Agile Scrum - An Overview

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 Article will discuss the Agile Scrum process and will end with variants of Scrum that can be used to aid in improving your software releases.

So what is Agile?

According to Wikipedia, Agile software development is a conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project. Simply put, Agile allows your team to identify the most critical features of the software that can be completed within a short time frame (normally 1 to 2 months), and it delivers a complete build with this set of limited features as the first iteration. Once that is done, you can move those features to production or continue on to the next iteration. By breaking the releases into shorter stints, it allows you to gain quicker releases and to capture return on investment more quickly by putting the working (but limited) features into production sooner. This is in stark contrast to the more traditional "Waterfall" approach, where you design all features upfront, code each one, test each one, then move into production. Agile projects are iteratively released to production months where Waterfall projects normally span a year or more before they are released to production.

So what is Scrum?

Scrum is process of implementing Agile, where features are delivered in 30 day sprints. Scrum borrows its name from Rugby, where a sprint is the process of stopping play, then vigorously playing until the sprint ends and a new one begins. The same idea applies here, where you define the requirements for a 30 day sprint and work on them with vigor for 30 days without being sidetracked by other things or having things re-prioritized. A specific feature is not recognized as being completed until it is analyzed, designed, coded, tested, re-factored and documented. At the end of the 30 day sprint, most features defined in the 30-day sprint should be completed. If some did not get finished (because of being underestimated), the uncompleted features can be moved to a later sprint. A sprint is considered successful if all the completed features have high quality and can be put into production (or beta) upon ending the sprint.

Do Team Member Responsibilities Change?

Managing Scrum development requires a major change in how teams work together. In traditional Waterfall development, teams normally have a project sponsor, a project manager, analysts, designers, programmers, testers, and documentation specialists. Each team member has specific duties which normally do not overlap and they have a specific reporting structure (most team members report to the project manager). With Scrum,

you have just 3 team roles and is normally limited to 7 or less individuals (however, you can have multiple Scrum teams in sets of 7 or less):

- **Product Owner** - This is the person that identifies and prioritizes the features that will appear in a 30 day sprint. This is normally the CEO, CTO, or some other high level stakeholder that ultimately is responsible for shaping the roadmap of their product.
- **ScrumMaster** - The ScrumMaster is akin to the Project Manager in Waterfall environments, but does not manage the team deliverables at a micro level. Instead, this person is responsible for ensuring that the 30 day sprint stays on course, no new features are added to the sprint, code inspection, and ensuring everyone plays by the rules.
- **The Team** - With Waterfall, a team consists of analysts, designers, testers and documentation specialists. With Scrum, each team member is empowered and expected to self-manage themselves and to participate in all duties needed to deliver a feature. This includes analysis, design, coding, testing and documentation.

So how does Scrum Work on a Day-by-Day Basis?

Scrum begins with an 8 hour **Scrum Kickoff Meeting**. The Scrum Kickoff meeting is divided into (2) 4 hour segments, where you first determine what features are desired for the 30 day sprint. The last 4 hours are used to provide rough estimates for the items identified for the sprint. If the estimates exceed the available resources, the features are prioritized and less important features are dropped from the sprint. An important component of Scrum is using a time-box approach, where meetings and events have a definite time period (e.g. no more than 8 hours for the kickoff meeting) and this time-box is strictly enforced. Once the features are locked in for the 30-day sprint, no changes are allowed (new features can not be introduced until the next sprint). When estimating features for a sprint, the estimates must include time for analysis, design, coding, testing, re-factoring, and documentation. A feature is not considered complete until all those things are done. Each day, a Daily Scrum Meeting is held to determine how the features are progressing. The meeting is no longer than 15 minutes, and each team member is asked 3 questions:

- What have you accomplished since the last Daily Scrum Meeting?
- What will you do before the next Daily Scrum Meeting?
- Is there anything that is impeding your progress (and remedies are discussed)?

From a programmer's perspective, Scrum development is a new paradigm which is very empowering but does require them to follow specific rules:

- Code is only checked out for the duration needed to complete a feature. No exceptions. Most code will be checked in daily, as most features are broken down into small feature sets.

- Time must be entered daily. For each feature, you will have estimated hours, actual hours and hours remaining to complete the feature. This information must be updated at the end of every day so that the ScrumMaster can determine if the release progress is trending as required.
- Programmers are not allowed to be pulled off on tangent projects, they must stick to the features they have been assigned for the sprint.
- All team members must attend the Daily Scrum Meeting and must be on time.
- Code is compiled and deployed to a test server daily. Teams can use automated build tools to speed this process. Automated tests should be run against the daily releases to discover any issues introduced by the release.

Once a Scrum 30 day sprint is completed, all features that were completed can then been moved to a beta or production environment. Following the sprint is a Retrospective (post mortem), where team members discuss and document things that went well and things that can be improved upon in the next sprint.

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. Steve's email is steve.miller@PragmaticSW.com.