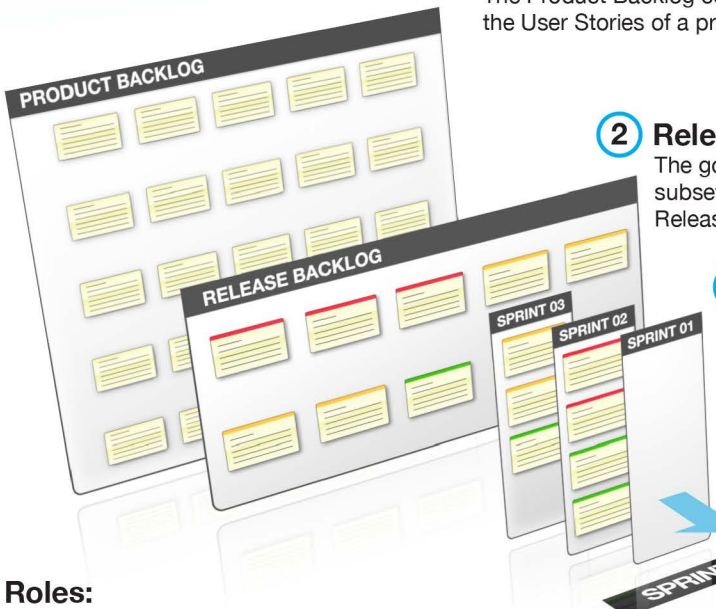
axosoft

Scrum Diagram:

1 Product Backlog

The Product Backlog contains a wish list of all the User Stories of a product.

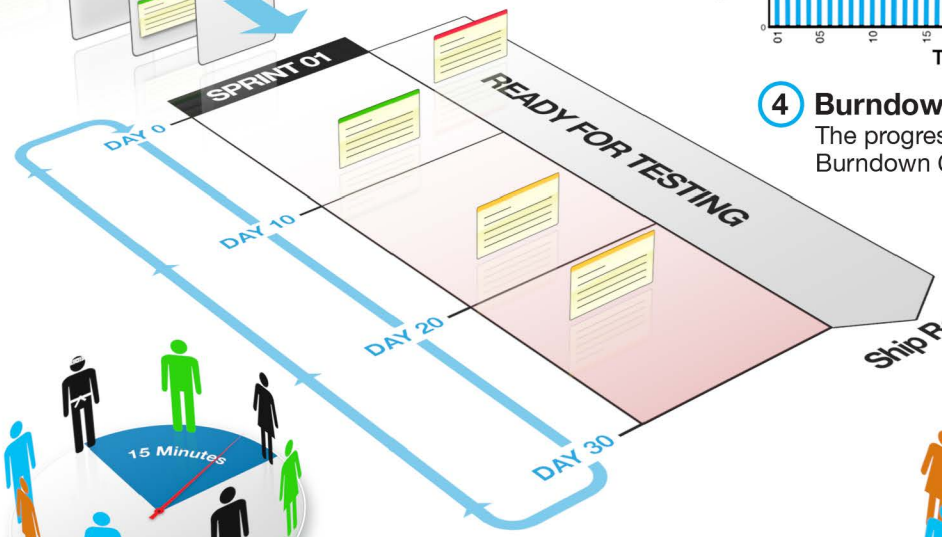


2 Release Backlog

The goal of a given release is to deliver a subset of the Product Backlog, known as the Release Backlog.

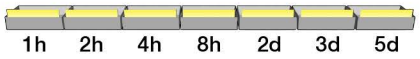
3 Sprint Backlogs

Each Sprint (or short duration milestone) takes a small chunk of the Release Backlog and gets it Ship-Ready!

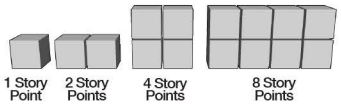


Estimation Techniques:

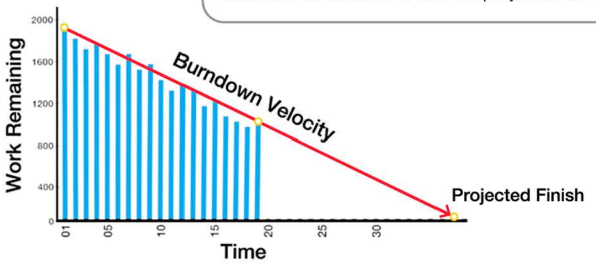
Hours: When estimating work, it's important to have the entire team use some standards. Use 1h, 2h, 4h, 8h, etc. No estimations in-between.



Story Points: You can also estimate work in comparison to the complexity of a well known but simple component.



Remember that in Scrum, the estimates are only part of the story. The Burndown Velocity is the true indicator of whether or not the project is on track.



4 Burndown Chart

The progress of the team is monitored using a Burndown Chart.

Team Roles:

Product Owner: Is responsible for what goes into the product backlog and prioritizes it. Would probably make a good dictator if given the chance.

Scrum Master: A team facilitator. Ensures teams have what they need to get the job done. Also, sets up meetings and monitors everything. Also, kicks ass when necessary.

Developers & Testers: They write code and make sure it does what it's suppose to do. Duh!

5 Daily Scrum Meetings

Short daily standup meetings ensure everything is on track and everyone has the tools they need.

6 Sprint Retrospective

After each sprint, a longer retrospective meeting helps fine-tune the process.

For more info:

Intro to Scrum Video: www.axosoft.com/scrumvideo

http://agileanarchy.tumblr.com/post/20795405232/top-5-scrum-workshop-donts

Top 5 Scrum Workshop Dont's

There is a great framework for teaching people about Scrum. It is called Scrum. Oddly, many Scrum trainers and facilitators forget about it, creating very incongruous experiences for the participants. We still get caught up in a training mindset, believing the trainer to be the expert, and expecting the group to hang on to our every word. But teaching Scrum is not training, it is exploration [1]. There are many good techniques for keeping workshops open, dynamic, Scrum-infected, and engaging. There are also many techniques that should be avoided. I may write something about the former in a future post, but today I am focusing on what *not* to do.
And yes, I know it is usually unwise to focus on negatives [2], but sometimes one just needs to clean out the crap before our minds are able to take on new ways of thinking and behaving. Smart readers will turn these statements around and draw out the positive. It's there, hiding :)

1. Don't plan the whole workshop.
Big upfront planning is anathema to Scrum. We seek lightweight, flexible visions and architectures that can change on the fly according to market changes or technical innovations/restrictions. Teaching Scrum from a plan is not teaching Scrum at all, it is playing safe, and it is imprisoning yourself and your group to an imperfect outcome. Instead, have a workshop goal, and a lightweight framework. Run workshops in iterations, allowing for reflection and replanning between each. As the workshop progresses, let the participants drive the direction. Allow for the workshop goal to change.

2. Don't ask questions you already know the answer to.
This technique is widely used. It is both fear-driven and patronizing. Fear-driven, because the facilitator is always safe, never taking any risks, always knows the correct answer. Patronizing because it treats people like children, vying to get the gold star in class. Avoid putting your participants in a teacher-pleasing position. Only ask a question if the answer will help inform you, as a facilitator, to better understand the context you are working in.

3. Don't play games with predefined outcomes.
For one, you'll get bored pretty quickly after a few repeats of these. And your mind will become closed to possibilities. Play open-ended games, where the process of the game itself—different every time—will inform the participants. Catch learning moments as they happen, and allow time to explore those. Enter each game as if it is entirely new to you. Because it is. Scrum is about people, not repeatable process, and no two people, not two groups will ever behave the same under the same guiding principles, or rules. Invent variations on games as you go, take risks, be prepared to fail… and use that itself as a learning moment for all. Especially, don't play predefined outcome games you yourself have not experienced as a player. This is disingenuous, and you will only be able to facilitate with an academic mindset. Playing open-ended games you have not played yourself is fine, as you are a player and risk taker in this process.

4. Don't answer questions.
Especially don't answer "How" questions [3]. These are usually asked as challenges, and attempting to answer them will likely lead to positioning and entrenchment. Instead of offering answers, whether from intuition or experience, seek instead a better question—there is almost always one or more of those lurking below the surface. A classic example is the "How do we manage bugs in Scrum?" question. This is not a useful question, a better one being "Why do we have bugs?" [4].

5. Don't talk about yourself.
Frankly, no one cares. Spend the time finding out about the group instead. Ask challenging questions. Seek to understand, rather than to be understood. Keep case studies and stories from experience to a minimum. These moments are more about your own ego than any real learning. Teach from a principles and values perspective, not a solutions perspective. Ask about the experiences of the group members, and use those as pivots for exploration.

1: http://agileanarchy.tumblr.com/post/20697179226/thoughts-on-training
2: http://en.wikipedia.org/wiki/Positive_and_negative_%28NLP%29
3: http://www.amazon.com/The-Answer-How-Is-Yes/dp/1576751686
4: http://www.agilelearninglabs.com/2012/04/no-more-bugs-an-excerpt-from-jeff-mckennas-book-conscious-software-development/