Session 2

Day 2 Intro
How to stay connected with the team
Exam
Sprint Planning
Daily Scrum
Scenarios
Sprint Burndown
End of Sprint Review
End of Sprint Retrospective
    Agile Retrospectives Overview
Backlog Refinement
    Estimation

Facilitation tools - Miro, Mural, Jamboard, Liquid or Lucid chart

Day 2 Intro

- Highly recommended to take the exam at the end of the day (60 minutes)
    - If taken within the 3 days and fail, will provide a 1-1 tutoring
    - 2 attempts total

How to stay connected with the team

- Setting up a podcast
- A slack channel will be created
- Team details - https://docs.google.com/spreadsheets/d/1ZOhtSiibrWCX4xVwu6QdDzSvwPca7-_woJlqcITCfyM/edit#gid=0

Exam

1. Once you get the cert, add it to LinkedIn and tag the people who you took the class with.
2. Like/comment on others’ certification
3. Tag Scrum Alliance
4. Tag GMo

Sprint Planning

💡 POs do not decide on a sprint goal, they can propose, but cannot decide. The team needs to decide together.

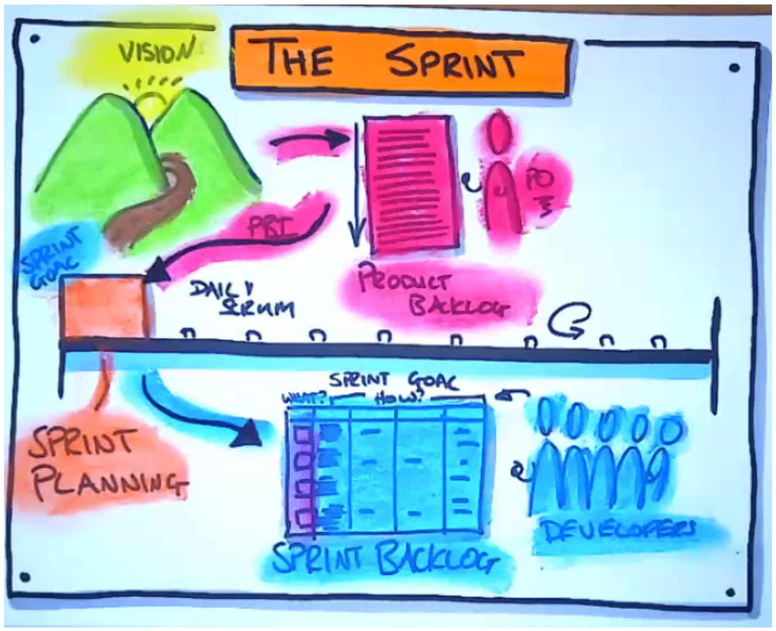
- Sprint planning and capacity is not a calculation, its a decision
- Developers decide how much they can do - based on who is available for that sprint
    - If taken too much or too less, it can be discussed during the retrospect

Daily Scrum

Day 1 - we plan the sprint, what about the other days? How do we track the progress? How do we know the roadblocks? That is when Daily Scrum comes into play. What do we do during that meeting?

- No more than 15 minutes
- It's a simple planning meeting for what we are going to do for the rest of the day
- Highly specialized, brief opportunity to adjust and sync with the team
- Discuss the priorities based on the teams’ availabilities and emergencies
- Discuss delays and adjust in real-time
- Anyone can join that meeting to get the update.
- Sprint planning is the 2-week plan. Think Daily scrum as 24-hour planning.
    - To agree on what we are planning on doing that day

💡 The daily scrum is a developers' meeting, not CSM or POs meeting.



Daily Scrum - 15 minute daily calls for developers to sync

💡 What worked on yesterday, what are we working on today, what are the roadblocks - this is how it used to go. It does not have to be just these. it's not prescriptive. It's open.

Scenarios

1. Scrum is taking too long
    a. Why
        i. Someone is taking too long

        ii. Insufficient sprint planning
        iii. Amount of attendees
        iv. Off topics
        v. The team is too big
    b. What
        i. Setting up a parking lot
        ii. Split the teams into smaller chunks
        iii. Time checks during the meeting
        iv. Keeping the team engaged and focused
    c. Notes: Imagine staring at someone for 9 minutes, you get distracted in 30 seconds. So to keep them focused, it needs to be precise and engaging. As CSM it's a responsibility to increase transparency. CSM has to make sure the Daily Scrum is 15 minutes valuable.
        i. Having clear expectations and hardtops
        ii. Parking lots
        iii. Setting up 16th minute - a follow-up meeting
        iv. Show a visible countdown timer
2. Not everyone is attending the scrum
    a. Why
        i. The timing might be booked by all the developers
        ii. Internal tension within the developers
        iii. Not enough value
        iv. Too much or too little work
    b. What
        i. Gather feedback from each developer
        ii. Motivating the team
        iii. Changing the time of the scrum

c. Notes: Step 1 is always the same - talk to person 1-1. The reason Daily standup is called a standup is that it's so short that you don’t have to sit. Making standing up mandatory is to make sure it's quick and does not get uncomfortable. The more of a ritual you make it, the more of a tradition it becomes.

    i. Can set up who wins the best virtual background

    ii. Can share the trivia question

    iii. Can share a one-liner

Sprint Burndown

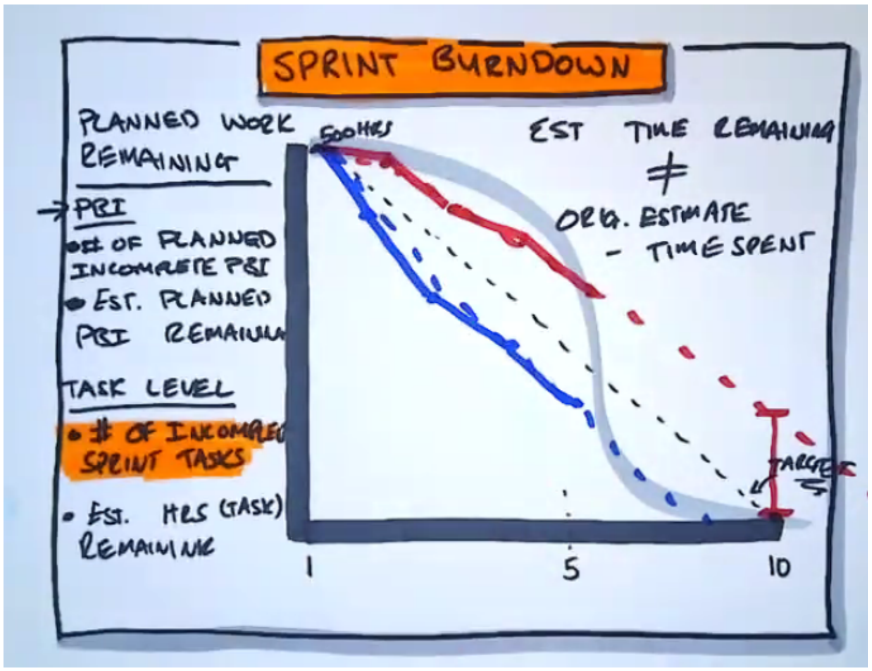
The PBIs are listed for the sprint and then they are broken down into tasks. Each task will have an estimated hour to complete. A sprint burndown is a graph and projection of where we are at, how much is left, will we be able to finish the sprint goal.

Y-axis - Planned work remaining

X-axis - Sprint days

Blue - we are ahead of time and might be able to finish the work early. Can risk by choosing what to focus on.

Red - we will not be able to reach the sprint goal. Cannot risk making a mistake. have to be careful where the time is invested and focus on high-value items.



- The estimated time remaining is not the original estimate minus time spent
- The estimated time remaining is a re-estimation of how long it would take to finish the task.
- Developers use this tool

💡 It's a tool. It's not mandatory to use it.

💡 GMo's way - Preferred way of showing planned work remaining is by showing the Number of incomplete sprint tasks. Prefers exporting the data using tools and drawing his own chart using google sheets or excel.

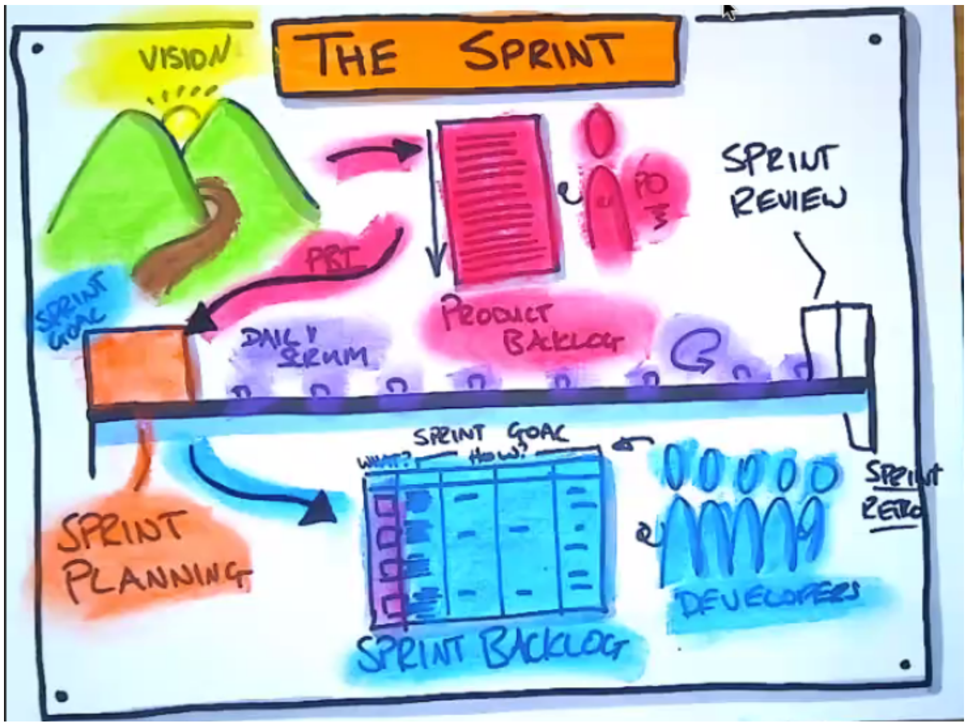
💡 As CSM - have to be careful about who is using it and who has access to it. Because its not always linear and who don't know that (managers) might panic without knowing the trend.

End of Sprint Review

💡 Sprint - Act, inspect, and adapt.

The review is when we present the latest increment and hear the ideas of the customers and the stakeholders. The output of the review is the change to the product backlog. If there is no change to the Product backlog, it means we have a perfect backlog (which is unlikely).

CSM's responsibility is to facilitate the quality of the feedback. For example, talking about this sheet as a product,



1. the visual representation is great
2. The color-coding
3. The process arrows

4. Appreciates that it's done live and precise
5. It's simple

By doing this in meeting and asking them directly, got 5-6. However, if asked the class to put it on sticky, 3 per member, there will be 90+ comments.

> 💡 The CSM's responsibility is to properly facilitate the review meeting and to get quality feedback which can influence the quality generated and the product backlog.

A sprint review needs to be a product-centric/user-centric/customer-centric review of the overall product but not that of the work done on just that sprint. The feedback should be focused on the overall product but not the tasks that the developers worked on in just that sprint. Remind them what users used to do before the feature was implemented, show them how easy it is now

> 💡 Review is not where PO **GIVES** feedback, it's where they **GET** feedback. PO should provide feedback during the sprint.

Often time GMo attended reviews/demos where each developer go through the changes they went in detail which most of the stakeholders might not understand.

End of Sprint Retrospective

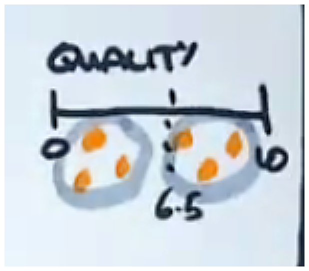
The final thing we do in the Sprint. We will be opportunistic in finding ways to improve. If we find any which can improve our process, we implement them. *Facilitated techniques are like a knife, cuts well initially and dull down eventually over time.*

> 💡 Recommended book - **Agile Retrospectives** by Diana Larsen/Esther Derby

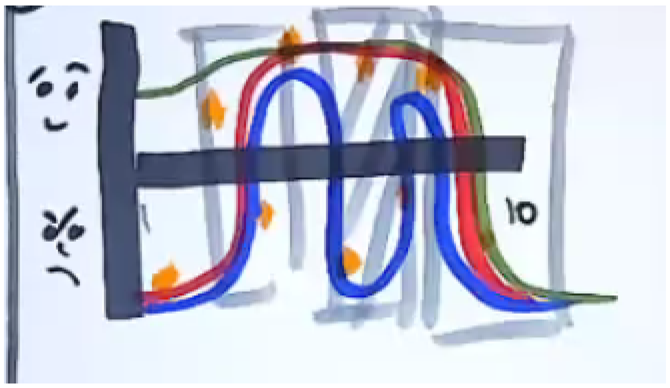
Agile Retrospectives Overview

It's impossible to cover all these 5 steps in 60 minutes. GMo takes 90 minutes to do it.

1. Setup - Setting the scene.
    a. Need to decide who gets to participate
    b. How much time do we allocate
    c. Tools we use (Miro,..) and do they require training in these tools
    d. Decide - Generic retrospective or targeted retrospective
    e. Keeping them private
        i. Pros - Safety. It will not impact their careers,...
        ii. Cons - Other teams will not be able to learn from our experiences.
2. Data Collection (25 - 30% of the time) - Asking participants to look back and recall what happened, and what transpired. It's brainstorming.
    a. Ask the team to rate the quality (from 0-10) - capture and visualize the results

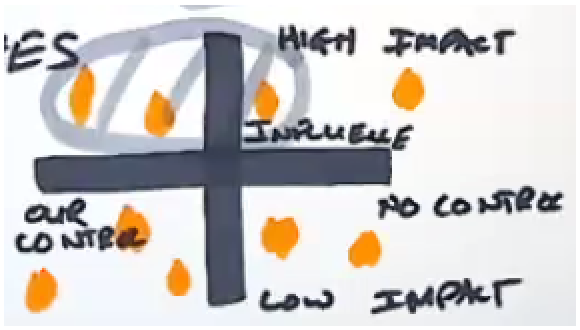


    b. Happiness level
        i. Better than last sprint
        ii. similar to last sprint
        iii. worse than last sprint
    c. Experience - Xaxis being sprint days, Yaxis being goods and bads. Map every memebr’s comments and draw how it formed
        i. what were the good experiences
        ii. what were the bad experiences (server went down,...)



d. Sailboat drawing
   a. boat - sprint
   b. horizon - sprint goal
   c. anchor - what brought us down
   d. motor - what helped us through
   e. hurdles - what all did we have to navigate around

3. Synthesis (30 - 35% of the time) - tackling the root cause by analyzing the data acquired.
   a. Recognizing anomalies vs systemic issues -things that might never happen again vs which are happening frequently during the sprints
   b. Recognizing impact vs control



4. Action items (30 - 35% of the time)
   a. identify action items that are most likely to be implemented. For example - the team found a new way to test which might help
      i. Cannot overhaul because the risk of failure is high
      ii. However, an iterative approach might be better to address and also safe
5. Closeout (5-10% of the time) - Mini retro of the retro. CSM getting feedback
   a. Feedback
      i. did we like the tools we used
      ii. should we invite someone else for our
      iii. what in this retrospect do you like vs dislike
   b. High Note Celebration
      i. Sharing of mutual appreciations

There might be a plateau in improvement. Because

→ Getting from Worse to Bad is super easy

→ Getting from Bad to Average is easy

→ Getting from Average to Good is tough

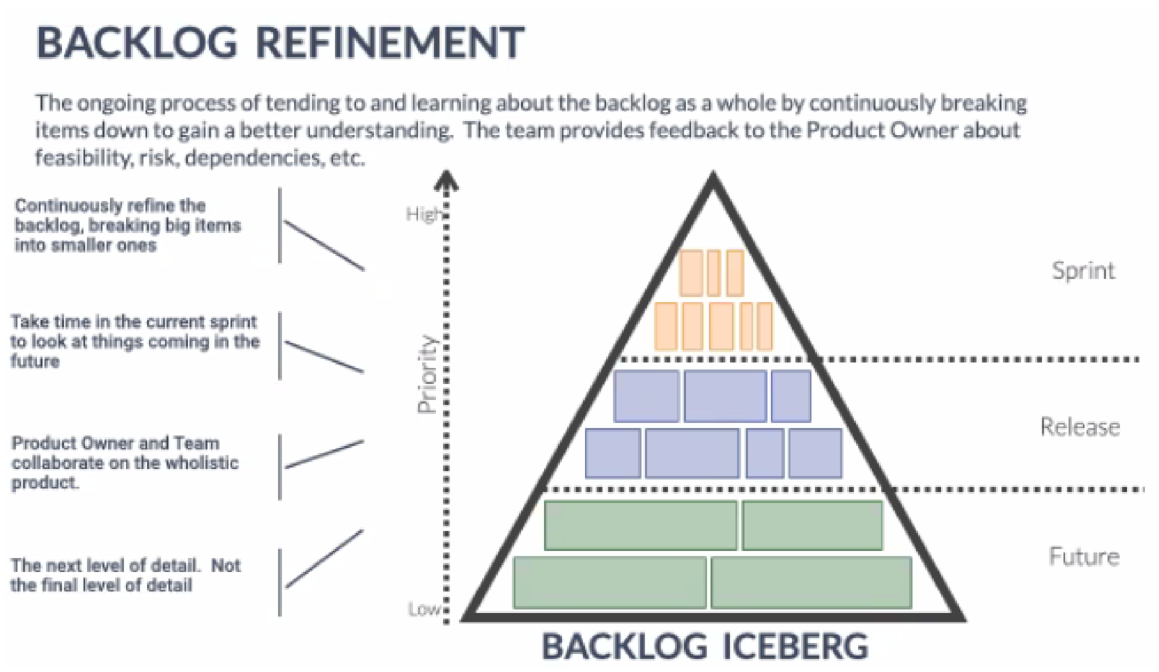
→ Getting from Good to Great is very difficult

Your team might have peaked efficiency and increasing the efficiency even by 1% might take a while. Updating the retrospective approach might help achieve this (but it will still take time).

Backlog Refinement

Product backlog items. They are broken down enough so that they can fit into 2-week sprints. And, you don't break down all the items, we break down the items that are on the top of the list.

Backlog refinement - need to happen on a continuous basis. Can be - weekly, monthly. Developers spend 10% of their time in backlog refinement. The team should focus on the current sprint but should have a sight of where we are headed which is achieved by backlog refinement.

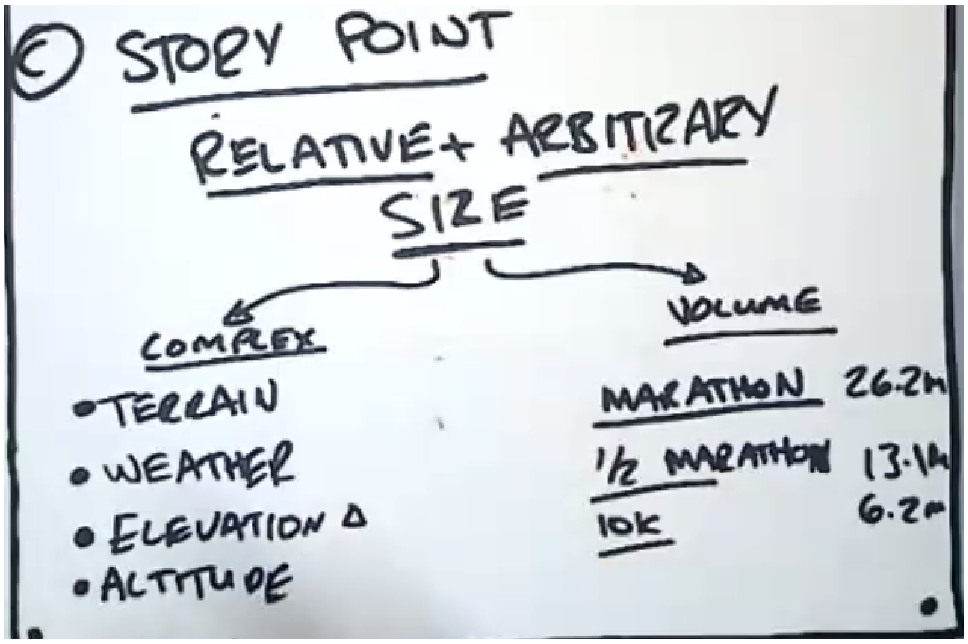

BACKLOG REFINEMENT
The ongoing process of tending to and learning about the backlog as a whole by continuously breaking items down to gain a better understanding. The team provides feedback to the Product Owner about feasibility, risk, dependencies, etc.
Continuously refine the backlog, breaking big items into smaller ones
Take time in the current sprint to look at things coming in the future
Product Owner and Team collaborate on the wholistic product.
The next level of detail. Not the final level of detail
High
Priority
Low
Sprint
Release
Future
BACKLOG ICEBERG

Backlog refinement - priority and order

Estimation

As part of Backlog refinement, estimations happen. Scrum guide does not have an opinion on how to do it. There is no prescription. There are different ways to do it. Estimates are really important in any organization, very few are good at it. Estimation is not just coming up with a number, it's more of a facilitation for the entire team to come to something that everyone agrees on.

1. Tshirt estimates - XS, S, M, L, XL (ex: as long as we agree M is bigger than S and smaller than L. Ms can be different sizes)
    a. Downsides
        i. does not give the granularity
2. Time-based estimates
    a. Ideal time (effort) - how long will something take you to work on if you have everything you need and you have nothing else to work on other than this (without interruption) (ex: NFL is 60 minutes - 4 15-minute quarters)
    b. Elapsed time (duration) - from the time you start to end, how long did it take. (ex: NFL takes up to 3 hours - including halftime, stoppages, commercials,...)
3. Story point - A relative and arbitrary unit of size. A unit is a meaningless entity. It's purposely meaningless. The number of units however represents the size of the work. Point estimates are low accuracy estimate and are not high accuracy estimates. Factors that determine the size are
    a. Complexity
    b. Volume



Running half a mile on gravel, while it's raining, uphill is more difficult than running a full marathon on a paved path in perfect weather downhill.

When comparing the work, need to compare the volume and complexity.

Next, once the volume and complexity are confirmed, person 1 can run that marathon faster than person 2. The size of the marathon is the same, however, different runners run at a different pace.

As the team gets more expertise, they will get faster. If they lose someone, they might get slower. If estimating based on time, have to go back frequently to update the estimates. No one has the patience to keep updating the product backlog.

A point estimate is relative. You estimate something based on previous items and by comparing. No story can be estimated on its own.

Scale

a. Fibonacci - 1 3 5 8 13...

b. Regular - 1 2 4 8 16...

> 💡 If starting brand new and there is nothing to compare and create points. Would start with the tShirt estimate, if its medium, would sign it 5 points which will give a reference, and the rest of the points per story are assigned based on the reference.

Points are used for

1. Long term forecasting

2. Prioritizing - if something of high point does not provide as much value, POs might push it down the product backlog.

😑 DO NOTS 😑

1. Comparing points to time

Agile & Scrum

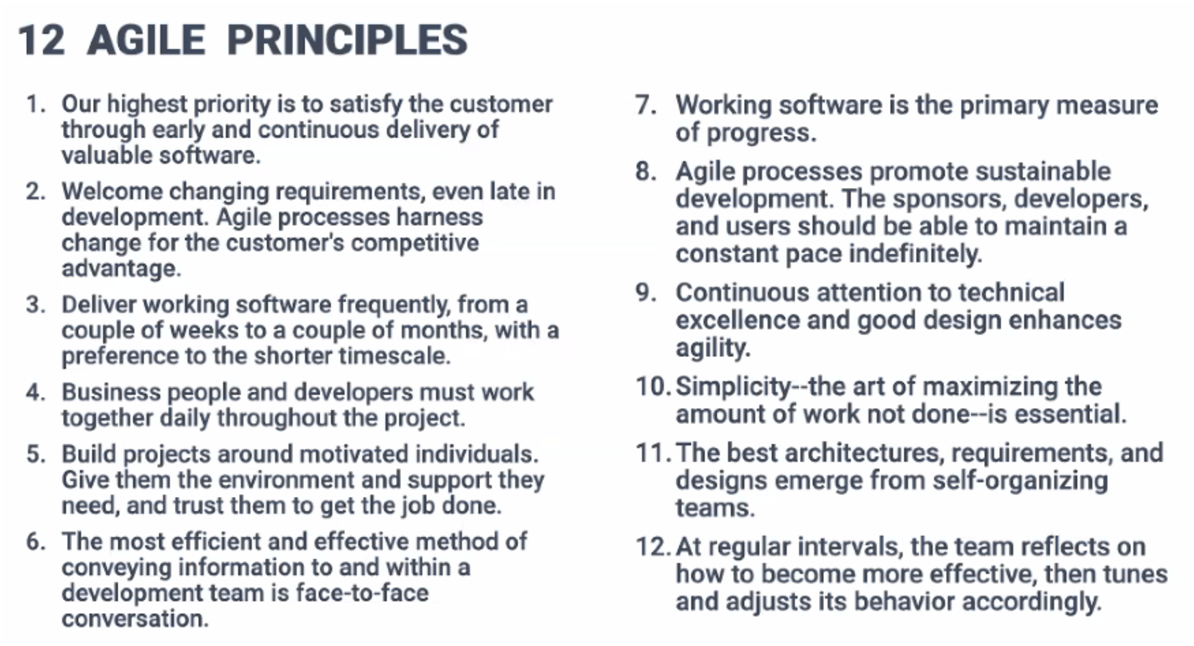
Agile is an umbrella of values and principles underneath which are some frameworks, such as,

- Scrum
- Lean
- Kanban
- Extreme Programming
- Crystal
- Feature Driven Development (FDD)
- DSDM

Agile manifesto

- Individuals and interactions over process and tools
- Working software over documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

12 Agile Principles - Interpretation



12 AGILE PRINCIPLES

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

1. Customer satisfaction through iterative deployment
2. Embrace change to benefit the customer
3. Working product frequently delivered
4. Emphasize Stakeholder and developer collaboration
5. Empower self-organizing teams
6. Communicate face-to-face
7. Measuring progress through working product
8. Its a marathon, not a sprint
9. Technical excellence + Good Design = Better Agility
10. Simplicity - prioritization and batching - making sure to work minimum by working on just necessary stuff

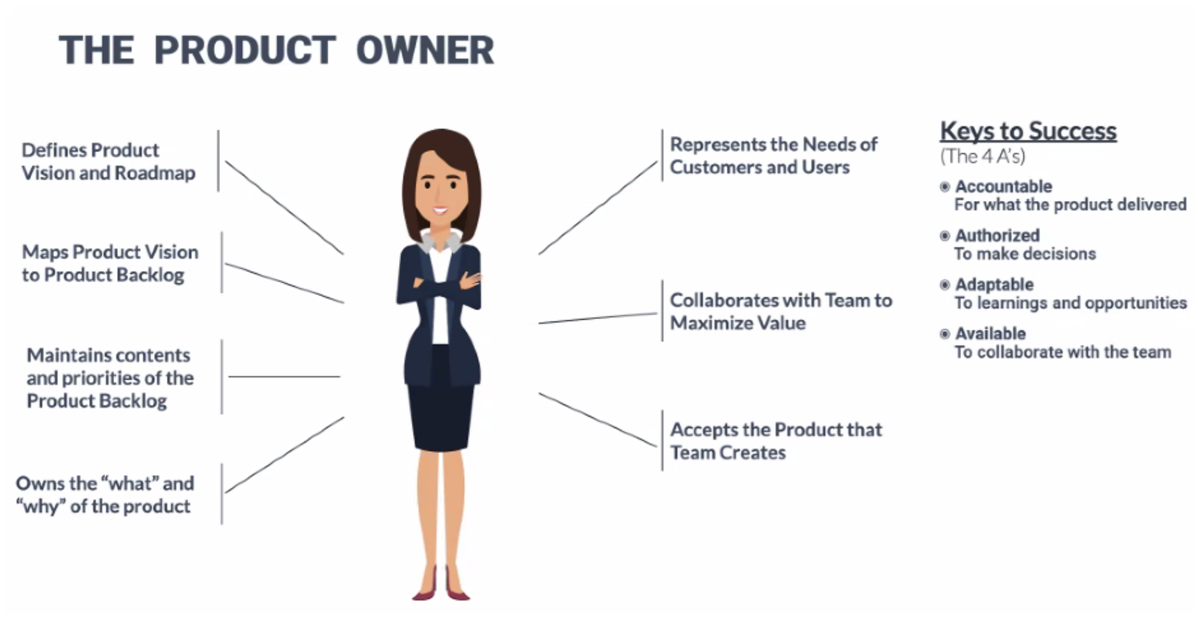
11. No one is handing developers the best architecture. They are coming up on their own through iteration.
12. Adapt, adjust and excel regularly

Scrum values

1. Commitment - commitment towards achieving not finishing, commitment to helping peers,...
2. Focus - by creating small batches (2-week sprints), by having an agreed sprint goal
3. Openness - open kitchen concept exhibiting transparency (stakeholders or other teams can get involved)
4. Respect - respect towards peers (talking 1-1 instead of public shaming or tattletale, respective developer decision on how much they can take on)
5. Courage - doing the right thing, to say no to unreasonable demand, to say no to cutting corners, to say no to PO squeezing in points

Roles & Responsibilities Summary

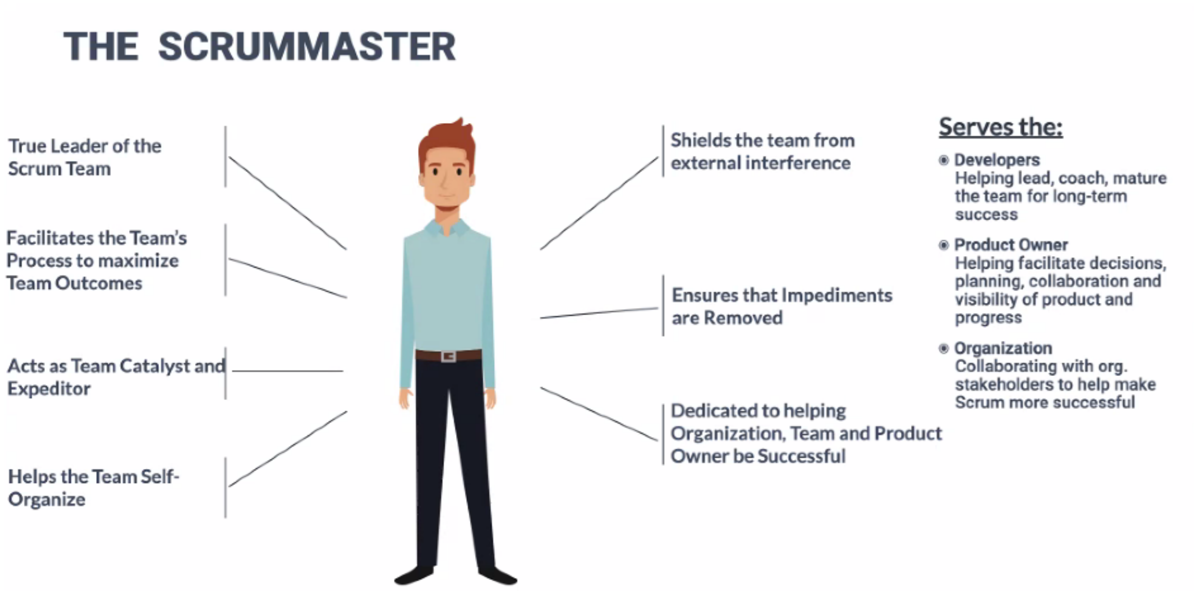
Product owner


THE PRODUCT OWNER
Defines Product Vision and Roadmap
Maps Product Vision to Product Backlog
Maintains contents and priorities of the Product Backlog
Owns the "what" and "why" of the product
Represents the Needs of Customers and Users
Collaborates with Team to Maximize Value
Accepts the Product that Team Creates
Keys to Success
(The 4 A's)
Accountable
For what the product delivered
Authorized
To make decisions
Adaptable
To learnings and opportunities
Available
To collaborate with the team

1. Decides the vision, goals, backlog priority

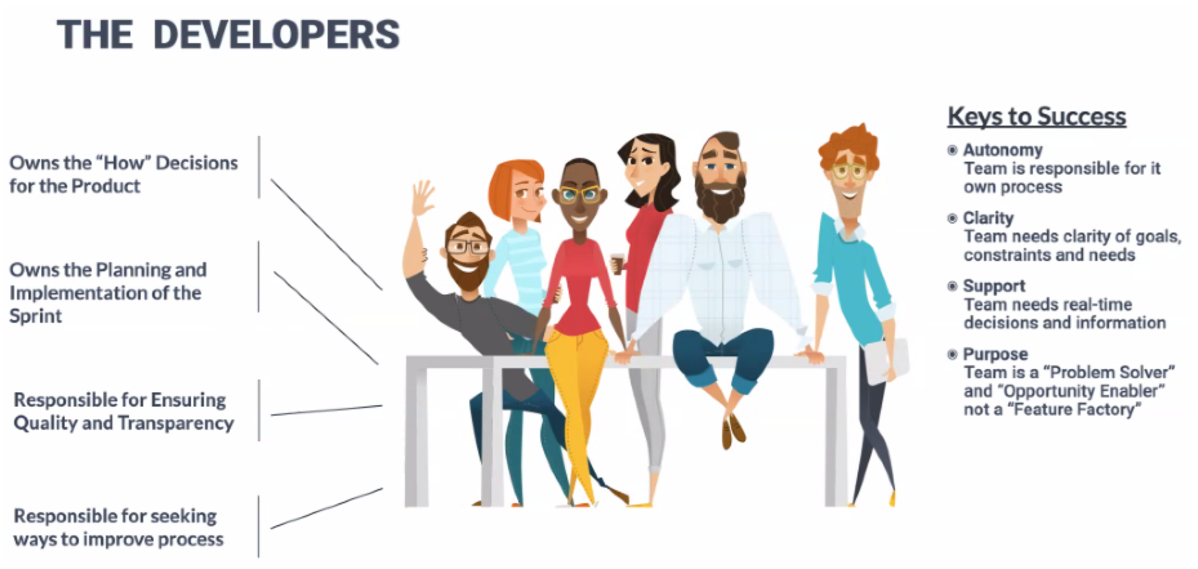
2. Not waiters or drive-throughs where orders are placed and receive the changes right away
3. Not liaison - POs decide what the product is going to be
4. POs decide the investment - bug fixes or enhancements,...

Scrum Master


THE SCRUMMASTER
True Leader of the Scrum Team
Facilitates the Team's Process to maximize Team Outcomes
Acts as Team Catalyst and Expeditor
Helps the Team Self-Organize
Shields the team from external interference
Ensures that Impediments are Removed
Dedicated to helping Organization, Team and Product Owner be Successful
Serves the:
Developers
Helping lead, coach, mature the team for long-term success
Product Owner
Helping facilitate decisions, planning, collaboration and visibility of product and progress
Organization
Collaborating with org. stakeholders to help make Scrum more successful

1. Service to POs, Developers, Organization
    a. Help POs in facilitation and interaction with the teams and prioritization
    b. Coaching and motivating scrum team
    c. Organizational change agents - collaborating with SMs of other teams and changing the org standards to alleviate the policies which will benefit all future teams forever.
    d. Help resolve impediments
2. Prioritizing the Team over PO over Organization.

Developers (product developers)



THE DEVELOPERS

Owns the "How" Decisions for the Product

Owns the Planning and Implementation of the Sprint

Responsible for Ensuring Quality and Transparency

Responsible for seeking ways to improve process

Keys to Success

- **Autonomy**
  Team is responsible for it own process
- **Clarity**
  Team needs clarity of goals, constraints and needs
- **Support**
  Team needs real-time decisions and information
- **Purpose**
  Team is a "Problem Solver" and "Opportunity Enabler" not a "Feature Factory"

1. Developers run Daily scrums
2. Owns the "how" decisions for the product

💡 Only 3 Artifacts in the scrum - Product backlog, Sprint backlog & The Increment