5 Do Not Completely Trust your Automation

Automated tests are code, and code can have bugs. Make sure you carefully review the code. Sometimes a critical bug in a project is hidden behind an automation bug.

The automated acceptance tests are good for the main functionality of your project. The aim is breadth, not depth. Use manual tests for the depth tests and let your testers' experience lead them to the important bugs. Also, different testers have different ideas of how to test things. When you trust only automated tests, you lose the new ideas and always repeat the same tests.

7. References

- **http://www.devx.com/codemag/Article/38611/1763/page/1** Introduction to Scrum, Hector J. Correa
- **http://martinfowler.com/bliki/TechnicalDebt.html** Technical Debt, Martin Fowler
- **http://www.methodsandtools.com/archive/archive.php?id=72** Acceptance TDD Explained, Lasse Koskela
- **http://www.51testing.com/ddimg/uploadsoft/20091221/51Testing_salon_41.pdf** ATDD in Practice, Xu Yi (徐毅)
- **http://www.io.com/~wazmo/succpap.htm** Success with Test Automation, Bret Pettichord