

Fairway Technologies

Scrum Overview

Date: 12/30/2009

Author: Noah Heldman

Scrum Overview and Roles

Scrum is an agile methodology that relies on a simple process to get complex work done. It is far less process-intensive than a traditional waterfall approach, but still provides the necessary controls and outputs to guarantee high visibility into the past, current, and future of a project. However, in order for Scrum to be successful, it requires a dedicated team that is willing to follow the well-established guidelines of Scrum.

Why Do We Need It?

Have you ever worked on a project like this?

- ⇒ Business Analysts define requirements along with a few key stakeholders
- ⇒ Architects analyze the requirements and create a design
- ⇒ Developers code to the requirements
- ⇒ QA tests the code
- ⇒ Operations deploys the code

Was it on time, on budget, and did it meet the needs of its customers? Most likely not, because it probably looked more like this:

- ⇒ Business Analysts define requirements along with a few key stakeholders
- ⇒ *The requirements change*
- ⇒ Architects analyze the requirements and create a design
- ⇒ *The requirements change*
- ⇒ Developers code to the requirements
- ⇒ *The requirements change*
- ⇒ QA tests the code
- ⇒ *The requirements change*
- ⇒ Operations deploys the code, *and the stakeholders see the result for the first time*

On paper, it is clear why the waterfall approach is prone to failure, but that doesn't stop thousands of organizations from implementing it every single day.

Scrum aims to greatly increase the involvement of the stakeholders, and ensure that the current status of the project is visible to all interested parties at all times. In addition, stakeholders get to see completed functionality (with business value!) and reorganize project priorities every thirty days.

Conceptually, you may think "that sounds great...sign me up!", but success with Scrum requires structure and dedicated team members.

An Analogy (stolen)

“Suppose I’m traveling from Chicago to Boston by airplane. Before and during the flight, the pilot gets instructions from air traffic control. We take off on command and follow the prescribed route. Once we are in the air, computers predict almost to the minute when we will land in Boston. If things change—say, the air is bumpy—the pilot must get permission to move to a different altitude. As we approach the airport, the pilot is told what runway to land on and what gate to go to.

“If, however, I set out for Boston in a car, I can take whatever route I want, whenever I want. I don’t know exactly when I’ll get there, and I probably haven’t planned what route I’ll take or where I’ll stop for the night. En route, I follow traffic laws and conventions: I stop at red lights, merge into traffic according to the prevailing customs, and keep my speed consistent with the flow. In an automobile, I am an independent agent, making decisions in my own best interests framed by the rules of the game of driving.

“It’s amazing to me that thousands upon thousands of people travel by car every day, accomplishing their goals in a framework of simple traffic rules, with no central control or dispatching service. It also amazes me that when I want to ship a package, I can enter a pickup request on the shipper’s Web site and a driver will arrive at my door before the time that I specify. The driver isn’t dispatched to each house; he or she receives a continually updated list of addresses and deadlines. It’s the driver’s job to plot a route to get all the packages picked up on time.”

Empirical Process Control

Every aspect of a McDonald’s has been defined and perfected to operate smoothly within the bounds of what is normal for a restaurant. You know exactly what to expect, whether you go to a McDonald’s in San Diego or New York. The same is true for Starbucks. This can be comforting, as humans are creatures of habit, and it is beneficial for the businesses, because they can predict effectively, and maximize their profits. This is called *defined process control*.

Software projects are a different animal. No two are alike, and any effort to nail down every aspect of a software project (as one would a chain restaurant), and repeat it, is unlikely to succeed. In fact, the more controls one tries to put on a very complex project, the less likely it is to move forward. Just think of the bureaucracy and budget overruns typically associated with large government projects.

Scrum favors *empirical process control*, which simply admits that we can’t possibly understand all of the interactions in a new, complex system, so we do three things to improve our chances for success:

1. **Visibility:** Ensure everything important about the project is visible to those responsible for its success
2. **Inspection:** Have all interested parties look frequently at the project and its outputs, to accurately identify where you are in relation to your goal
3. **Adaptation:** Make changes to anything that doesn’t look right from #2!

Chickens and Pigs

A chicken and a pig are walking down the road. The chicken says to the pig, "Do you want to open a restaurant with me?" The pig considers the question and replies, "Yes, I'd like that. What do you want to call it?" The chicken says "Ham and Eggs!" The pig says, "On second thought, I'll pass. I'd be committed, but you'd only be involved."

Roles

There are only three roles in Scrum (these are the Pigs). Everyone else involved in the project is a Chicken.

ScrumMaster

The ScrumMaster leads Scrum projects by providing guidance and coaching, removing roadblocks, navigating new complexities, and ensuring the team follows the Scrum process as closely as possible. The ScrumMaster is akin to a project manager in a traditional project, but with a few very important distinctions.

Product Owner

Represents the interest of all stakeholders, and establishes the product backlog, which is the prioritized list of the initial overall requirements. Defines ROI objectives, and release schedules.

Team

Teams are self-managing, self-organizing, cross-functional, and are responsible for turning Product Backlog items into completed business functionality within a sprint *in whatever way they see fit*.

Scrum Rules

Scrum has a very simple set of rules. Each item below is “time boxed”, meaning that the ScrumMaster should never allow the activity to run longer than the allotted time.

Day 1: Sprint Planning Meeting (8 Hours)

First 4 hours: ScrumMaster, Product Owner, and Team

The Product Owner prepares the Product Backlog (prior to the meeting), which is a prioritized list of requirements for the project overall. The Team selects the items they believe they can complete in the first sprint.

Second 4 hours: Team, ScrumMaster

The Team breaks the Product Backlog into tasks, and estimates the work.

Every Day: Daily Scrum Meeting (15 Minutes)

The Daily Scrum is a daily meeting guided by the ScrumMaster, where the Team answers three questions:

1. What did you do yesterday?
2. What will you do today?
3. Are there are any roadblocks preventing you from getting your work done?

Days 2-31: Sprint (30 Days)

The Team manages itself to get its work done, and can ask questions of the ScrumMaster and Product Owner as necessary.

The goal is to create a complete increment of business functionality; analyze, design, code, test, and document, and deploy to a server where it is available for demonstration.

Spikes can help to try out a new technology, to determine whether it is viable for use within the current sprint.

Sometimes, it is necessary to do more work on infrastructure items (server setup, security, complex architecture decisions) in early sprints, but each sprint *must deliver business functionality*.

Day 32: Sprint Review Meeting (4 Hours)

The Team demos the completed functionality to the Product Owner and any Chickens who want to attend. If any requirements or priorities have changed, the Product Owner can reorganize the Product Backlog.

Day 32: Sprint Retrospective Meeting (3 Hours)

The Team, ScrumMaster, and (optionally) the Product Owner talk about what went well, and what didn't, so adjustments can be made to help the next sprint run more smoothly.

Credits

Just about everything in this document was completely stolen from the book *Agile Project Management with Scrum* by Ken Schwaber.