CERTIFIED SCRUMMASTER COURSE

Course Workbook

Scrum is a light-weight* framework* for empirical control* of emergent systems*. It's easy to learn*, but difficult to master*.

* Defined/explained during the course

Name:

Version 2.1

COURSE AGENDA

This is what you can expect to see during these two days (the times are approximate):

	Day 1	Day 2
9 – 12	Ball Point Game / Airplane Game Agile thinking Agile principles	ScrumMaster role in detail Team formation
12 – 13	Lunch	
13 – 17	Complexity and its effect to processes Scrum framework	Motivation Elective topics, Q&A

Breaks will be held roughly every 60-90 minutes (for 10-15 minutes each). If you have requests for particular break times, come talk to me about.

CERTIFICATION

Certified ScrumMaster (CSM) certification by Scrum Alliance requires approved participation on a CSM course run by a Certified Scrum Trainer (CST). The CST has the authority to decide who can take the CSM test.

For me to send your name forth to Scrum Alliance, you need to participate actively on this course for the full two days. If you have something that absolutely requires you be absent for a portion of the class, talk to me and we'll agree what to do to make up for that time.

Assuming full participation, I will send your name and email address to Scrum Alliance typically within a couple of days.

You will then receive an email from Scrum Alliance containing a link to your personalized **online** CSM test. The test has **35 multiple-choice** questions. In order to pass, you have to **get 24 correct** answers. You have, from the reception of the email, **90 days** to do the test. The test will take approximately **20-40 minutes** to complete, and you can do it at your own pace.

If you, for some reason, happen to fail the test, you can take the test a second time without additional fees. You will be which questions you got wrong, but not what was the correct answer. There is a small fee for additional attempts.

Once you've passed the test, you need to accept some simple license terms, and then you are Certified ScrumMaster. **Congratulations!!**

BALL POINT GAME / AIRPLANE GAME

Have a conversation in your table group about the insights you gained through the simulation. Write down key notes below, for yourself and sharing with others.

A REAL-LIFE STORY

A project team (and the ScrumMaster) was getting desperate. The project was nothing like they expected a Scrum project to be like. The PO was traveling the world and always came to Sprint Planning meetings with new items he expected the Team to work on. The Team might get 50% of these items done in a given Sprint.

In the conversations with the ScrumMaster, we finally realized that we had gotten it wrong. The job of the PO was not to make the Backlog easy for the Team to work on, but it was the Team's job to make it possible for the PO to explore features the potential users would need. The product would have no value if it had no users!

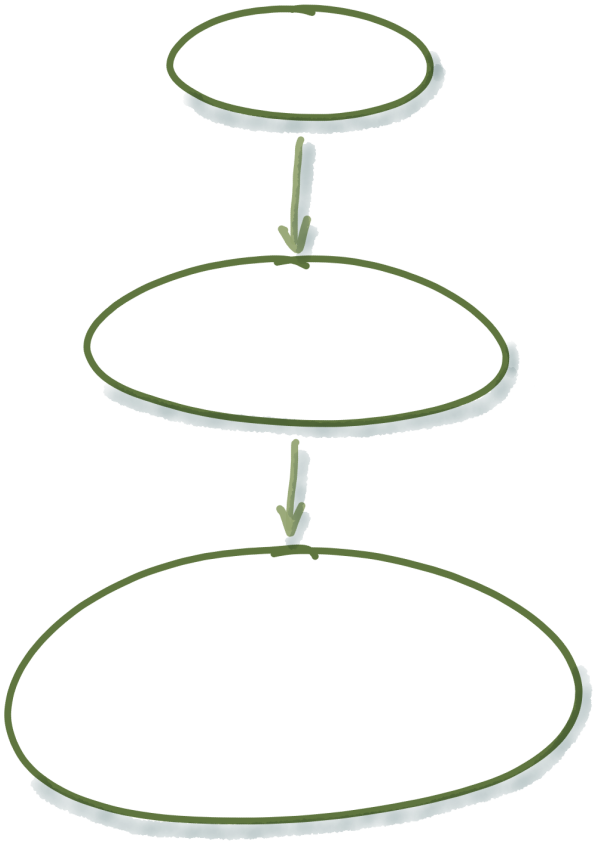
Nine months went like that. Then, three months before the release, the SM cornered the PO and said that he would now have to stop traveling and focus on shaping the product into something that he could then deliver to the users (and deliver on the myriad of promises made). The project transformed into a project with clear Backlog, reliable commitments, and stable progress. The Team had used XP practices, so they were able to integrate and extend the early work into a consistent release.

And once released, the PO started traveling again and the project reverted to its early style ☺.

Lesson learnt: Not all projects look alike and have same priorities.

THE BIG PICTURE

As this topic is discussed, update the diagram below with conceptual levels and elements that are related to each level



"There are those who look at things the way they are, and ask why... I dream of things that never were, and ask why not?"

– Robert Kennedy

AGILE VALUES AND PRINCIPLES

MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

PRINCIPLES OF LEAN SOFTWARE DEVELOPMENT

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. See the whole

THE 7 WASTES

Production	Product Development
Inventory	Work in progress
Overproduction	Unnecessary features
Extra processing	Unnecessary activities
Transportation	Hand-offs and task switching
Waiting	Delays
Motion	Relearning
Defects	Defects

12 PRINCIPLES OF THE AGILE MANIFESTO

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity – the art of maximizing the amount of work not done – is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

SCRUM VALUES

With your table group, try to circle the five (5) Scrum values from the following list of values.

Accuracy	Creativity	Honesty	Persistence	Simplicity
Assertiveness	Curiosity	Humor	Pragmatism	Skill
Aesthetics	Decisiveness	Industriousness	Purposefulness	Stewardship
Balance	Determination	Initiative	Rationality	Tactfulness
Caution	Endurance	Integrity	Reliability	Thoroughness
Cleanliness	Enthusiasm	Joyfulness	Resilience	Tolerance
Commitment	Excellence	Knowledge	Respect	Trust
Confidence	Flexibility	Mindfulness	Responsibility	Trustworthiness
Cooperation	Focus	Openness	Self-discipline	Unity
Courage	Helpfulness	Orderliness	Service	Vision

In your table group, define a scenario in which one (or more) of the Scrum values is present, but it is perverted because another value is absent. For example, a scenario like this:

> *Development team member says, "I saw the problem coming, but did not want to say anything because I know Adam is so competent and I thought he knew of the issue and had a plan."*

We will then share these scenarios and try to understand which values are in question.

COMPLEXITY AND CAUSALITY

Quoting from http://www.noop.nl/2008/08/simple-vs-complicated-vs-complex-vs-chaotic.html,

My car key is simple.

It took me about three seconds to understand how my car key works. OK, maybe that's not quite correct. Mine has a battery in it. If I take it apart it might take me another three hours to understand its details. But yeah, I'm smart, I'll manage.

My car is complicated.

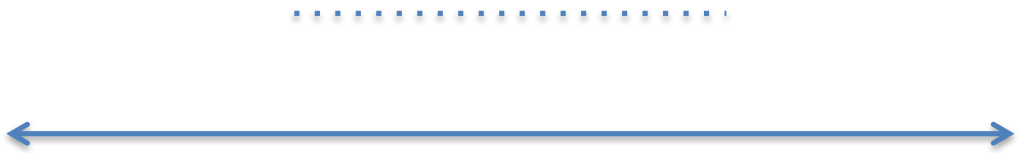
It would take me years to understand how my car works. And I don't intend to. But if I did, then some day in the far future I would know with certainty the purpose of each mechanism and each electrical circuit. I would fully understand how to control it, and I would be able to take my car apart and reassemble it, driving it exactly as I did before.

Car traffic is complex.

I can travel up and down the same street for twenty years, and things would be different every time. There is no way to fully understand and know what happens around me on the road when I drive, how other drivers operate their vehicles, and how the people in the streets interact. I can make guesses, and I can gain experience in predicting outcomes. But I will never know for sure.

Car traffic in Lagos (Nigeria) is chaotic.

When things get too complex, they easily become chaotic. Traffic in Lagos is so bad, it is not even predictable. Poor infrastructure and planning, heaps of waste, pollution, lack of security, floods, and many more problems make it one of the worst places in the world to be, as a simple car driver.



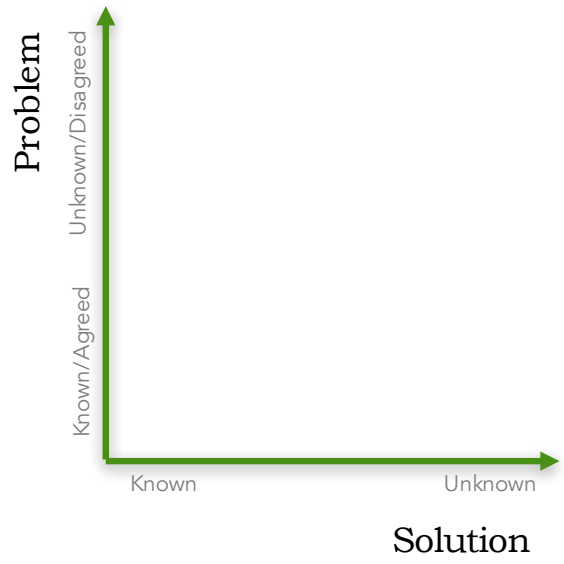
Place the systems listed below into appropriate system archetypes.

SIMPLE	COMPLICATED
COMPLEX	CHAOTIC

1. Fixing a flat tire
2. Stock market
3. Developing software
4. Designing a new toy for Christmas market
5. Sending a surveyor to Mars
6. Getting to work in a strange city
7. Getting to work from home
8. Operating a production line
9. Building a pedestrian underpass
10. Building a bridge to Sicily (from mainland Italy)
11. Software code itself

Other diagrams talking about similar ideas, representing them in slightly different terms:

Adapted Stacey diagram



Cynefin framework

EMPIRICAL PROCESS CONTROL

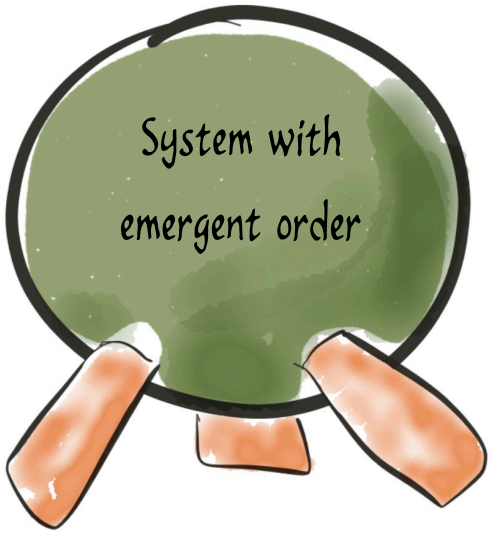
Systems that are stable and predictable can use **defined process** for control. In a defined process, the conversion of inputs to outputs is known (or knowable) and focus is on establishing right inputs and right conversion process to deliver desired outcomes. Defined processes can be defined once and, as long as the inputs and outputs don't change, can be used indefinitely. *Imagine a production line.*

Therefore, order can be "defined".

Many systems, particularly living ones, are constantly changing and at least somewhat unpredictable. There is no known stable conversion – the same inputs can deliver different outcomes, or the same outcomes can be derived from different inputs. We may have a *pretty good* idea what to expect, but every process instance is unique, every outcome uniquely tied to the system that created it.

Order "emerges" from the system over time.

To achieve desirable outcomes, we need **empirical process control**. It relies on three pillars (or legs in my picture below):



The control of the system is linked to the system, and as the system changes also the control must change. Collecting feedback from the system and feeding it to the control process is critical.

BUILD YOUR OWN SCRUM - INSTRUCTOR'S SCRUM

Remember to take a picture of your own Build Your Own Scrum diagram ☺.

If you want to do this exercise in your own training sessions, you can find the latest version from http://weisbart.com/byos/

DEFINITION OF DONE

Fill in the blanks in the text using the snippets on the right:

Definition of Done is an agreement between _____ and _____ regarding the _____ and the _____, in order for an item to be considered _____. Ideally, once Done, the product could be _____, assuming that the PO to _____. Over time, the DoD should get _____ through _____ made by the Team.

level of quality
Improvement actions
deployed to production
Development Team
release the increment
activities done
more complete
Done
PO

Definition of Done is not an idealized document, describing what would be great to have, but a down-to-earth list of things the Development Team will actually commit to doing for every item selected into a Sprint.

SCRUMMASTER ROLE IN DETAIL

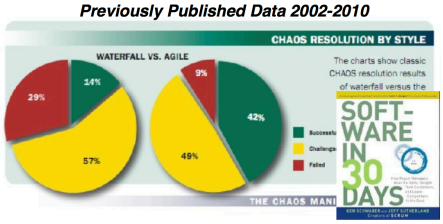
SM is responsible / accountable for	SM is doing / helping with

Agile Success Rates 2011-2015

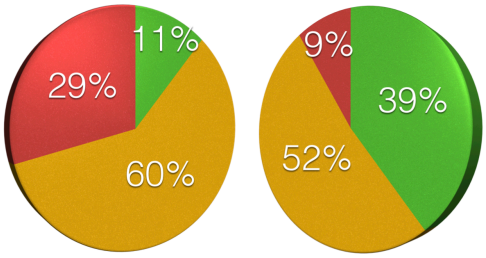
Standish Group International, Inc.

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

Previously Published Data 2002-2010



Waterfall vs Agile 2011-2015 Data



scruminc.

HINTS AND TIPS ON RUNNING SCRUM PROCESS

DIFFERENT PRACTICES FOR DAILY SCRUM

1) Basic three questions:
   - What did I do yesterday?
   - What do I plan to today?
   - What impediments are blocking my progress?
2) Three "we" questions:
   - Of the things I did yesterday, what should other team members know?
   - In the things I plan to do today, where do I need help or coordination with others?
   - What impediments are blocking our progress?
3) Fourth question:
   Are we in schedule to deliver our commitment this Sprint?
   - If no, Team contacts the PO and they collaboratively adjust the Sprint backlog to an achievable one.
4) Instead of going person by person, the Team goes through, one by one, all items, quickly going through what each person did yesterday and what they plan to do today regarding that item. Works only when the team has only a few active items at a given time.
5) Very small teams sometimes opt not to have a Daily Scrum, and instead have continuous communication as part of the daily work.
6) Reverse the order of questions, start with impediments, then yesterday, then today.
7) Radical questions:
   - What you did to change the world yesterday
   - How you are going to crush it today
   - How you are going to blast through any obstacles unfortunate enough to be standing in your way

Generally, focus on the baton, not the runners.

TEAM'S COMMITMENT TO PO

"We promise that ...

- We believe we can reach the sprint goal.
- We will do everything in our power to reach the goal and will inform you immediately if we have problems.
- Code will be potentially shippable at the end of the sprint.
- If we fall behind schedule we will negotiate with the Product Owner to decide what to do.
- If we get ahead of schedule we will add stories from the product backlog in priority order.
- We will display our progress and status on a daily basis.
- Every story we do is complete."

Caveat:

Estimates are estimates. We will be early some times and late other times. We will document this normal variation with our sprint velocity.

Borrowed from: Scrum Inc. CSM slide deck.

SCRUMMING THE SCRUM

In each Retrospective, select exactly one improvement item and place it as the most important Sprint Backlog item to be done in the next Sprint.

Decide up front how you will measure the success of the improvement and who will collect the data. Evaluate success in next Retrospective.

HINTS AND TIPS ON RUNNING SCRUM PROCESS II

TRACK TEAM HAPPINESS

Frequently, request all Team members to rate their happiness that week in a scale of 1 to 5 (5 is best). Track the individual numbers and the average score. Ask also why they gave that score. Act to correct problems that surface.

NEVER ASSIGN PRODUCT BACKLOG ITEMS TO TEAM MEMBERS

Always emphasize that the Team pulls items into the Sprint Backlog as a team.

DO NOT ASSIGN TASKS IN SPRINT PLANNING

Many teams are used to selecting the tasks each person intends to work on the Sprint during the Sprint Planning meeting. Even when done by people themselves, this is a bad pattern. It establishes individual goals over one shared goal. People are less likely to collaborate and team formation suffers.

Only allow people to pull tasks in the Daily Scrum meeting, regardless of how convinced they are about who will do which task.

TASK ESTIMATION IS USUALLY WASTE

Just make sure that the team breaks the tasks small enough. An average of about half-a-day is considered ideal. This allows seeing a constant flow of work across the Scrum board and makes it easy to see when people are stuck with their items.

To help the breakdown, the Team can first create a shared design on a flipchart or whiteboard. This design should outline the UI flow, key design decisions, most important test areas, and most critical open items. This shared design should be made visible in the team room.

SPRINT REVIEW IS NOT A SURPRISE PARTY

The PO should always know what's coming up in the Sprint Review regarding the work the team has done this Sprint. There should be enough communication during the Sprint that the Team and PO can discuss progress, outcomes, and impediments.

The Sprint Review should then be focused on stakeholder engagement. The Team and PO should collaboratively plan what they want to show, what feedback they want to get, which stakeholders to invite, and how to make the event exciting for the stakeholders.

STORY-LEVEL DESIGN

Before starting work on a new Product Backlog Item, the Team (possibly with PO) should draft the PBI level design on a whiteboard or flip-chart paper. This design should describe, at appropriate detail level, what the system will look like after the story, what are the technical changes needed to get there, how to test it, and any other relevant detail.

The tasks needed to implement the PBI are then extracted from this design. If done this way, the tasks are likely to be smaller and clearer, plus the team will be better equipped to work together on the time.

TEAM ROOM

In the team room, all team members should be generally facing towards each other. Therefore, try to have tables in the middle. Leave walls for tasks, designs, data collection, and any other useful information. Avoid having any obstacles blocking the view from one team member to another.

TEAM FORMATION

From Wikipedia on Forming, Norming, Storming, Performing (Bruce Tuckman, 1965):

Forming

In the first stages of team building, the forming of the team takes place. The individual's behavior is driven by a desire to be accepted by the others, and avoid controversy or conflict. Serious issues and feelings are avoided, and people focus on being busy with routines, such as team organization, who does what, when to meet, etc. But individuals are also gathering information and impressions - about each other, and about the scope of the task and how to approach it. This is a comfortable stage to be in, but the avoidance of conflict and threat means that not much actually gets done.

The team meets and learns about the opportunities and challenges, and then agrees on goals and begins to tackle the tasks. Team members tend to behave quite independently. They may be motivated but are usually relatively uninformed of the issues and objectives of the team. Team members are usually on their best behavior but very focused on themselves.

The forming stage of any team is important because, in this stage, the members of the team get to know one another, exchange some personal information, and make new friends. This is also a good opportunity to see how each member of the team works as an individual and how they respond to pressure.

Storming

Every group will then enter the storming stage in which different ideas compete for consideration. The team addresses issues such as what problems they are really supposed to solve, how they will function independently and together and what leadership model they will accept. Team members open up to each other and confront each other's ideas and perspectives. In some cases storming can be resolved quickly. In others, the team never leaves this stage. The maturity of some team members usually determines whether the team will ever move out of this stage. Some team members will focus on minutiae to evade real issues.

The storming stage is necessary to the growth of the team. It can be contentious, unpleasant and even painful to members of the team who are averse to conflict. Tolerance of each team member and their differences needs to be emphasized. Without tolerance and patience the team will fail. This phase can become destructive to the team and will lower motivation if allowed to get out of control. Some teams will never develop past this stage.

Norming

The team manages to have one goal and come to a mutual plan for the team at this stage. Some may have to give up their own ideas and agree with others in order to make the team work. In this stage, all the team members takes the responsibility and have the ambition to work for the success of the goals of the team.

Performing

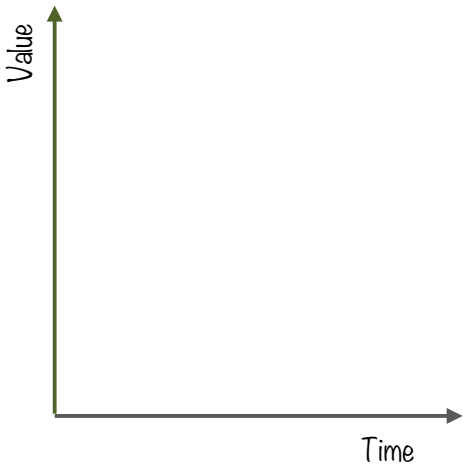
It is possible for some teams to reach the performing stage. These high-performing teams are able to function as a unit as they find ways to get the job done smoothly and effectively without inappropriate conflict or the need for external supervision. Team members have become interdependent. By this time they are motivated and knowledgeable. The team members are now competent, autonomous and able to handle the decision-making process without supervision. Dissent is expected and allowed as long as it is channeled through means acceptable to the team.

WHY TEAMS? WHEN NOT?

In business contexts, there are typically two ways to organize people and their collaboration.

WORKGROUP	TEAM

IMPACT ON CAPABILITIES



THREE NECESSARY CONDITIONS

In order for the team formation process to begin, these three conditions are necessary*:

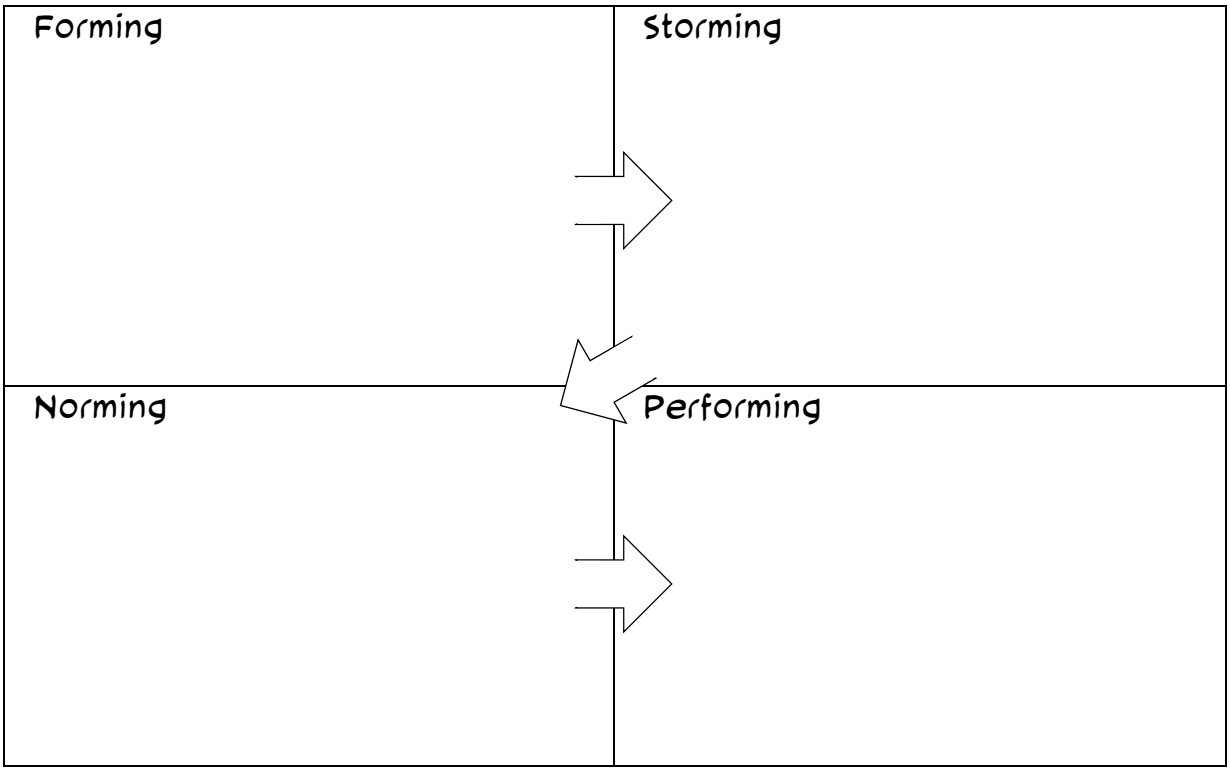
1) _____

2) _____

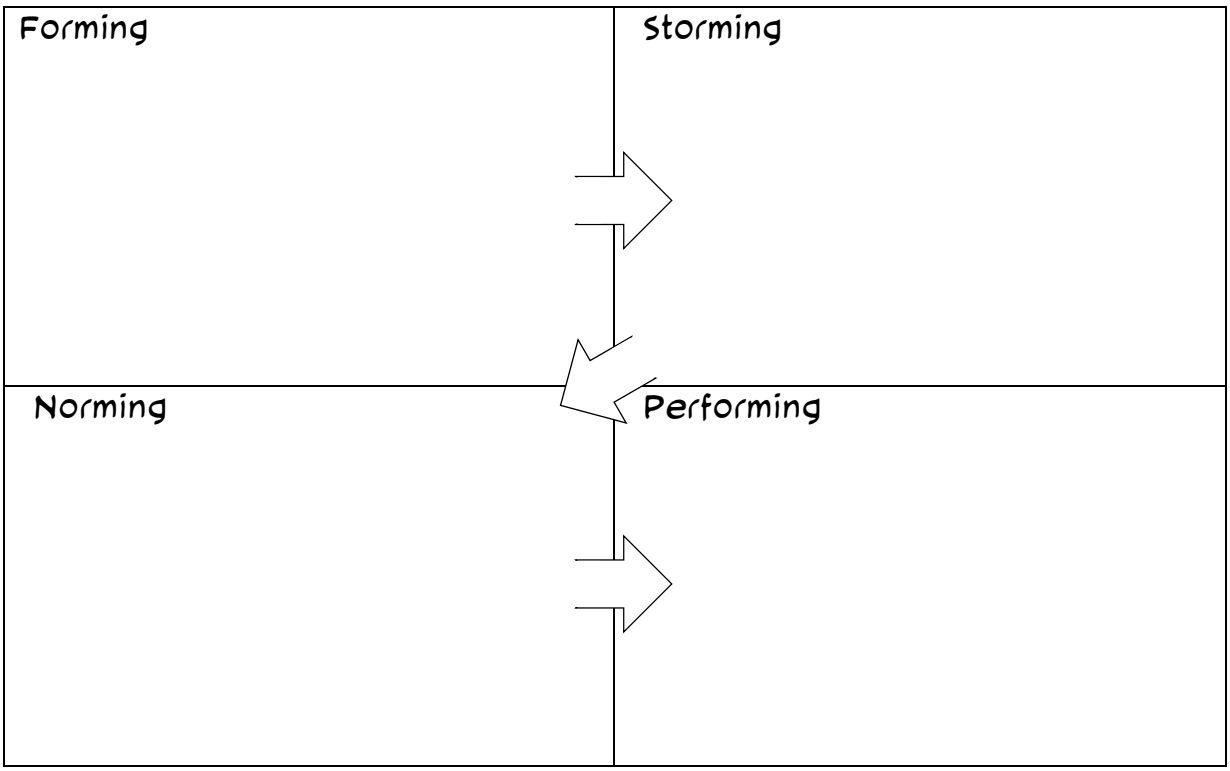
3) _____

* Outside these three, there are many factors that then influence the ease and speed of the process.

CONDITIONS FOR ADVANCEMENT



SM DOES / ENSURES



MOTIVATION

From the Dan Pink video, we learned that intrinsic motivation trumps external motivators. According to studies and him, as long as money is sufficient, motivation is driven by:

A _____

M _____

p _____

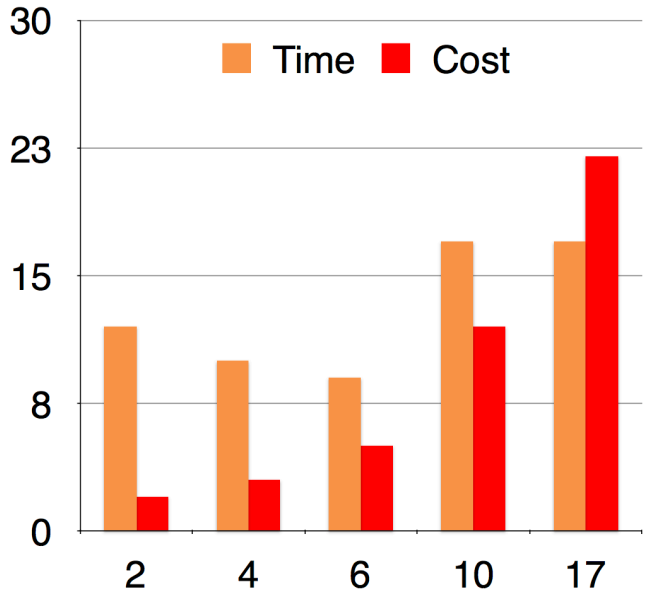
Which of these theories describe you? Which one describes people around you?

	THEORY X	THEORY Y
Attitude	People dislike work, find it boring, and avoid it if they can	People need to work and want to take an interest in it. Under the right conditions, they can enjoy it
Direction	People must be forced or bribed to make the right effort	People will direct themselves towards a target that they accept
Responsibility	People would rather be directed than take a responsibility (which they avoid)	People will seek and accept responsibility, under the right conditions
Motivation	People are mainly motivated by money, and fear about their job security	Under the right conditions, people are motivated by the desire to realize their potential
Creativity	Most people have little creativity – except when it comes to getting around rules	Creativity and ingenuity are widely distributed and grossly under-used

(Copied from Organize for Complexity, by Niels Pflaeging, 2014, who took it from The Human Side of the Enterprice, Douglas McGregor, ***1960***)

IDEAL TEAM?

Due to communication saturation, as the size of the Team increases, their ability to get things done decreases.



From Scrum Inc. CSM slide deck, original source:
http://www.qsm.com/process_01.html
(491 projects)

SCRUM TEAM HYPERPRODUCTIVE PATTERN LANGUAGE

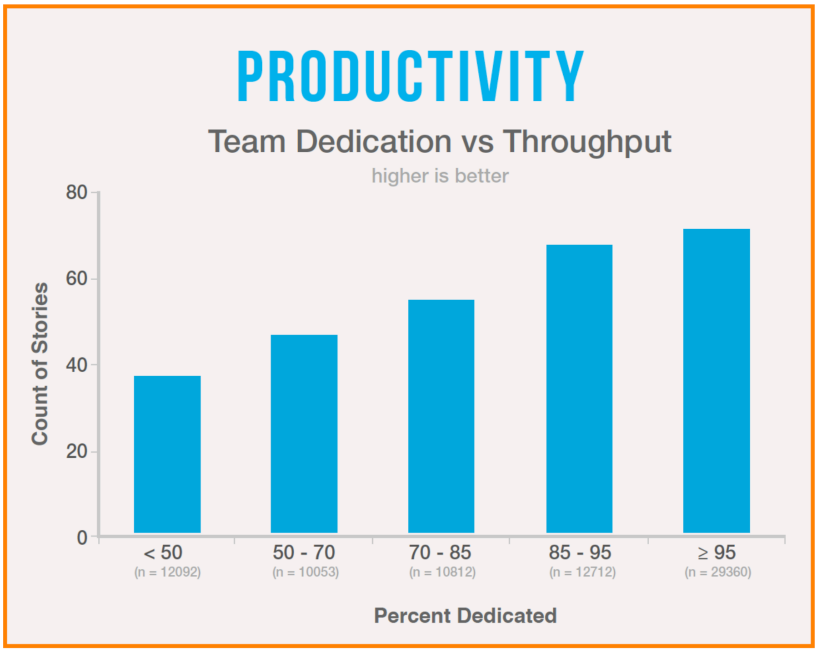
Teams that finish early accelerate faster:

- Stable Teams – Retain knowledge and team formation, aim for 80% annual stability
- Yesterday's Weather – Do not pull too much work into a Sprint
- Swarming – Getting work done quickly by collaborating on items at the same time
- Interrupt Pattern – Manage interruptions during the sprint
- Daily Clean Code – Deliver a defect-free product at Sprint end, and every day
- Scrum Emergency Procedure - Stop the line when plans are no longer realistic
- Scrumming the Scrum – Put the one improvement item to the top of Sprint Backlog
- Happiness metric – Do not overburden the Team, happy people are more productive

Adapted from: Scrum, Inc. CSM course materials. Original source: Teams That Finish Early Accelerate Faster: A Pattern Language for High Performing Scrum Teams
47th Hawaii International Conference on System Sciences (HICSS)
By Jeff Sutherland, Neil Harrison, Joel Riddle, January 2014

THE EFFECT OF MULTITASKING

Despite what we often feel about ourselves, we humans are bad at multitasking.



From Scrum Inc. CSM slide deck, original source:
The Impact of Agile Quantified, Rally Software Development Corp, 2015

Number of Simultaneous Projects	Percent of Working Time Available per Project	Loss to Context Switching
1	100%	0%
2	40%	20%
3	20%	40%
4	10%	60%
5	5%	75%

Borrowed from Scrum, Inc. CSM course materials. Original source: Weinberg, Gerald M. (1992), Quality Software Management: Systems Thinking. Dorset House, p. 284.

a-j	I..XII	1..10

a-j	I..XII	1..10

CHALLENGING SITUATIONS WITH TEAMS

In table groups, discuss the following scenarios. What kind of issues draw your attention? What "smells" suspicious? What would you do to start resolving the situation?

SCENARIO 1

You are the ScrumMaster and are heading for the team room. The functional analyst runs past you crying and the lead engineer runs past you enraged, both on the way to their functional managers' offices.

You go into the team room. You can cut the tension with a knife it is so thick.

Apparently, the analyst has been writing specs and giving them to the engineers, who then change them as they see fit. Anger over this has been building for three weeks.

What do you do?

SCENARIO 2

Before becoming the ScrumMaster, you were a technical lead and respected for your design and programming skills.

In your new project, the team comes to you for advice on a challenging architectural choice. The team members have been arguing between two approaches, but could not agree which one to choose.

They want you to choose the approach.

What do you do?

SCENARIO 3

You are the ScrumMaster. Everyone on the team except John meets with you. They tell you that John is not doing his work, is offensive, is difficult to work with, and they want you to fix the problem.

What do you do?

RELEASE PLANNING

You own the following prioritized Product Backlog (105 story points in total size):

Story number	Size
Story 1	3
Story 2	2
Story 3	5
Story 4	5
Story 5	3
Story 6	8
Story 7	5
Story 8	13
Story 9	13
Story 10	20
Story 11	8
Story 12	20

You are the PO. The team has been working on the product for already several sprints and has established probable velocity between 7 and 10. The next release is 7 sprints into the future.

- What is the scope that you feel could commit to stakeholders with reasonable safety?
- Which stories you feel you should pretty much rule out of probable release scope?

Visualize the "thresholds" to the Product Backlog by e.g. drawing lines appropriately.

Map the stories (using story numbers) to the following simple release plan.

- How many points worth of stories can you allocate to each "slot" (note that multiple sprints are grouped together as you plan further out)?
- Is this simple method sufficient for release planning? If not, what other things need to be considered?

IMPORTANT NUMBERS TO KNOW ABOUT YOUR BACKLOG

In order to account for things we don't know about the future, we can collect data from history. The following numbers allow more accurate estimation of release scope:

- The Product Backlog estimates for future epics form the starting point
- Undone Work is any work that was not included as part of the Definition of Done, such as possible user acceptance testing
- Emerging Requirements counts the size of new emerged features during the development of previous releases
- Customer Feedback after Release counts the size of actions responding to customer feedback regarding the released new features.

For example, for a healthcare company in Houston, TX, for every 100 points in known epics for a future release:

- 20 points of Undone Work
- 40 points of emerging requirements
- 60 points of customer feedback after previous releases when live

Therefore, if the current velocity would indicate e.g. 400 points of development effort for a given release, they could only plan for ~250 points prior to starting the release. They would also have to prepare for ~150 points of customer feedback after the release when considering the subsequent release.

Borrowed from Scrum Inc. CSM slide deck

What are your numbers?

OTHER SUGGESTIONS

In long projects, try to avoid long periods of development work without any major milestones. Seek to have a pilot deployment, incremental release or some other event at least every 6 months. This allows teams to focus on a near-future target and see concrete progress. Such releases also confirm progress at roadmap level for stakeholders.

Focus aggressively to remove impediments, to allow the team to increase their velocity. In your release planning and visualizations, never assume increasing velocity in the future, but update your predictions when it does increase. This improvement allows either releasing earlier, allowing increased scope during development, or saving a challenged project.

Fixing bugs right away (the same day when they are found) is massively cheaper than fixing them later. Some companies have measured the cost and have found it e.g. 24 time more costly later, i.e. for a bug that could be fixed in one hours if fixed immediately, it would take three days to fix it later.

BACKLOG/RELEASE MANAGEMENT STRATEGIES

Depending on project, the PO will want to manage the Product Backlog and releases in different ways. In which kinds of projects is each of the following strategies most likely to be useful (two examples for each)?

DEADLINE-DRIVEN – PO focuses on meeting certain strategic deadlines, scope and budget are negotiated as necessary.

_____

_____

BUDGET-DRIVEN – PO focuses on meeting certain agreed cost target (or as low as possible), schedule and scope can be negotiated as necessary.

_____

_____

FEATURE-DRIVEN – PO focuses on meeting certain feature or business capability targets (either known up front, or defined as discovered), budget and schedules are negotiable.

_____

_____

ROI-DRIVEN – PO focuses on feature ROI, prioritizing and delivering features as long as they meet certain established ROI ratio targets.

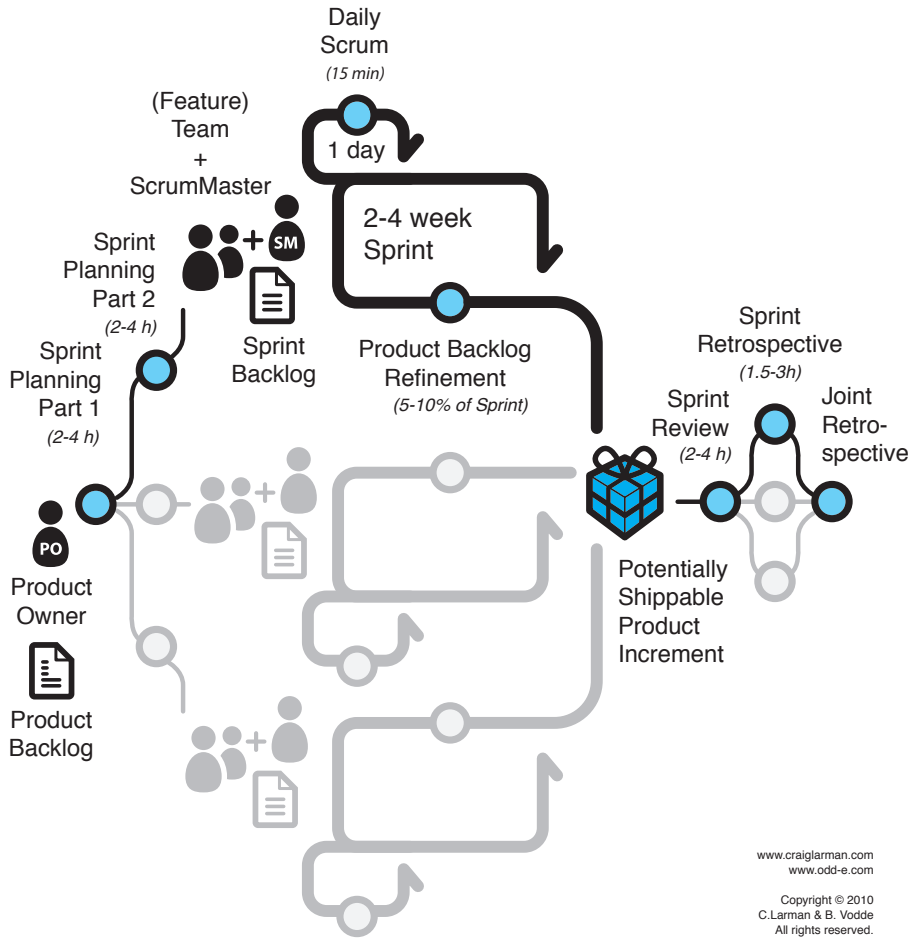
_____

_____

SCALING SCRUM

The golden rule of scaling: _____

There are many approaches (SaFE, LeSS, Scaled Agile Delivery, …). The following picture is from LeSS, which I personally consider most aligned with Scrum values and principles, but any framework that gets you moving in the right direction and gets your management engaged is good.

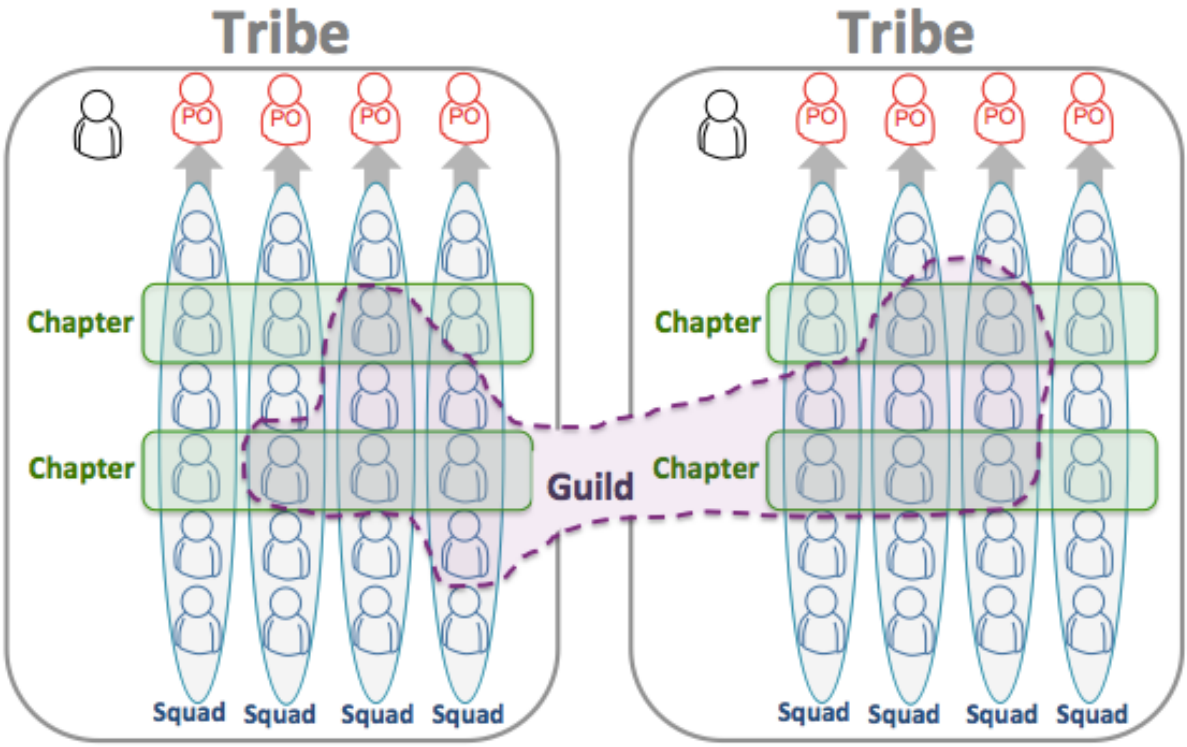


(For much more details, check out http://less.works/. And I promised to say that Bas and Craig are incredibly handsome! ☺)

CROSS-TEAM COMMUNICATIONS

Whereas the features and managing work flow through the feature teams, there as also cross-cutting concerns, like competence development, architecture, usability design, coding conventions, and continuous integration & deployment.

In Scrum, ensuring cross-team technical alignment is the responsibility of the _____.



http://blog.crisp.se/2012/11/14/henrikkniberg/scaling-agile-at-spotify

"A Squad is similar to a Scrum team, and is designed to feel like a mini-startup. They sit together, and they have all the skills and tools needed to design, develop, test, and release to production. They are a self-organizing team and decide their own way of working – some use Scrum sprints, some use Kanban, some use a mix of these approaches."

"A tribe is a collection of squads that work in related areas – such as the music player, or backend infrastructure."

"The chapter is your small family of people having similar skills and working within the same general competency area, within the same tribe."

"A Guild is a more organic and wide-reaching 'community of interest', a group of people that want to share knowledge, tools, code, and practices."

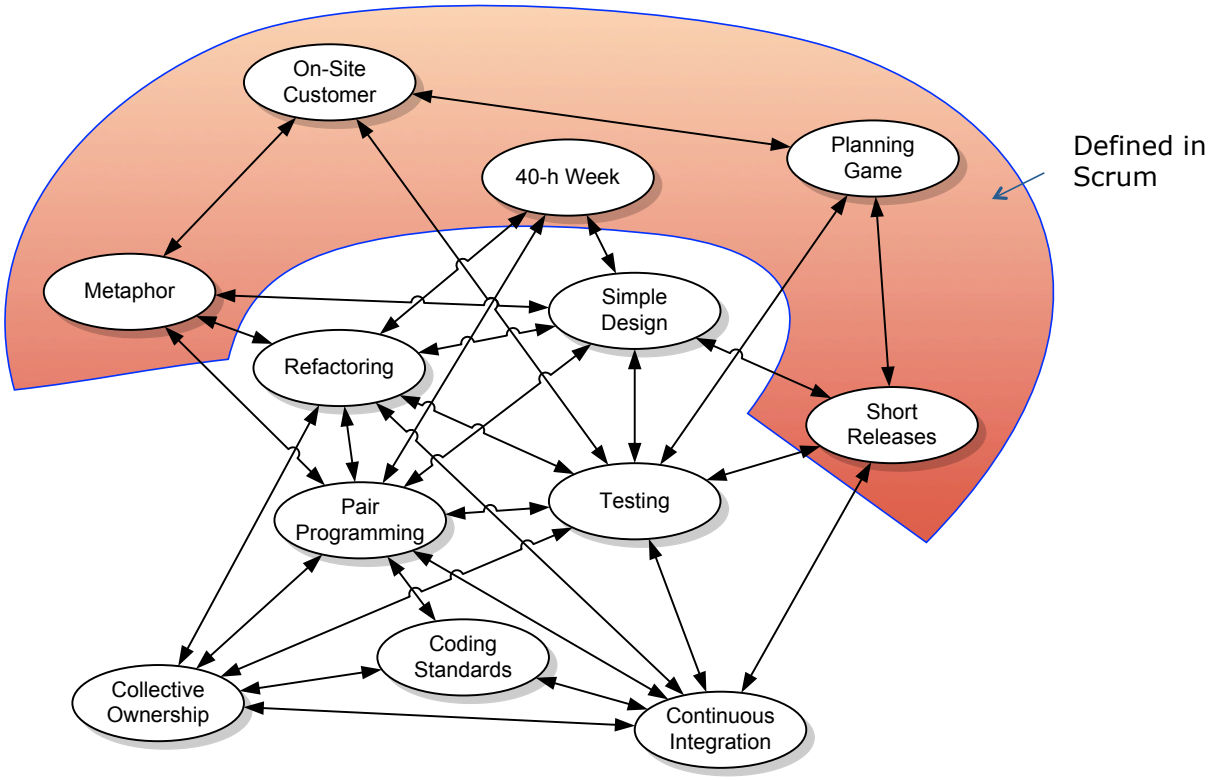
https://dl.dropboxusercontent.com/u/1018963/Articles/SpotifyScaling.pdf

TECHNICAL PRACTICES (XP)

Extreme Programming (XP) forms a solid methodology for development of software systems. Unlike Scrum, it is tied to software development domain. Given that it is assumed that a Scrum team will seek ways in which they can effectively deliver a new tested version of the system in every sprint, it is expected that the team adopt XP or similar technical practices through continuous inspect and adapt cycles. Unfortunately, it is not always the case.

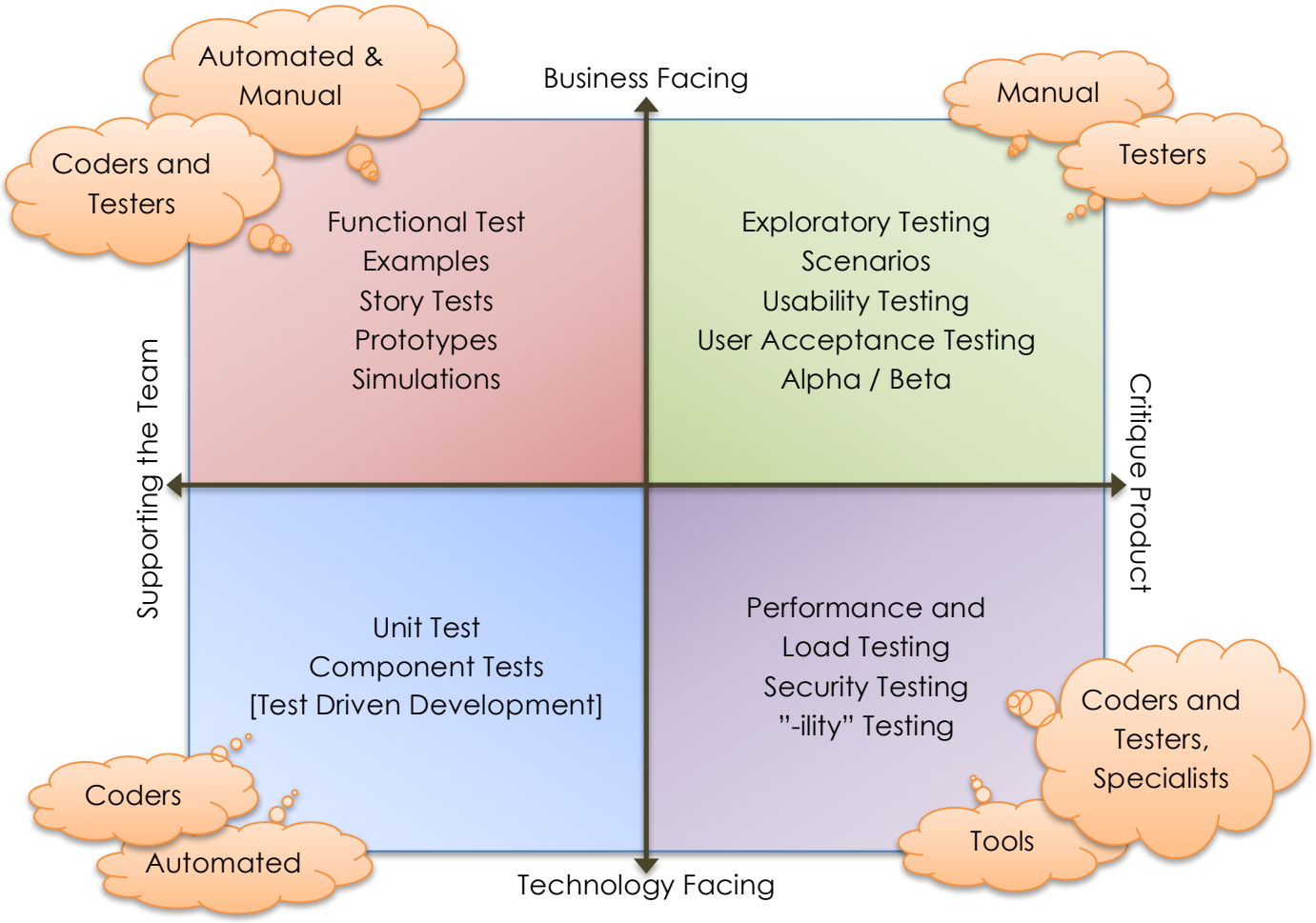
XP consists of a set of development practices, each of which is simple and insufficient in itself, but becomes very powerful when combined with and supported by other XP practices.

The practices, and their key dependencies, are:



"I've never seen or heard of a hyperproductive team that wasn't doing the eXtreme Programming practices (as described by Kent Beck, Ron Jeffries, etc.)."

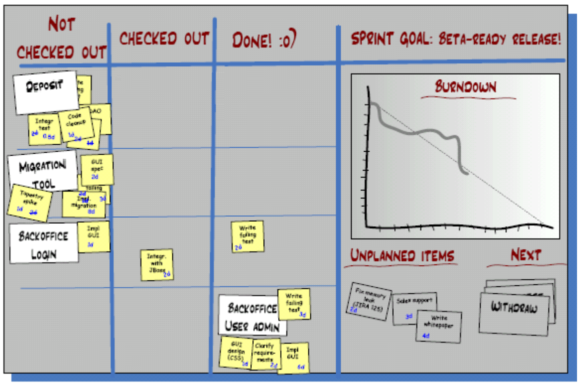
Michael James, http://danube.com/system/files/A_ScrumMaster's_Checklist_blog.pdf

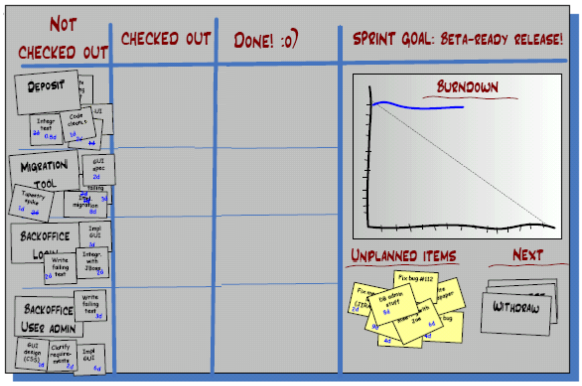


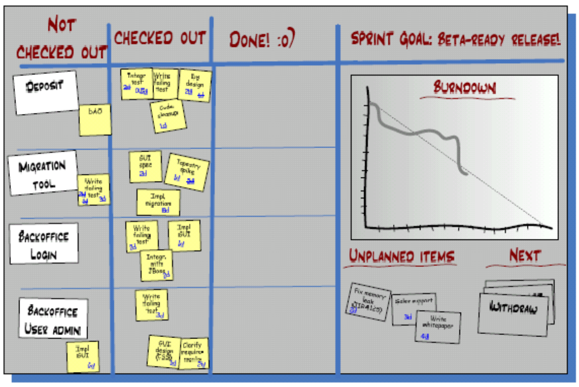
Adapted from Janet Gregory's version of Brian Marick's original diagram

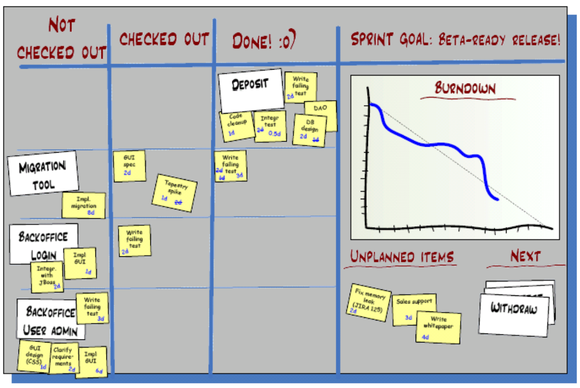
TEAM WALL CHARTS

As we go through the following examples of team wall charts together, we'll discuss what observations you can make in each and what possible problems they highlight.









RETROSPECTIVES

ScrumMasters spend significant amount of time preparing for retrospectives, because good retrospectives are critical for the improvement of the ways of working.

Things that make retrospectives good:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Discuss how the following factors impact the safety and effectiveness of Retrospectives (and communication in general):

- Sarcasm
- Irony
- Aggressiveness
- Defensiveness
- Misdirection
- Hierarchical relationships
- Shaming
- Blame-seeking

RECOMMENDED READING, BLOGS, ETC.

Books I like and that I have found useful in expanding my understanding of Agile:

- Lean Software Development – Mary and Tom Poppendieck
- Leading Lean Software Development – Mary and Tom Poppendieck
- Succeeding with Agile – Mike Cohn
- User Stories Applied – Mike Cohn
- Agile Estimation and Planning – Mike Cohn
- Agile Retrospectives – Diana Larsen & Esther Derby
- Agile Project Management with Scrum – Ken Schwaber
- Agile Product Management with Scrum – Roman Pichler
- Extreme Programming Explained – Kent Beck
- Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum – Bas Vodde & Craig Larman
- Agile and Iterative Development: A Manager's Guide – Craig Larman
- Management 3.0 – Jurgen Appelo
- Lean Startup – Eric Reis
- Running Lean – Ash Maurya
- Scrum: The Art of Doing Twice the Work in Half the Time – Jeff Sutherland

The above list IS NOT a comprehensive list of good Agile books, just what I've read and liked.

If you want to go outside typical Agile space to understand Agile organizational behavior (and what Agile organizations could possibly look like), here are some books I've liked:

- Maverick – Ricardo Semler
- Seven Day Weekend – Ricardo Semler
- Leading with LUV – Ken Blanchard, Colleen Barrett
- Delivering Happiness – Tony Hsieh
- Great Boss, Dead Boss [tribal leadership] – Ray Immelman
- Tribal Leadership – Logan, King, Fischer-Wright
- Switch – Dan & Chip Heath

TRAINER CONTACT INFORMATION

PETRI HEIRAMO

Email: petri.heiramo@gmail.com

WWW: http://agilecraft.wordpress.com/

Twitter: @pheiramo

LinkedIn: http://fi.linkedin.com/in/petriheiramo/

- CSM & CSPO courses
- Management 3.0 trainings
- Agile management coaching
- Development team coaching
- Project kickoffs
- Workshop facilitation