Agile Scrum - Retrospectives

Many of us have experienced projects that drag on much longer than expected and cost more than planned. Companies looking to improve their software development processes are now exploring how Agile can help their Enterprise more reliably deliver software quickly, iteratively and with a feature set that hits that mark. While Agile has different "flavors", Scrum is one process for implementing Agile. This newsletter is one in a series of newsletters that will discuss the Agile Scrum process and will end with variants of Scrum that can be used to aid in improving your software releases. Here are the prior newsletters in this series:

- **Feb 2008: Agile Scrum - An Overview**
  http://www.pragmaticsw.com/newsletters/Newsletter_2008_02_SP.htm
- **Mar 2008: Agile Scrum - Team Composition**
  http://www.pragmaticsw.com/newsletters/Newsletter_2008_03_SP.htm
- **Apr 2008: Agile Scrum - Understanding Scrum Rules**
  http://www.pragmaticsw.com/newsletters/Newsletter_2008_04_SP.htm
- **May 2008: Agile Scrum - Scrum Kickoff and Product Backlog**
  http://www.pragmaticsw.com/newsletters/Newsletter_2008_05_SP.htm
- **Jun 2008: Agile Scrum - The 30 day Sprint and the Daily Scrum Meeting**
  http://www.pragmaticsw.com/newsletters/Newsletter_2008_06_SP.htm
- **Jul 2008: Agile Scrum - Reporting and Metrics**
  http://www.pragmaticsw.com/newsletters/Newsletter_2008_07_SP.htm

Overview

Few Agile sprints go exactly as planned. Many sprints encounter problems that must be corrected and some go smoother than planned. Regardless of how successful or disastrous a sprint is, it is important to review the sprint in detail once it is over. This allows your team to figure out what things were done well and to document the things that need improvement. It also aids in building a knowledge base that teams coming behind you can review to ensure they get the most out of their upcoming projects. The key to future successful sprints is to learn from past mistakes. The process of formally reviewing your sprint is called a **Retrospective**.

5 Steps for Conducting Retrospectives

Below are the steps for conducting successful Retrospectives:

1. **Plan Your Retrospective** - Upon completion of a sprint, the team should conduct a Retrospective. This is where the Scrum Master invites all the major players of the team (Product Owner, Team Members, Software Quality Engineers, etc.) to a meeting to review the successes and failures of the sprint.
2. **Require Team Participation** - Ask the attendees to bring a list of 2 items that were done well during the sprint and 2 things that could be improved upon.
3. **Hold the Retrospective Meeting** - Go around the table and have each person to discuss the 4 items they brought to the meeting. Keep track of how many duplicate items you get from each team member. At the end of the round table discussion of items, you should have a count of the most common items that were done well and the most agreed upon items that need improvement. Discuss the top success items and the top items that need improvement.

4. **List Items Done Well and Things Needing Improvement -** Upon listing of the success and improvement items, discuss specific things that can be done to avoid the items that need improvement upon the next release. If some items need more investigation, assign specific individuals to finding solutions.
5. **Create a Retrospective Report** - The best way to keep this information organized is to create a "Retrospective" report, where you document your findings. Send the Retrospective report to all team members. Before team members embark on their next sprint, make sure they review the Retrospective report from the prior project to gain insight from the prior project. We created a template that you can use for the document, download it here: http://www.pragmaticsw.com/Template_Retrospective.doc .

Our Experience with Retrospectives

Agile is an iterative process and if your team makes use of Retrospectives, they will continually improve their Agile process. We have found this to be true in our own organization, below are the results of the first few Retrospectives we conducted:

Sprint 1 Retrospective

Our first sprint did not result in a releasable piece of software because we had too many items that were not fully completed by the end of the sprint. When we conducted our Retrospective, we had these recommendations of items to improve:

1. **Better Requirements** - We started by utilizing User Stories for analysis and found that User Stories did not provide enough detail and caused too much re-work. So we replaced User Stories with a more detailed requirement document called a Work Order, here is an example: http://www.pragmaticsw.com/Template_WorkOrder.doc. We also found that providing estimates (in number of hours) and tracking hours remaining was more efficient, objective and reliable than using Story Points, so we changed our measurements from Story Points to hours.
2. **Test Case Development** - In the first sprint, we had our software quality engineers create test cases for each requirement, but we did not share those test cases with the programmer. Once the code was moved to quality assurance, we found a lot of re-work because many of the test cases failed. So we decided for the next sprint we would have the software quality engineers define all test cases before coding began and we required the programmer review the test cases before coding began. Upon finishing the code, the programmer was required to run all established test cases. We found in Sprint 2 that this simple suggestion reduced quality assurance time by 30% and improved quality.

Sprint 2 Retrospective

By implementing better requirements and by reducing our quality assurance time, our second sprint went much better and we released the new software into Beta. However, our team members had to work too many hours to get the sprint completed, so when we conducted our Retrospective for Sprint 2, we had these recommendations of items to improve:

1. **Better Estimates** - We ran reports that showed that our estimates were about 12% underestimated in Sprint 2. To fix this, we decided to add a 15% estimate buffer to our next sprint and allocated team members to 7 hour days instead of 8 hour days (to account for meetings and planning).
2. **Leadership Oversight** - We reduced the workload of our lead programmer to ensure he had time to provide leadership for code inspections, code refactoring and general team leadership.
3. **Regression Automation** - In Sprint 1 and 2, our regression test cases were run manually and it took about 2.5 full days to fully regress the existing features. Due to

this, we could only afford to do full regression testing about twice during the sprint. To resolve this, we decided to invest in an automated testing tool and to create a set of automated regression test cases that could be run each time we did a new build of the software (daily). You can learn more here: http://www.pragmaticsw.com/WhitePaper_TestCase_Automation.pdf.

Summary

The Retrospectives from the two prior sprints identified critical issues that needed to be addressed and allowed us to finish our third sprint with more functionality than was originally planned and with higher quality than our prior sprints.

What's Next?

Upcoming newsletters will discuss the following topics:

- Agile Scrum - Site specific variants of Scrum

Helpful Templates

Below are some helpful templates to aid you in developing software solutions on-time and on-budget:

- **Pragmatic Agile Development - http://www.pragmaticsw.com/PADOverview.pdf**
- **Software Development /QA Templates - http://www.pragmaticsw.com/Templates.asp**
- **Software Planner - http://www.SoftwarePlanner.com**
- **Agile Training - http://www.PragmaticSW.comServices.asp**

About the Author

Steve Miller is the President of Pragmatic Software (http://www.PragmaticSW.com). With over 23 years of experience, Steve has extensive knowledge in project management, software architecture and test design. Steve publishes a monthly newsletter for companies that design and develop software. You can read other newsletters at http://www.PragmaticSW.com/Newsletters.asp.