Crazy Little Thing Called Scrum

IT'S HAKA TIME!

[STANDARD INTRO STUFF]

Did anyone think this presentation was going to be about rugby?

How many of you work in a scrum environment?

Anyone worked in a waterfall environment?

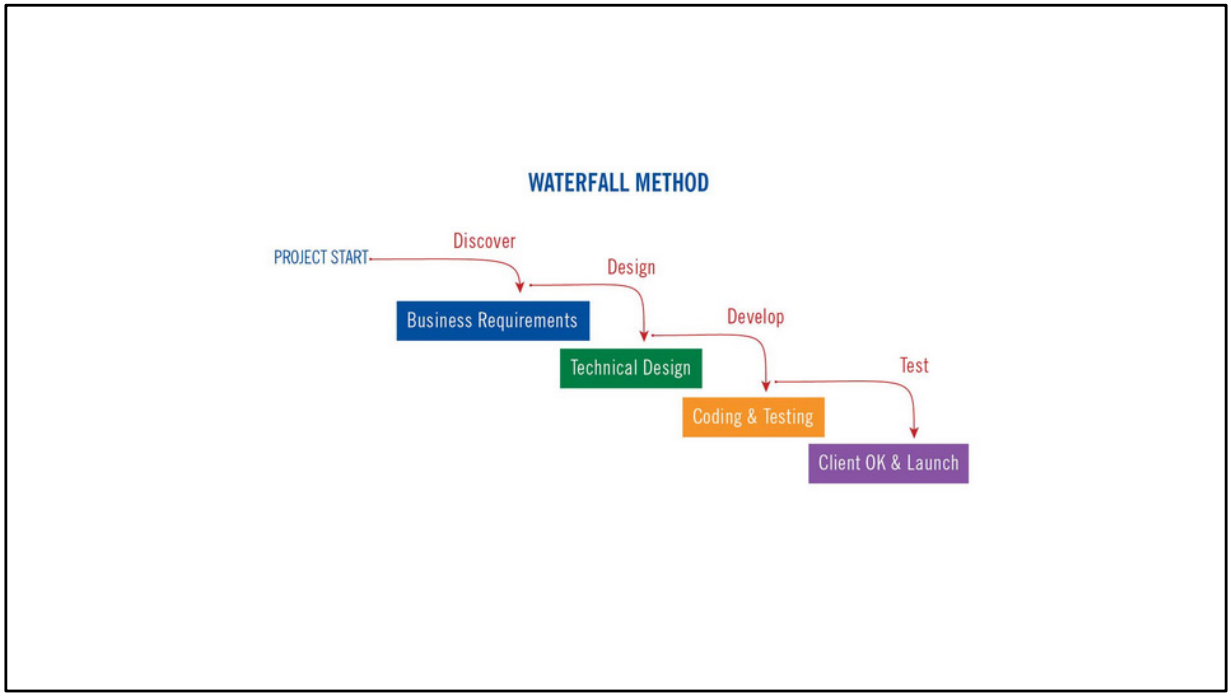
Waterfall was broken.

Thousands of pages of inaccurate documentation
Months of requirements gathering
Years of development

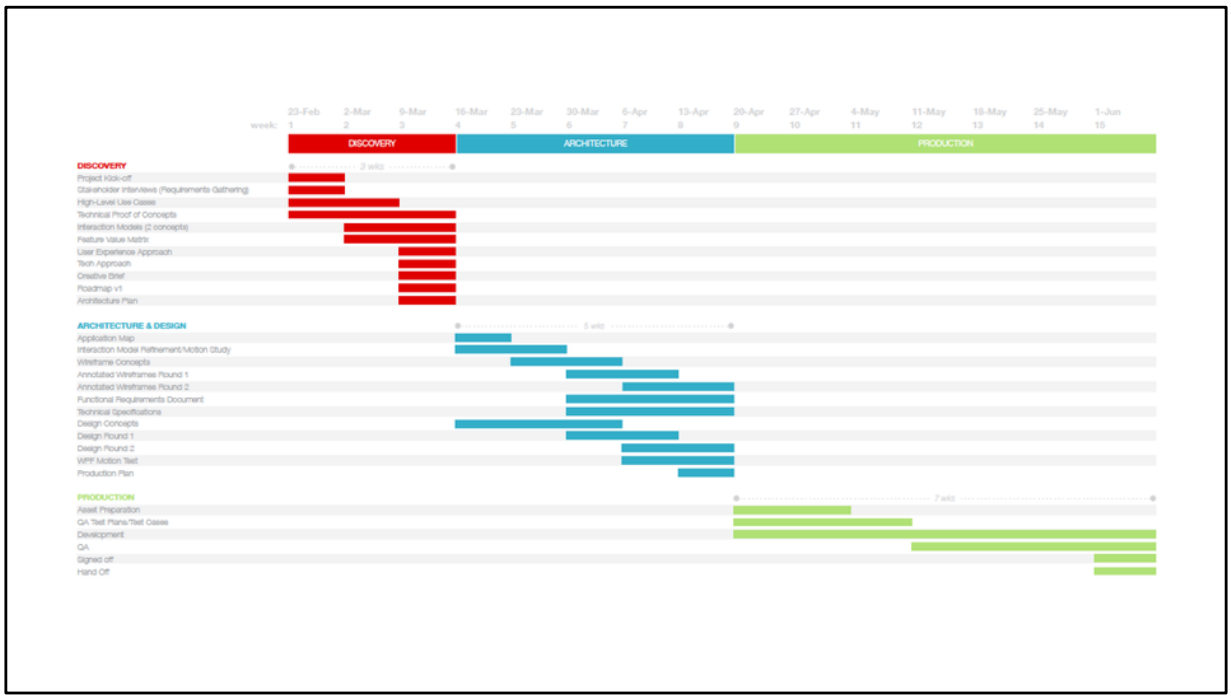
All before delivering any form of a workable product

Waterfall is illustrated by these things called Gantt charts, which are a literal way of illustrating throwing things over a wall.

They look a little something like this…



In reality they looked more like this, but like 4 pages longer.

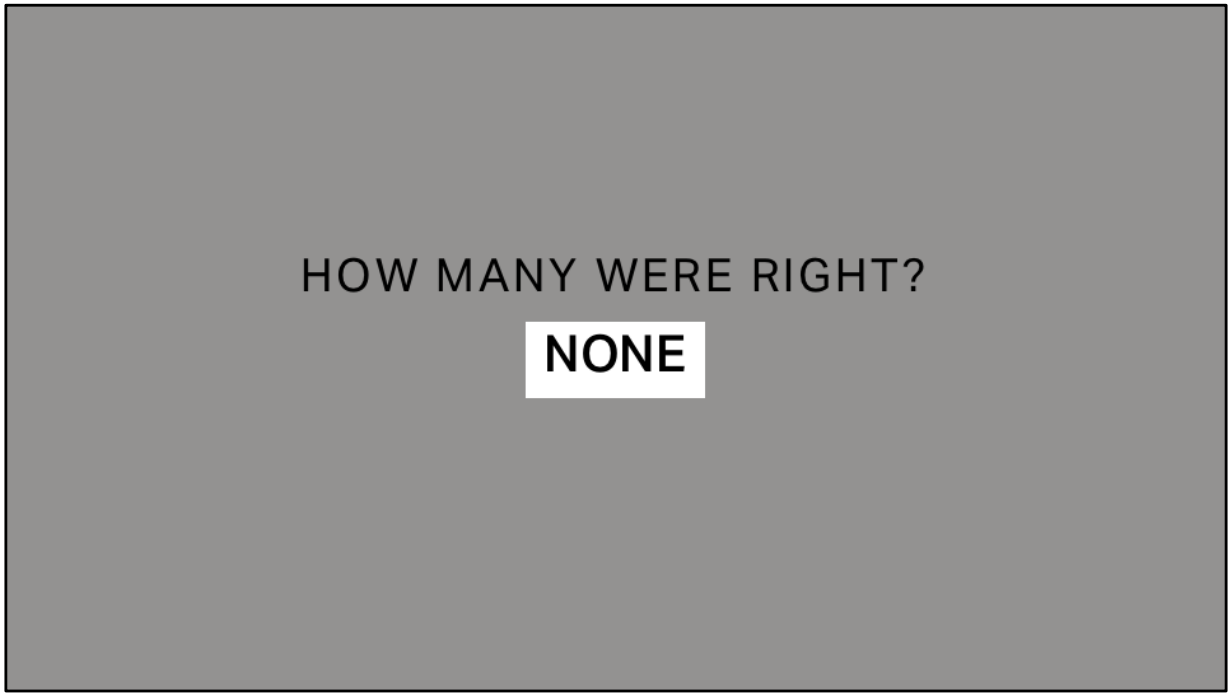


[Tell story about Sustainability Management Program]

Jeff Sutherland was talking to a CEO, whose major project was behind, by a long way. He asked him two simple questions…. (next slide)

HOW MANY GAANT CHARTS
HAVE YOU SEEN?

HUNDREDS

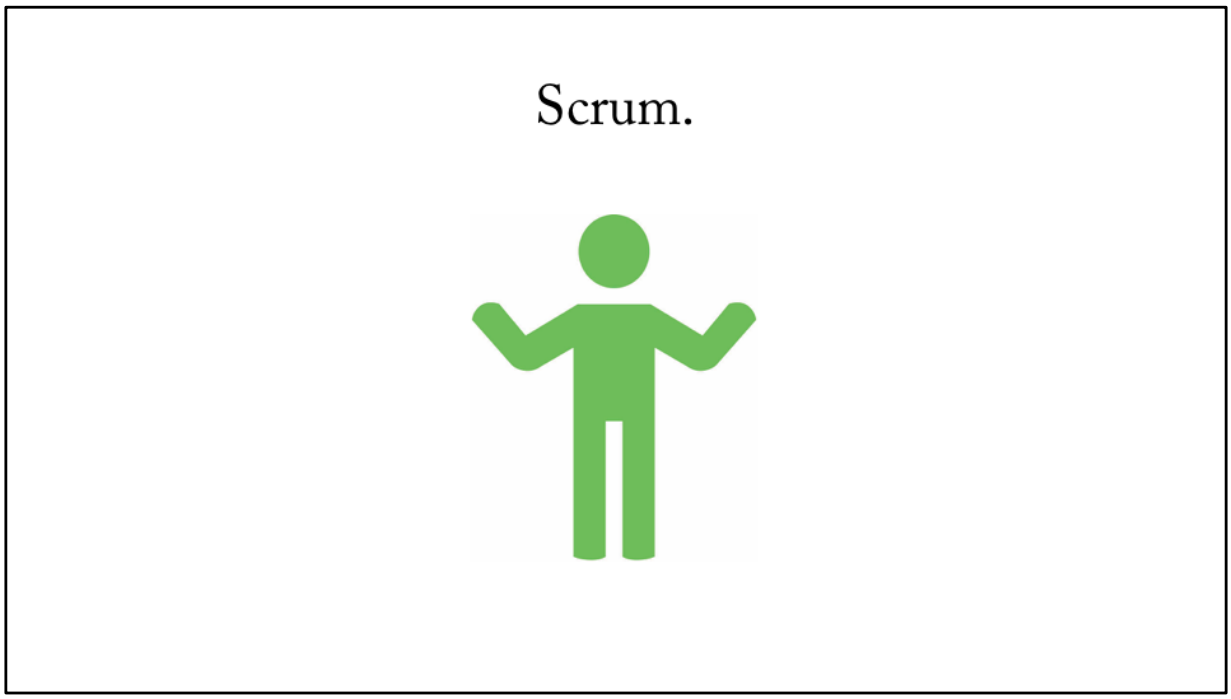


So let’s change the process…

Scrum!

Which brings us to modern day and scrum.

Most people's introduction intro scrum looks a little something like this…



So let's dive deeper…

Back in the mid-80's Harvard Business Review published a paper by two Japanese Business Professors: Hiotaka Takeuchi and Ikujiro Nonaka. It was titled "The New New Product Development Game." They studied places like Honda, Fuji-Xerox, 3M and others. They argued that waterfall product development was the wrong approach. That these "best teams" used an overlapping development process that was faster and more flexible. Teams were cross functional. They had autonomy. They had a transcendent purpose. The professors compared the teams' work to that of a rugby team. The best teams acted as if they were in a rugby scrum, and it refers to the way a team works together to move the ball down the field. Careful alignment, unity of purpose, and clarity of goal come together.

7 years rolls by and eventually Sutherland gets this idea to take theses principles and apply them to software development, and thus modern day scrum is born.

Basically, scrum is just a framework for the most important thing in software development…

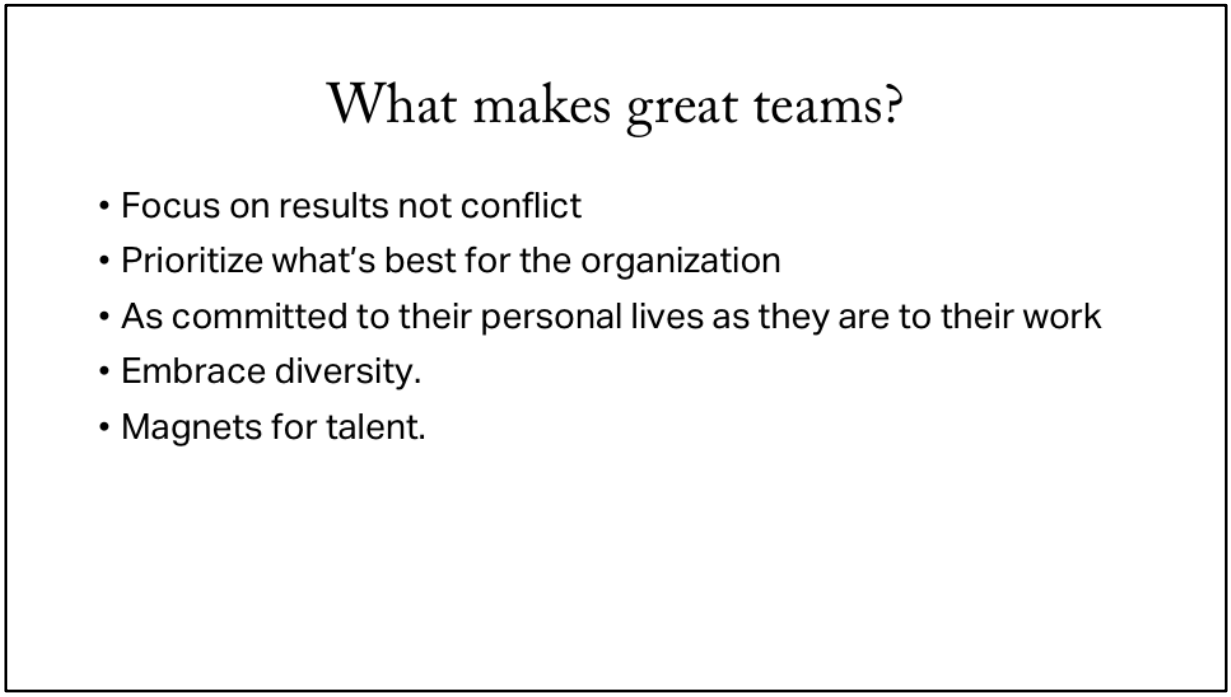
So, who makes up an agile team? The core team is often developers, a scrum master, a tech lead and a product owner. There's other people who are relative to the team: business stakeholders, designers, usability specialists, copy writers, etc. Really it's whoever is needed to make sure the platform development runs smoothly, the core team can even fluctuate depending on the platform and type of projects.

We've got that settled, and we're all on teams, but what we all really want to know is …

When I started investigating what makes great teams, to help shape my team into one of the best, I pulled information from multiple places, different agile resources and some of Gallup's own research.

When a team knows how to leverage each person's strengths, that team can quickly find new ways to drive organization performance.

[Hampton story from Strengths Based Leadership]



Gallup has done years of research on what makes great teams, we even have a book called Strengths Based Leadership where we've illustrated five things that strong teams have in common.
- Conflict doesn't destroy strong teams, because strong teams focus on results.
- Strong teams prioritize what's best for the organization then move forward.
- Members of strong teams are as committed to their personal lives as they are to their work.
- Strong teams embrace diversity.
- Strong teams are magnets for talent.

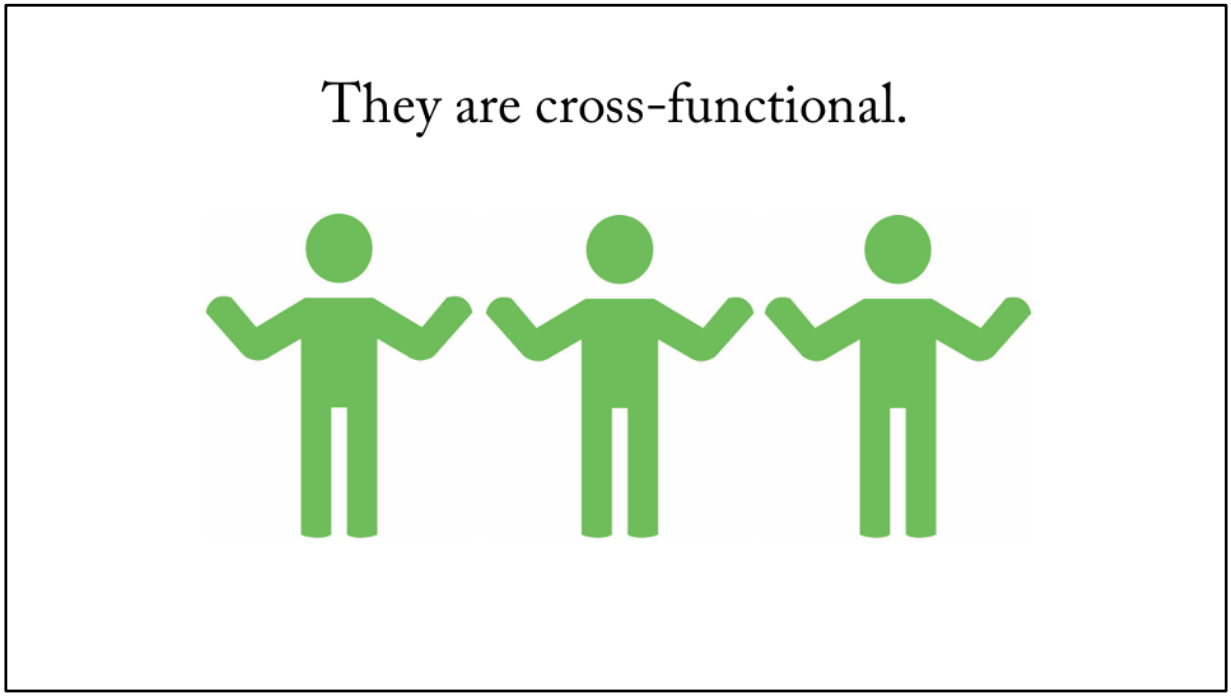
These five themes tie very closely into what we believe causes good scrum teams to become great scrum teams…

They are transcendent.

They have a greater purpose than just the ordinary. They make the decision to be extraordinary.

They are autonomous.

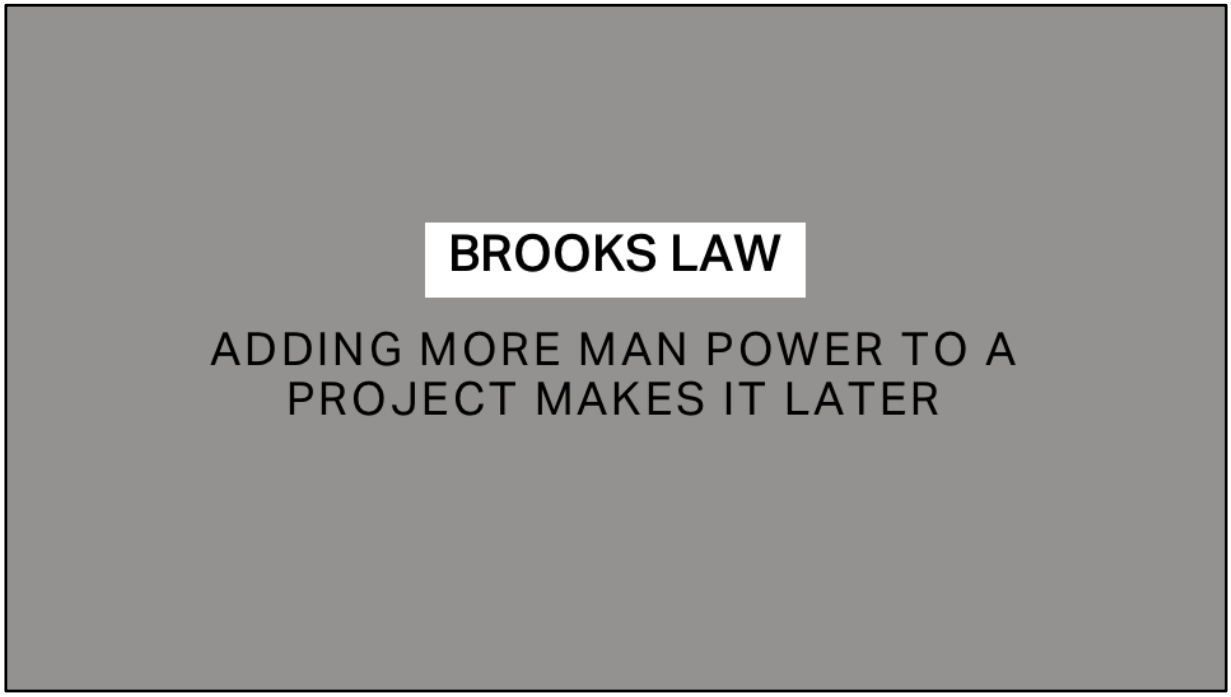
They are self organizing and self-managing. They make their own decisions. They don’t need someone to tell them what to do.



Cross-functional – They have all the skills they need to finish the project. They knowledge share, and learn from each other's specialties.

When great scrum teams have combined this these ideals, teams are able to achieve “hyperproductivity.” Teams can see a 300 to 400 percent increase in productivity among groups that implement scrum well.

[TRANSITION] Now I know what some people are thinking, I want the best team ever to work on best project ever so I’m going to take my top ten performers and create a team… And we’ll be the team that gets everything done because we’re so awesome. Well I’m here to tell you why that’s a bad idea.

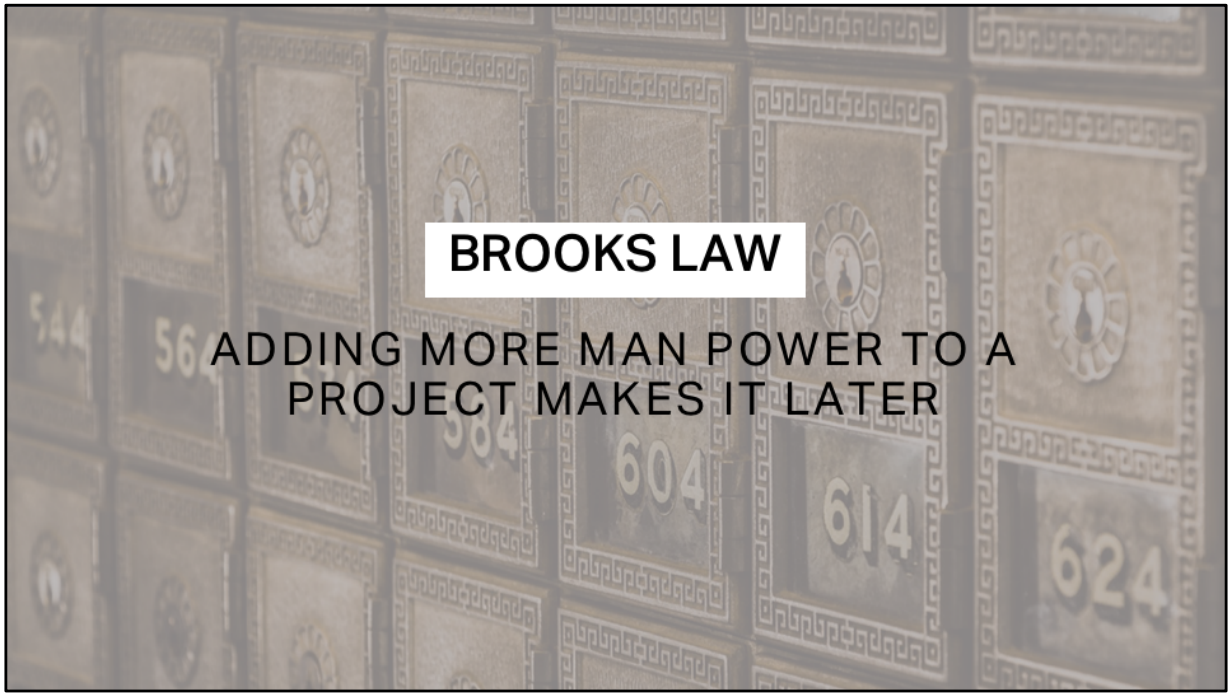


Named after Fred Brooks, from his book The Mythical Man-Month.

Study after study has illustrated that a large team takes about five times the number of hours a small team does.

Brooks studied 491 medium-size projects at a hundred+ different companies of people developing new features. He divided the projects by team size and noticed that once teams grew larger than 8 members they took dramatically longer to get things done. Groups from 3-7 require about 25% of the time that groups that had between 9 to 20 people in them to complete the task.

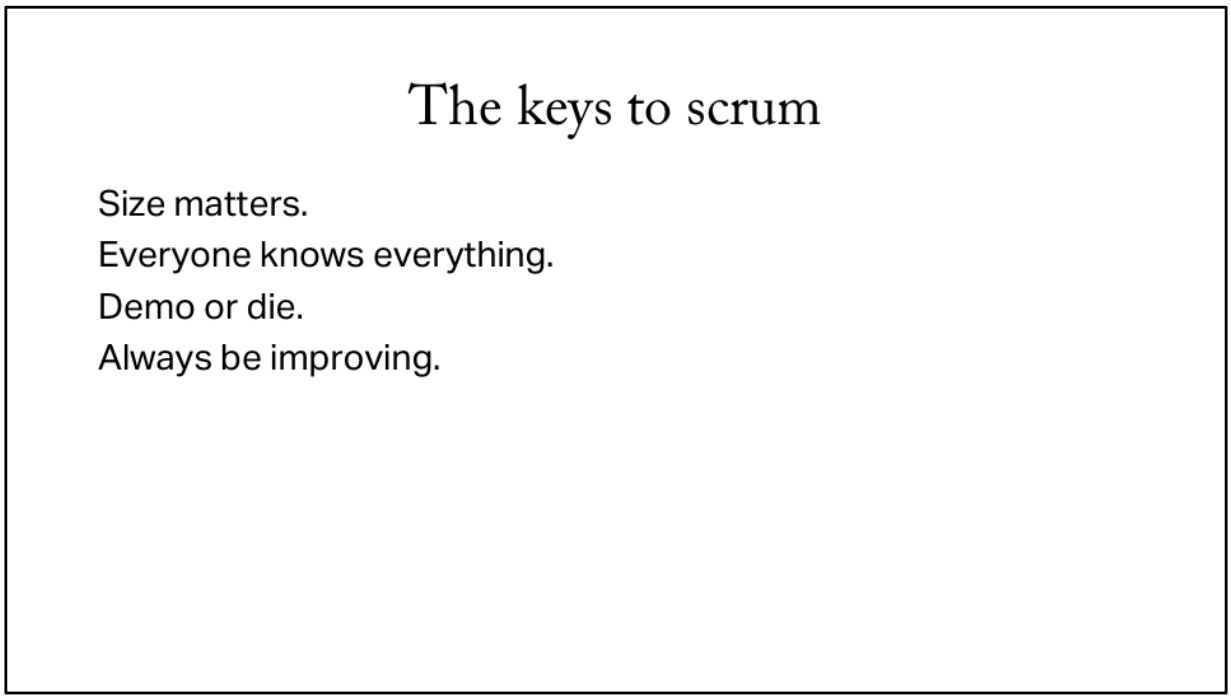
Why?



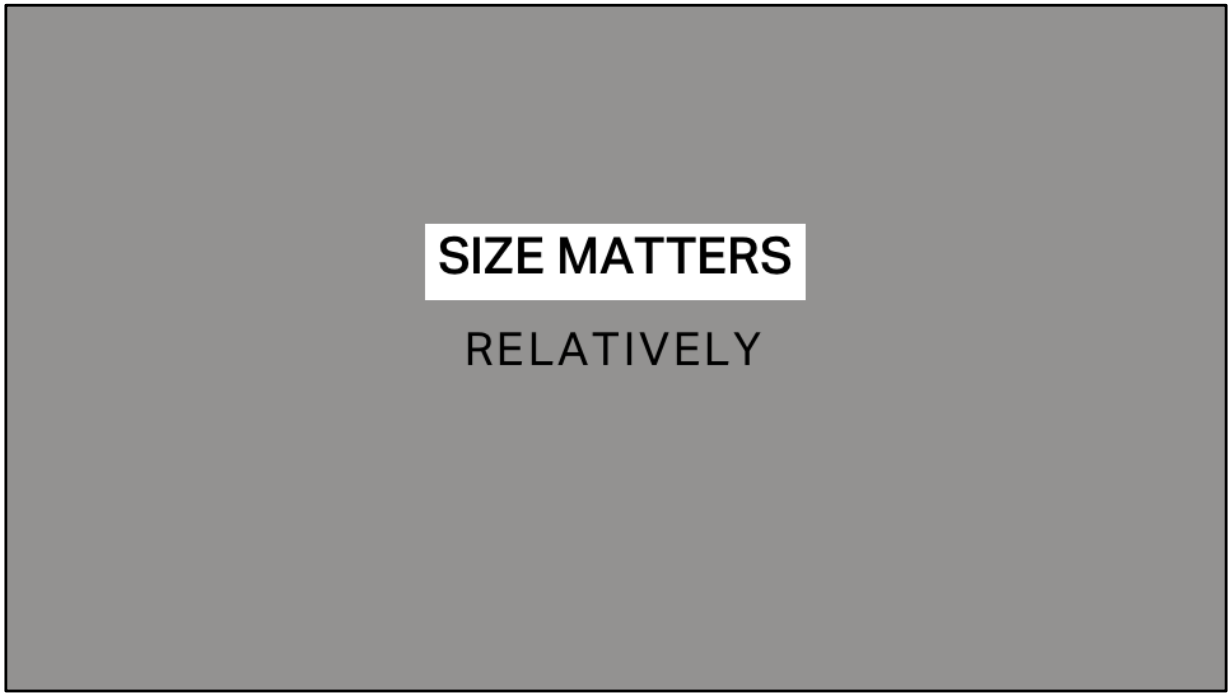
George Miller did a study in 1956 that showed the maximum number of items an average person can hold in short term memory. It's 7, which is why telephone numbers are 7 digits long. Well in 2001, a University of Missouri study wanted to prove if this was actually true. The answer is the average person can only hold FOUR items in their short term memory.

How does this relate to software development? There's two things: The time it takes people to get up to speed with new projects and the number of communication channels grow so great that the human brain can not handle all the info it is receiving.

[TRANSITION] We've talked about great teams and what make great teams, how do you actually execute on scrum?



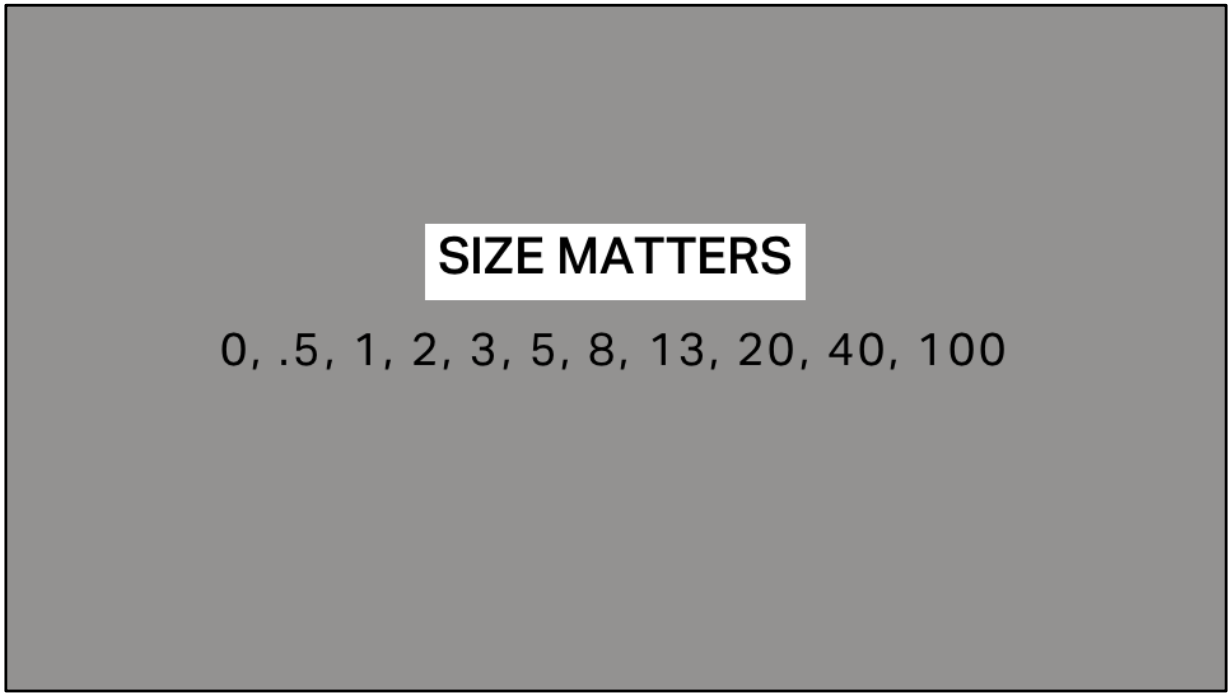
Size matters (in pointing and planning) Is this project a Yorkie or a herd of Great Danes?
Everyone knows everything. Communication saturation accelerates work.
Demo or die. At the end of each sprint, have something that is done and able to be shown off.
Always be improving. At the end of each sprint the team should decide one thing that would make life easier and work on that.



Back in the waterfall days you would get a task and the project manager or whoever would ask, how many hours/days will this take? And you would give them some rough answer that was a range like 10 to 16 hours. They would plug it into their gaant chart as 10 hours and tell the person after you that they would get what they needed from you on August 3rd, 2018, because in the waterfall world everyone delivers everything on time.

Know what humans are notoriously awful at? Figuring out how long it's going to take them to do something. That's one of the many reasons waterfall was so wrong, so often. Often times as a developer your time estimate would be in an "ideal day" where you got to code for 4-6 hours of it. Well how many days are actually ideal? Few. Maybe none.

So scrum did something about estimates.



Instead of estimating hours, we estimate on complexity. This is a struggle for most people at first, but I promise it does get better. You start thinking of projects on a sizing scale, dog sizes is a pretty common one. Project X might be a Yorkie in size, something that will for sure take you less than half a day, probably a .5 on the scale. Then you might get a project in that's a couple days of work, it might be a Labrador. Somewhere between the 3-8 on the scale. Then another project comes in, like redoing an entire 1000+ page website, for instance, that's a herd of Grant Danes. It's definitely 100 or an infinity and it needs broken down more.

So we talked about rugby before and how it relates to scrum, let's talk about the haka. When Sutherland was first developing our scrum principles, he would often make his teams watch the haka, so the could emulate the energy and synergy.

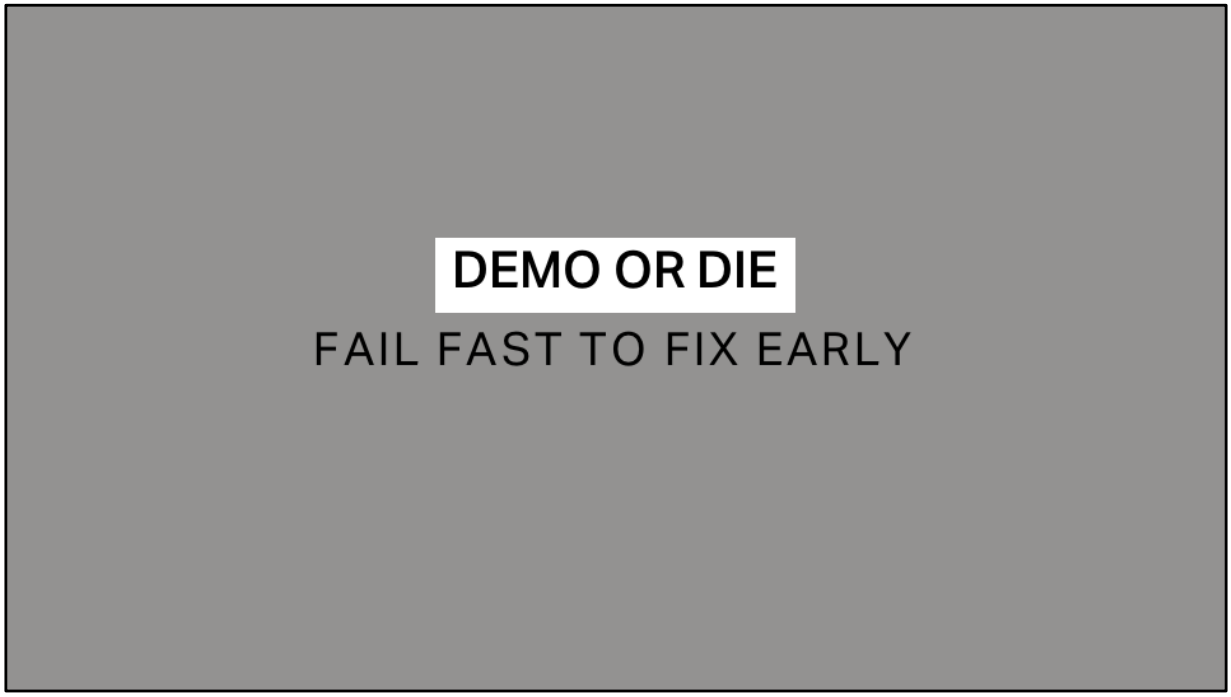
For those of you that don’t know, the haka is a ritualistic dance that the New Zealand All-Blacks rugby team does before each game. It is done in sync and creates this energy amongst the team before they go into battle. I say battle because it was originally a war dance. One of Sutherland’s questions he wanted to answer was how do we energize software teams?! The answer… Daily standups.

Standups should answer the 3 questions:
What did you work on yesterday?
What are you working on today?
What obstacles are getting in your way?

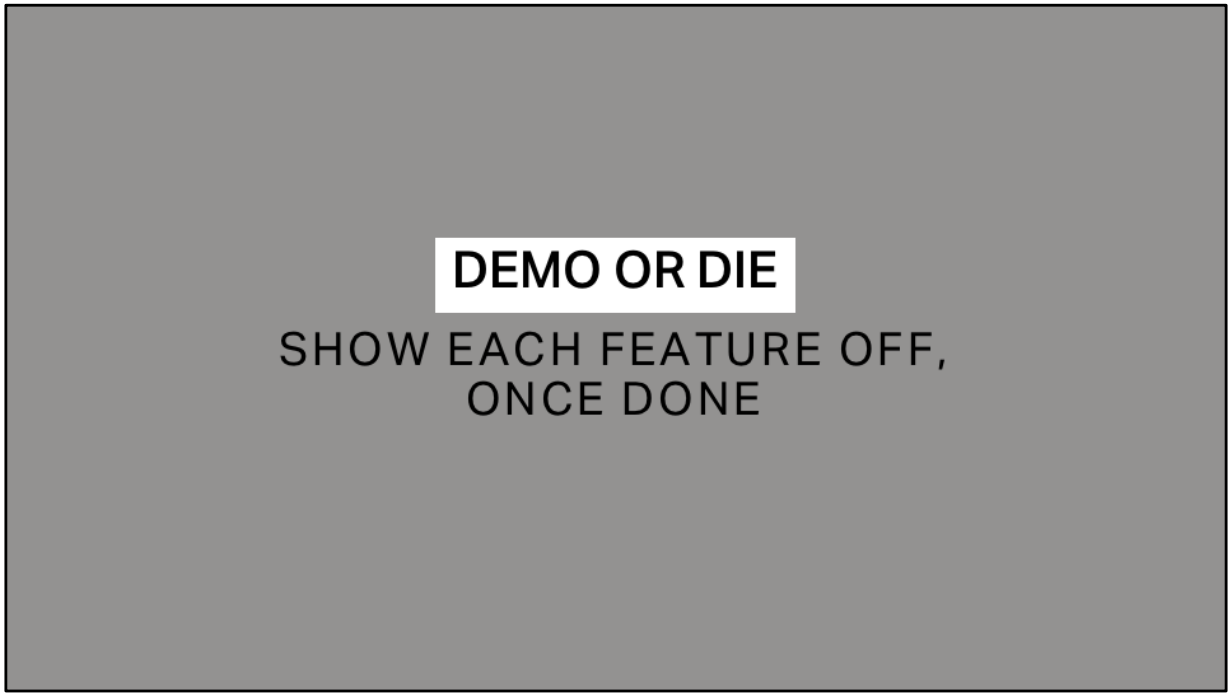
The whole team should know where everything is at in the sprint, and where they should be helping out if they can. They should be able to rearrange their individual prioritizes to help the team accomplish its goal.

They should operate like a football huddle. Dev 1: ”I’m having issues with X.” Dev 2: ”Oh I fixed something similar to X a couple months ago, let’s chat after standup.”

[Tell story about daily standups and the construction projects from Scrum: The Art of Doing Twice the Work in Half the Time]



When we look back at waterfall one of the biggest failures besides time estimates was the fact that it would take literally years before a product saw a product owners eyes. So after years of development and requirements gathering to build giant systems, you could be told that that was in fact not what they wanted and to start over.



Scrum addresses this in the form of sprint demos. At the end of each iteration of work, you should be showing off the new features you have developed. This allows for feedback from the product owners, and it makes sure that you are building the right thing at the right time. Instead of something no one wants or something that doesn't meet the requirements.

Demos should be both a celebration of the work completed and a focusing on the next thing to be built.

Along the same lines as the sprint demo, back in the waterfall days, the teams couldn’t easily get a failing process to get better. Mainly because that process had been decided on 2 years ago, and everything was so planned out that there wasn’t anyway to change it.

So scrum introduced a developer favorite, the retrospective, where your scrum master makes you talk about your feelings.

Retrospectives are a way of figuring out how the team can be/function better. There should be at least one goal or task that comes out of each one to drive improvement in the next sprint.

Don't discuss what you did, but how you did. How can we work together better? What was getting in our way?

[Reference Arthur's Talk about Retrospective biases.]

There's TONS of different ways of doing retrospectives, so find one that works for your team. What went well? What didn't? What can we do better? is just one way. Some others are the 4 L's (Liked, Learned, Lacked, Longed For) and FLAP (Future Considerations, Lessons Learned, Accomplishments, Problem Areas). Many, many more can be found on funretrospectives.com

Resources

BOOKS

Scrum: The Art of Doing Twice the Work in Half the Time (Sutherland)

Essential Scrum: A Practical Guide to the Most Popular Agile Practice (Rubin)

OTHER

Scrum Master Toolbox Podcast

Omaha & Lincoln Agile Meetups

Future Presentations

NEBRASKA.CODE()

1/5th of your users may not be able to use your site

Thursday 2:15 Room L

Mapping the User's Journey

Friday 3:30 Room I

NEBRASKAUX

UX Dark Patterns

June 13th 6:00 @ Gallup

CONNECT

Twitter: @courtneyxann

Questions?

Fill out the evalutions, win prizes!