Introduction to Scrum

Recorded by Michael James

[Existing slide with MJ]
Welcome to Module 1 of CollabNet's Scrum Training Series: Introduction to Scrum. This is a brief introduction to topics that are covered in greater depth in the other modules. I'm Michael James. I help organizations do Scrum and related Agile practices.

[new slide: Thumbnail of Scrum Reference Card, clicks through to online document]
This module is subdivided into three chapters. At the end you'll find a challenging quiz that's been shown to increase scores on certification tests and may be a prerequisite to your certification class. I also recommend downloading the six-page illustrated Scrum Reference Card.

[existing slide: Scrum Framework]
Scrum is a framework for dealing with complex work such as new product development. It is an alternative to traditional approaches that were more suited to manufacturing and construction. Why do you need an alternative?

[existing slide: "OLD" assembly line]
[cue scratchy music from the 1940s or 50s]
In the previous century, most people's work required more of a focus on execution than innovation. People with defined roles used defined models and defined best practices to execute defined plans that didn't change very quickly.

[new slide: "OLD" assembly line is superimposed on the lower left quadrant of the Stacey diagram, in the "Predictable" zone.]
Life was more predictable because we knew more about *what*

[slide or animation emphasizes "Requirements"]
we were trying to accomplish, and *how*

[slide or animation emphasizes "Technology"]
we would accomplish it. People like Henry Ford used ideas from Frederick Taylor and H.L Gantt to do predictable or repeatable work.

[zoom out to existing slide: Stacey diagram]
[scratchy old music stops, or changes to modern music, whatever that is]
Today, a lot of the predictable, repeatable work is done much faster, by machines. Things change more quickly. To stay competitive, we need to inspect and adapt more quickly, and deal with greater uncertainty. A transparent framework imposing timeboxes and feedback loops can help us master uncertainty.

[new slide: “Only *learning organizations* will be able to keep up with the future.”]
Only *learning organizations* will be able to keep up with the future.

[return to existing slide: Stacey diagram]

[new slide: Stacey diagram with framework image in the chaotic zone]
Scrum is a framework for learning about work and the processes we use to do it. It’s an attempt to put chaos in a box, making the most of uncertainty. While it’s mostly been used for developing new software products, it may be useful for other kinds of complex work.

[existing slide: blank circuit diagram]
Scrum introduces feedback loops

[existing slide: colored circuit diagram]
encouraging us to inspect and adapt the

[new slide: circuit + product measuring instrument only]
product that we’re building and the

[existing slide: circuit + both instruments]
processes we’re using to build that product.

[existing slide: balanced Agile Movement]
Scrum is associated with the Agile movement described at
http://agilemanifesto.org/. We value:
• • • •
individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.

[existing slide: Agile Movement tips left]
This doesn’t mean that we don’t value the things on the right.

[optional animation: balance teeters]
We do value the things on the left more.

[new slide: Running Tested Features diagram]
The basic intent of all the Agile approaches (and they are quite similar), is an early and sustainable rate of valuable feature delivery. Early in our product development cycle, we will want to see some features working, rather than one huge release at the end. Ideally,

we'd be able to continue incremental improvements indefinitely. Product development is ongoing, with frequent releases, rather than a one-shot "project."

[old scratchy music comes back]
[existing slide: waterfall diagram]
Here's what Scrum is NOT: The attempt to use traditional Gantt ideas for developing software was first described by Dr. Winston Royce in 1970. He drew a picture, a bit like this one, showing a series of phases.

[cycle through existing slide sequence showing relay-race baton handoffs]
One phase connects to the next phase connects to the next phase and each one has a hand off. In theory, you would do all of your analysis in the beginning, get that completely done, and then you would start on your design, get that completely done, and then start on your code, get that all the way done, and then integrate, etc. That was a nice theory about how things could work, and we suppose it could work if we had perfect knowledge in the beginning, and never made mistakes along the way.

The next sentence in Dr Winston Royce's paper (unfortunately, no one read this far), said

[slide: Dr. Royce in the barrel falling down the waterfall]
"I believe in this concept but, the implementation described above is risky and invites failure."

[slide sequence: blindfolded team members fail to hand off batons to each other]
The reason it fails for complex work: We rarely have perfect knowledge at the beginning. In fact, we know less about our project when we are starting out then we're ever going to know in the future. Today is the dumbest day of the rest of our project. But a plan-driven approach requires us to make our most important decisions right at the beginning, when we know the least. Waterfall projects eventually descend into anarchy, when the actual work finally happens with low quality and too much overtime.

[existing slide: ScrumMaster throws waterfall into blender]
Scrum throws all those phases in the blender. All the ingredients are mixed into every Sprint.

[existing slide: ScrumMaster with shot glasses]
Instead of dividing the work into activity-specific phases, we divide it into fixed-length iterations, called Sprints. Every Sprint contains some combination of analysis, design, actual implementation, testing, planning for the future, maybe we'll actually be deploying or shipping more frequently.

[optional future animation: shot glasses turn into rainbow circles, then add the loops on top making Scrum swirls]

[existing slide: Scrum swirl + unicycle, bicycle, tandem bicycle]
Right from the first Sprint, a Scrum team tries to build a *working, tested, potentially-shippable* product increment, even if it starts out small. Every Sprint they demonstrate what little bit of shippable product they have to everyone. Customers often need to see the wrong product before they can specify what they really need. With short iterations and more feedback we have a better chance of discovering the *right* product. While the Product Owner doesn't have to actually ship every Sprint, it's the team's job to make it possible.

Sprints are officially 30 days long and most Scrum teams are doing them in less, such as two weeks.

[existing slide: Scrum swirl, etc. + cross-functional team]

To make a shippable product every two weeks, we're going to want people with skills in testing, in design, in business requirements, in coding, all on one *cross-functional, self-organizing team*.

Now, instead of waiting for the end to start our testing, we'll start testing with our very first Sprint. Instead of doing all the design up front, we'll do a bit of design every Sprint, until it becomes *continuous design.* We'll do small amounts of continuous redesign and refactoring to avoid *technical debt*. Every Sprint combines all aspects of the work. A Scrum team should collaborate together instead of working in phases with handoffs.

[existing slide: Scrum is hard]
Scrum brings challenges to individuals, teams, and organizations. If it's not disrupting your organization, you're probably not doing Scrum. That's the case most of the time people claim to be doing Scrum. You usually find that they've overlooked the parts of Scrum that would provide the most benefit because they're too hard to do to. The purpose of Scrum's strict rules are to reveal organizational impediments that you can start fixing, if you have the courage and commitment. Healing the organizational impediments exposed by trying to do Scrum is more important than "doing Scrum." But this path to mastery starts with learning the framework properly.

[existing slide: Roles, Meetings, Artifacts]

Scrum provides a structure of roles, meetings, rules and artifacts. Let's do a quick overview of them now.

[existing slide: 3 roles]
There are only 3 roles defined by Scrum and we'll go through each one: Product Owner, Scrum Development Team, and ScrumMaster.

[existing slide: Product Owner]
The Product Owner is the single individual responsible for return on investment (or *ROI*) of the product development effort.

The Product Owner mostly exerts that influence through the prioritization of the product backlog. The Product Owner is the final arbiter of requirements questions. That doesn't mean that he'll give you all of the detailed requirements up front, or even all the details at the beginning of each Sprint. But the Product Owner does make the final call about those things.

[new slide: Product Owner has vision. For example, a thought bubble with a light bulb in it.]
The Product Owner must have the vision behind the product development. Given high uncertainty, it often doesn't make sense to have a detailed roadmap, but a vision is important.

[new slide: stakeholders funnel concerns through the Product Owner]
If anyone else wants anything from the team, they need to work through the Product Owner to get what they want. Sometimes we call the Product Owner "the single, wringable neck." He's the one person making the prioritization decisions for that team. He makes the business decisions for that team, focused more on the *what* than on the *how*.

[existing slide: Scrum Development Team]

Now we have the Scrum Development Team.

The Scrum Development Team is a cross functional group responsible for self organizing to build a "potentially shippable product increment" every Sprint. This is hard to do in the beginning, and today more and more teams are learning how to do it. You don't want to be the *last* team that learns how to do this!

Organizations use the word "team" lightly. Think back to a time in your life you were on a *real* team, where you could all count on each other. That's the kind of Scrum team I want you to create. If we keep the same team together in the right environment full time, with effective Retrospectives, Sprint after Sprint, they'll probably get better at working with each other. They may become a *learning team*, the building block of a *learning organization*.

There are no externally imposed hierarchy or job titles on this team. We leave openings for leadership to emerge naturally, and for control to flow from person to person.

[new slide: derive from small_vs_large_team_illustration]
The Scrum Development Team is a small group of people, ideally between 4 and 9 people. We have millions of years of practice dealing with groups about the size of a family. Large teams don't self organize effectively until they divide into smaller teams, ideally *feature teams* with minimal interdependencies.

[existing slide: Team Room]
Team collaboration emerges most naturally in a team room.

[existing slide: planet-sized impediment]
If your organization is spread around the world, you're probably already suffering poor collaboration and coordination; Scrum will quickly bring this problem to the surface. You may experience breakthroughs by getting your remote people into one room for one or two Sprints in the beginning, and then every few months after that. Information sharing tools help with the practical problem of sharing information, but tools by themselves don't transform organizations.

[existing slide: ScrumMaster]
The ScrumMaster is the most misunderstood role in Scrum. If I'm your ScrumMaster, you're not my ScrumSlave. Its the opposite. The ScrumMaster has no management authority over the team. If you are the project manager or line manager of the team, by definition you are not the ScrumMaster. (Sometimes it turns out you're actually a Product Owner.) Scrum intentionally leaves out the project manager role.

[existing slide: ScrumMaster + PO + Team]

The responsibilities of project management are split up among the Product Owner and the Team, with the ScrumMaster acting as a kind of facilitator.

[optional new slide: ScrumMaster protects the team from wolves.]

[or return to ScrumMaster slide]

The ScrumMaster protects the team from distractions and interruptions, gets things out of the way of natural team self-organization, removes impediments affecting the team, facilitates the process, helps teach people how to use Scrum, promotes improved engineering practices, enforces timeboxes, provides visibility, and somehow does all this without any management power. How is that possible? It turns out “power” isn’t even the most powerful type of influence.

[new slide: illustrate the ScrumMaster checklist]
For further information on the elusive ScrumMaster role, see the “Example ScrumMaster’s Checklist” , an example list of things the ScrumMaster would be concerned with fixing.

[existing slide: artifacts]
The two important artifacts in Scrum are the Product Backlog and the Sprint Backlog.

[existing slide: Product Backlog]
The Product Backlog is a one-dimensional, force ranked list of customer-centric features, prioritized by the Product Owner. It’s a list of everything we might ever do. If it’s not in the backlog, it doesn’t exist. Anyone can add items to the Product Backlog, but the Product Owner has got to prioritize them, and the ScrumMaster’s got to make it visible.

A well formed Product Backlog does not contain tasks, only well-formed Product Backlog Items (or PBIs) which might be written in user story form, or maybe use case scenarios.

[existing slide: Force Ranked]
Force ranked means there is only one thing in the top position. Getting organizations to do this is usually a breakthrough.

[existing slide: Sprint Backlog]

The Sprint Backlog is what we’re planning to do right now to meet our current Sprint Goal. It has an end date. It has a subset of Product Backlog Items selected for the Sprint, and a plan for how to do them, such as a list of Sprint Tasks.

[existing slide: Meetings]

Now, let’s have an overview of the meetings in Scrum.

[existing slide: Meetings flowchart]
There are four meetings defined by Scrum and a fifth one that just about everyone has found useful to do. The four meetings defined by Scrum:
The Sprint Planning Meeting, The Daily Scrum, The Sprint Review Meeting, and the Sprint Retrospective Meeting

The fifth one has no official name so we’ll call it the Backlog Refinement Meeting.

[calendar]
Here’s an example of how those meetings might fit into a two-week Sprint. I personally like to end the Sprint on a Friday so we don’t work over the weekend.

[existing slide: Sprint Planning Meeting]
At the Planning Meeting, the Team and the Product Owner choose which items to attempt in the sprint.

[existing slide: Sprint Planning Meeting with Sprint Backlog]
The team pulls selected items into the Sprint Backlog, plans how they will do them, and decides whether it’s the right amount of work for them to do. They plan one Sprint.

[existing slide: Daily Scrum]

[@VJ: add 15-minute timer]
During Sprint Execution the Team meets once per day for a 15 minute standup, or Daily Scrum. If we’re collaborating we’re meeting all the time of course, but this one official meeting is defined where we stand up and report to each other. We report to each other. Not to the Scrum Master, not to the Product Owner, not to any kind of boss but to the Team. I report to the six other people that are my team members. I look them in the eye and I tell them what I did yesterday (“here’s what I did yesterday”), and I tell them what I’m going to do today, and I tell them what impedes me, what are my blockers. Then I toss the ball to someone else on my team and they give the same report to the rest of the team.

[existing slide: Sprint Review Meeting]

At the Sprint Review Meeting, the team demonstrates a potentially shippable product increment to the Product Owner and anyone else who’s interested (sometimes called stakeholders). The Product Owner will declare which items are done, which items did not meet their acceptance criteria. We can measure velocity and we'll get feedback from stakeholders about how we're doing with the product.

A lot of times the feedback is "Hey guys you did what you said you'd do, but now that we see it we realize that we need something else. And we couldn't have known that until we saw the *wrong* product." People seem to need a functioning piece of software to react against before they can specify what they really want.

[Repeat existing slide: Product feedback loop]
Regular Sprint Review Meetings provide feedback about the product and its emerging requirements.

[new slide: Sprint Retrospective Meeting]
Every Sprint ends with a Sprint Retrospective Meeting for the team to inspect and adapt their own process. We inspect and adapt the way we worked together during the last iteration.
We typically talk about what went well, what could be improved, what we learned and what still puzzles us. We give feedback to each other. These kinds of things will come up in the Retrospective. Some practical techniques for doing this well are covered in Module 6. This is really the key of the whole thing: the Team eventually takes ownership of their own process.

[repeat existing slide: feedback loops with instrument measuring "Process"]
Regular Sprint Retrospectives provide feedback about the *process* the team uses to build the product. Remember: Scrum is a framework for learning about products and the processes we use to build them.

[new slide: Backlog Refinement Meeting]
The fifth meeting isn't given a name officially in Scrum. I'm going to call it the "Backlog Refinement Meeting." It might also be called Backlog Grooming, or maybe we'd do some of this work in a Release Planning Meeting.

During Backlog Refinement, the Team and the Product Owner get together and they look ahead a little bit into the next few items in the Product Backlog, the items that are candidates for the next couple Sprints. They clarify them, break the big Product Backlog Items (or *epics*) into small Product Backlog Items (for example: *user stories*) so they could imagine doing a few of them in a Sprint, get some input about prioritization, consider dependencies, etc.

While this work could also be done in the Sprint Planning Meeting, through experience, people have found it preferable to do it in a separate meeting on a different day.

[new slide: conclusion]
That was a brief overview of Scrum.

To learn a little more about Scrum, download the Scrum Reference Card and the ScrumMaster's Checklist. To learn a lot more about Scrum, attend one of our Scrum classes, or try our other training modules that go into more depth. To get coaching in Scrum, or tools such as ScrumWorks, TeamForge, or Subversion, give us a call here at CollabNet. Also drop us a line to give feedback on this training module.