22CSC51 - AGILE METHODOLOGIES

Prepared By,

Mr.M.Muthuraja,

Assistant Professor

Department.of CSE

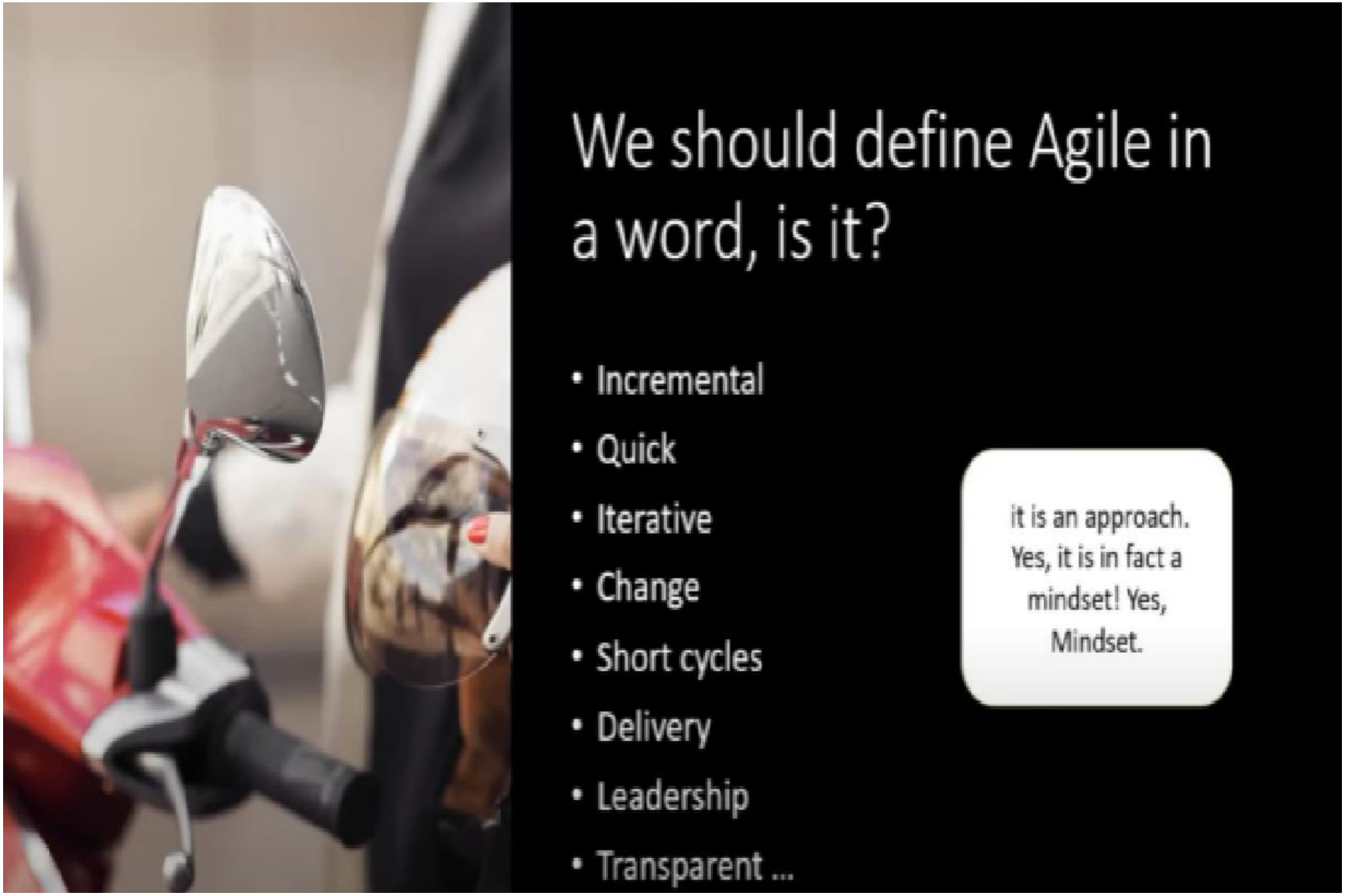
Kongu Engineering College

Unit - II

Agile Principles and Scrum

First, we shall learn
what is AGILE...

Agile?


We should define Agile in a word, is it?
• Incremental
• Quick
• Iterative
• Change
• Short cycles
• Delivery
• Leadership
• Transparent ...
it is an approach. Yes, it is in fact a mindset! Yes, Mindset.

History of Agile


Kent Beck


Mike Beedle


Arie van Bennekum


Alistair Cockburn


Ward Cunningham


Martin Fowler


James Grenning


Jim Highsmith


Andrew Hunt


Ron Jeffries


Jon Kern


Brian Marick

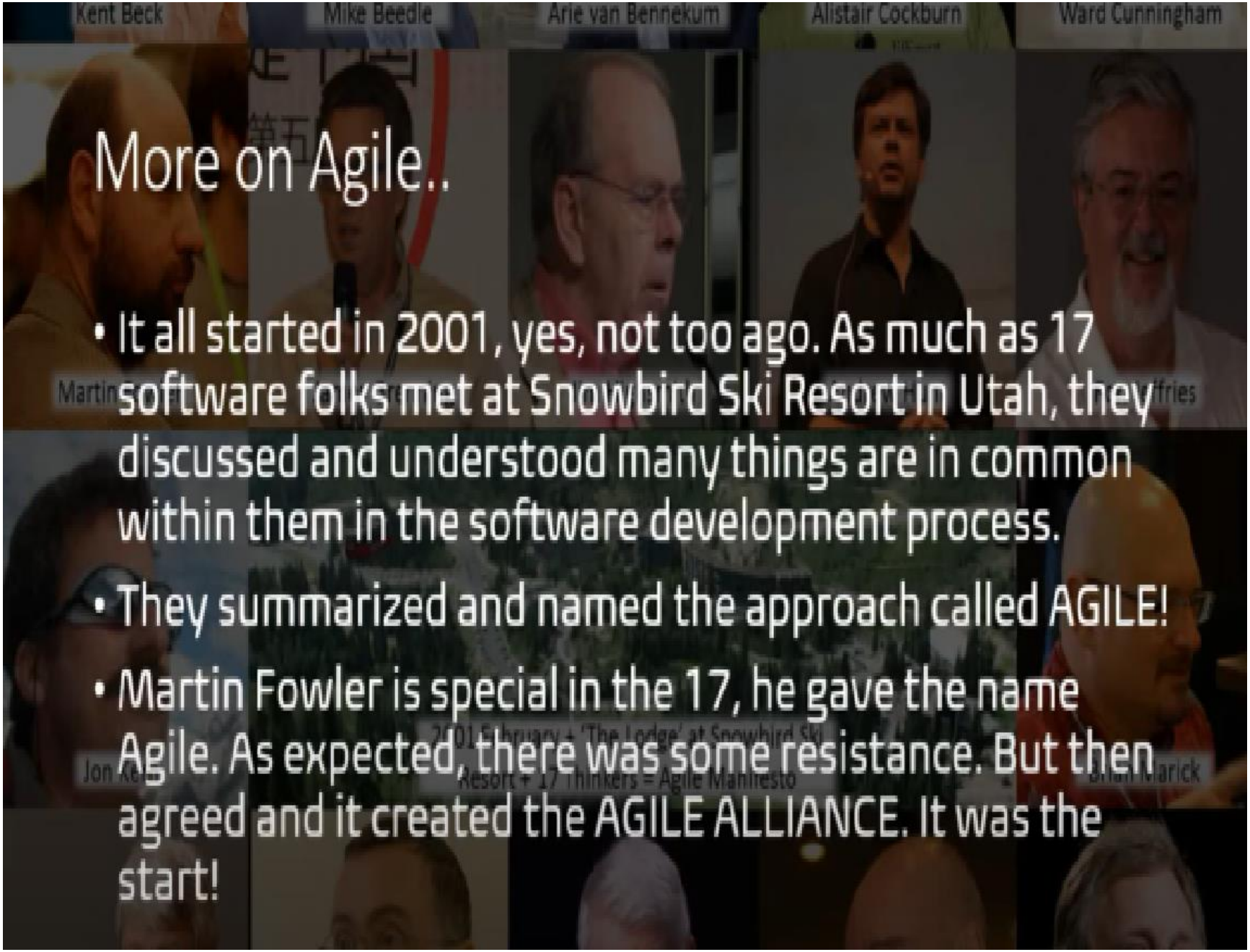

Bob Martin


Stephen Mellor


Dave Thomas


Jeff Sutherland


Ken Schwaber



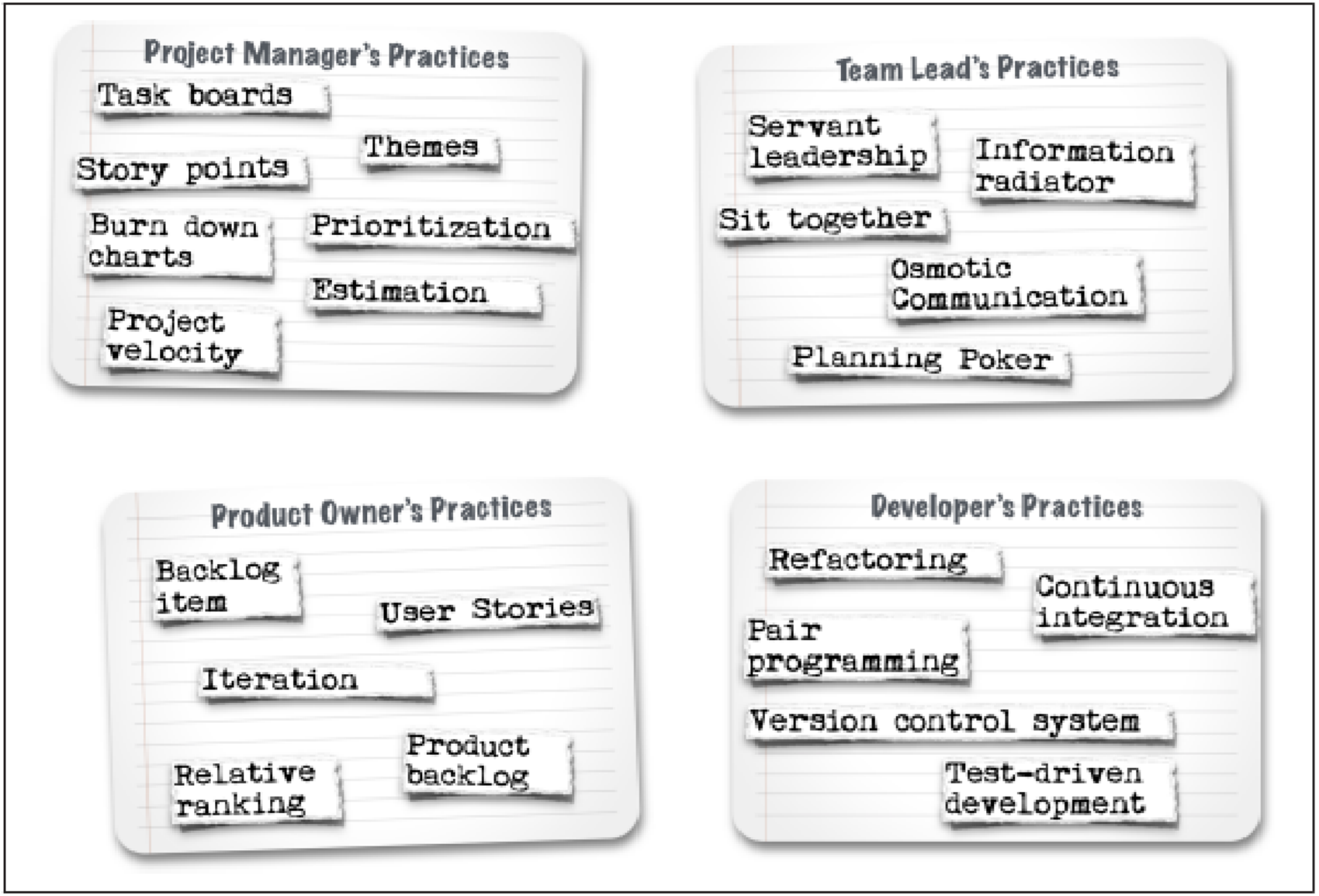
More on Agile..

- It all started in 2001, yes, not too ago. As much as 17 software folks met at Snowbird Ski Resort in Utah, they discussed and understood many things are in common within them in the software development process.
- They summarized and named the approach called AGILE!
- Martin Fowler is special in the 17, he gave the name Agile. As expected, there was some resistance. But then agreed and it created the AGILE ALLIANCE. It was the start!

Agile

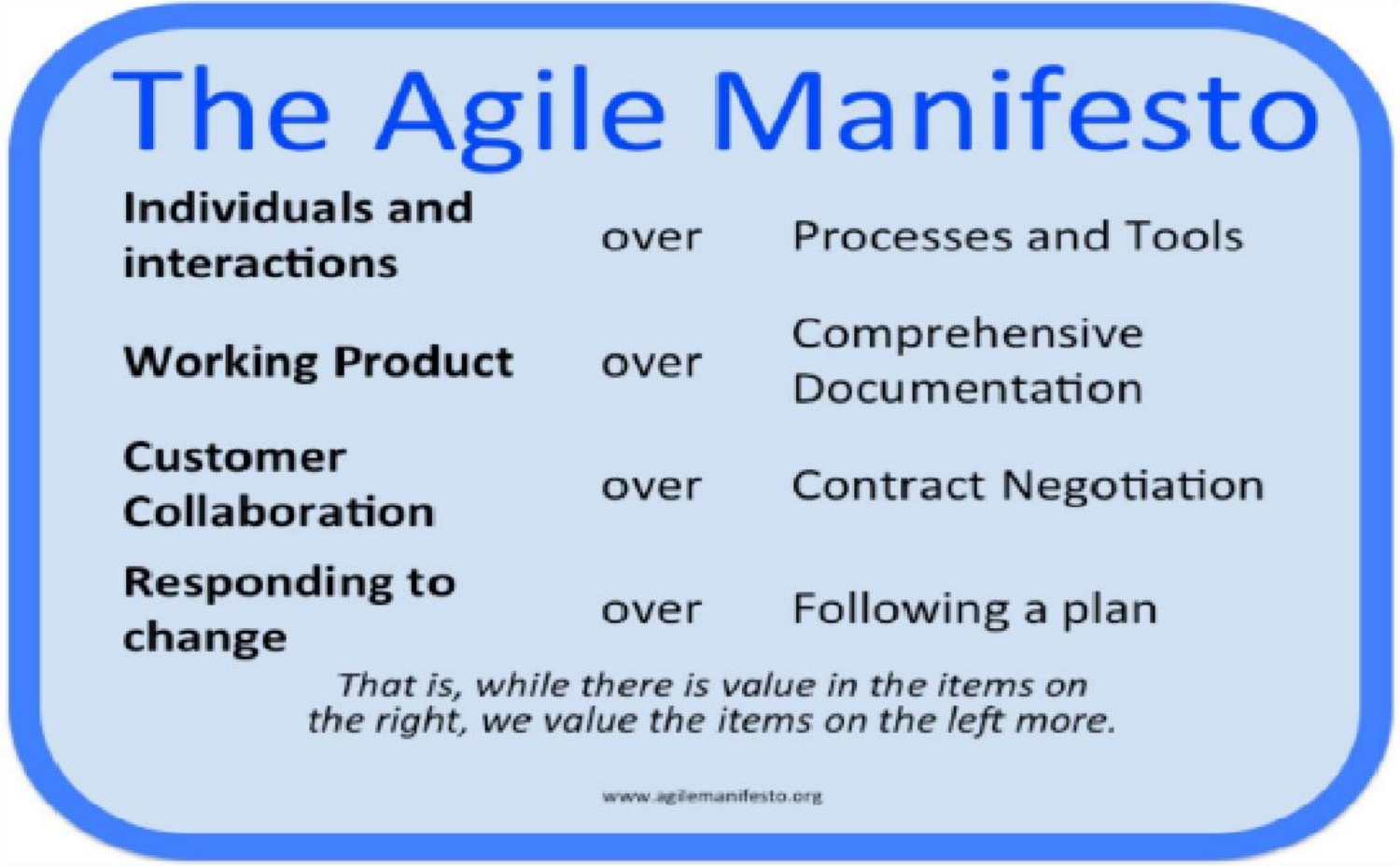
- Agile is a set of **methods and methodologies** that help your team to **think more effectively**, work more efficiently, and make better decisions.
- These **methods and methodologies address** all of the areas of traditional software engineering, including project management, software design and architecture, and process improvement.
- Agile is also a **mindset**, because the right mindset can make a big difference in how **effectively a team** uses the practices.

Agile


Project Manager's Practices
Task boards
Themes
Story points
Burn down charts
Prioritization
Estimation
Project velocity
Team Lead's Practices
Servant leadership
Information radiator
Sit together
Osmotic Communication
Planning Poker
Product Owner's Practices
Backlog item
User Stories
Iteration
Product backlog
Relative ranking
Developer's Practices
Refactoring
Continuous integration
Pair programming
Version control system
Test-driven development

Understanding Agile Values

Four Values



Agile Manifesto: The 4 Values

Individuals & Interactions

- Communication is a key reason for project's success!
- People build software products.
- Teams self-organize – to identify scope; negotiate; accept; define; collaborate; share; follow discipline; solve problems.
- Motivate individuals and foster interactions among team members; with customers; stakeholders. High-context interactions.
- Focus on people - they are the ultimate source of energy!

Working Software

- Primary goal is to create & deliver value! Satisfy/delight customer.
- Deliver frequently to give business advantage to the customer.
- Do documentation that adds value!
- Customer Value / Priority focus.

Customer Collaboration

- Be flexible and cooperative on customer's dynamic needs.
- Work with the customer to deliver the intent of the Contract.
- Closely collaborate & follow customer's product vision!.
- Have flexible contracting models. Maintain relationships.

Responding to Change

- Change is a reality; reality is the worst enemy of a Plan!
- Changes in customer's business environment impact our plans
- Embrace change by adaptive planning. Reflect & Improve.

Individuals and Interactions Over Processes and Tools

- People can go wrong when they **blindly follow a process**.
- A great tool can sometimes help you do the wrong thing faster.
- The software world is **full of great practices**: not all of them are appropriate for every project and every situation.
- However, it's universally important to understand the people on the team, **how they work together**, and how each person's work impacts everyone else.
- This is an especially useful idea for anyone who needs to improve the way his or her team works.
- That's why agile teams **value individuals and interactions over processes or tools**: because it's not enough to have the "correct" process or "best" practice.

Working Software Over Comprehensive Documentation

- There are binders full of **complete and comprehensive software** documentation sitting unopened on shelves all over the world.
- Agile teams value working software over comprehensive documentation.
- But the term **“working software”** may seem vague—after all, what does the word “working” really mean? To an agile practitioner, **working software is software that adds value to the organization**.
- Valuing working software over comprehensive documentation does not mean that you should not document.
- There are many kinds of **documents that are very useful for the team**.
- Concentrating on working software, on the other hand, is a great way to make sure that the team keeps on track.

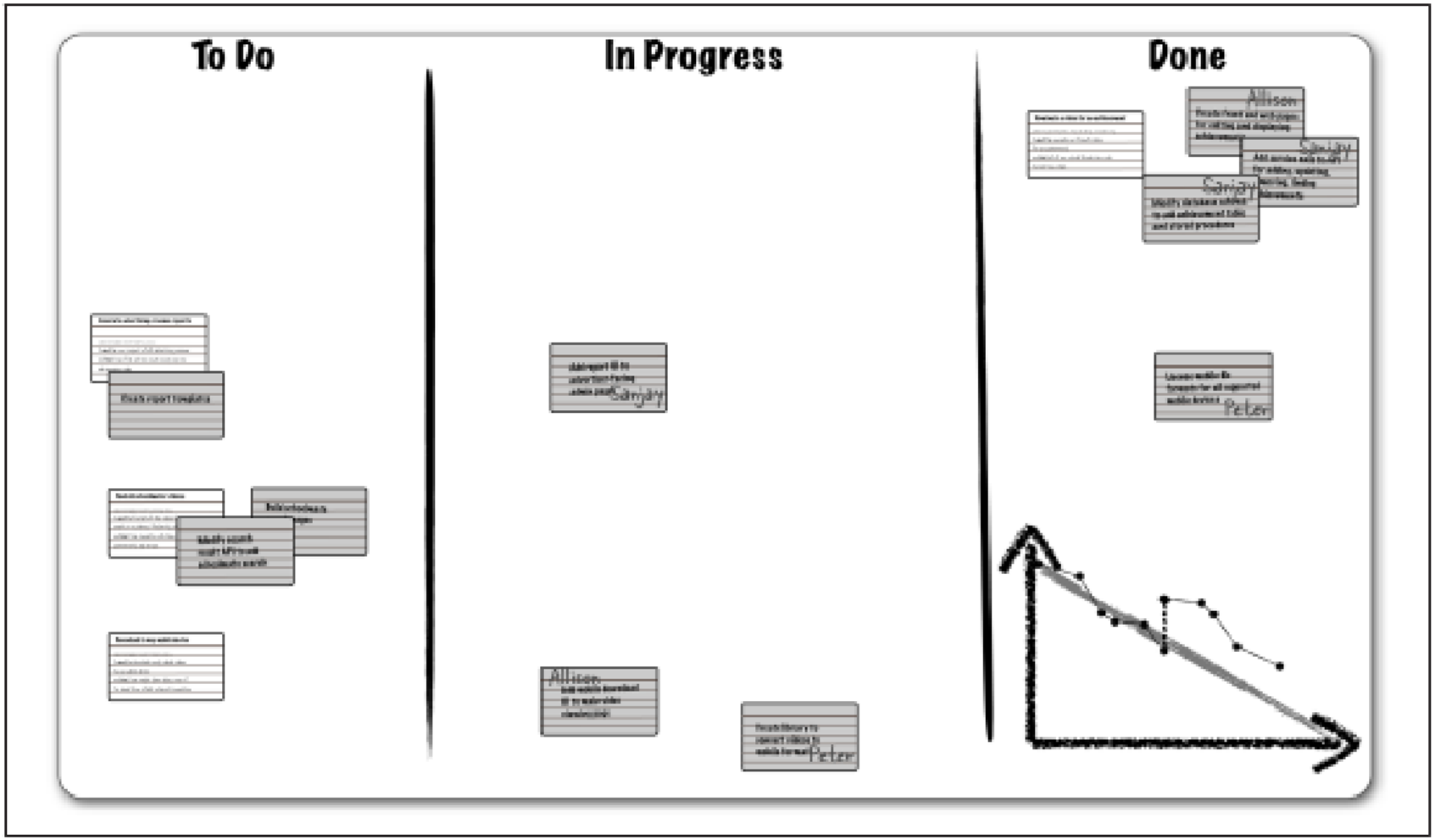
Customer Collaboration Over Contract Negotiation

- A lot of people might read **“contract negotiation”** and assume that this value only applies to consultants and contractors who work within a contract.
- Actually, this applies to many teams that work within a single company.
- Product owners will often use **user stories** as a way to collaborate with the rest of the team.

Responding to Change Over Following a Plan

- There's an old project management saying: **"plan the work, work the plan."**
- Unfortunately, if you **work the wrong plan**, you'll build the wrong product.
- That's why **teams need to constantly look for changes,** and to make sure that they respond appropriately when there's a change in what the users need, or in how the software needs to be built.
- If the circumstances change, the project needs a new plan.

Task Board



Agile teams often use task boards to show tasks and track progress. They'll write tasks or user stories on index cards, and move them across the board as they progress. Many teams also draw charts on their task boards to help them track progress.

Agile Elephant

The agile "elephant" is greater than the sum of its practices.

12 Agile Principles

@OlgaHeismann

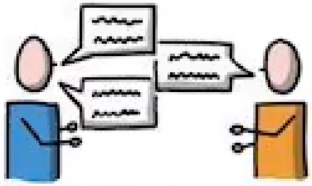
Satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development.

Deliver working software frequently.

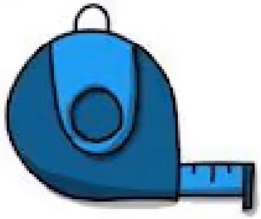
Business people and developers must work together.

Build projects around motivated individuals. Give them the support they need. Trust them.



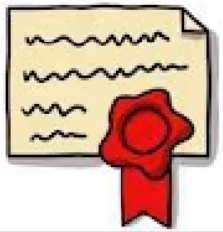
The most efficient and effective method of conveying information is face-to-face conversation.

Working software is the primary measure of progress.



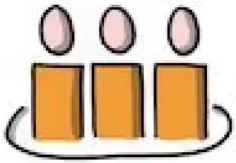
The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design.

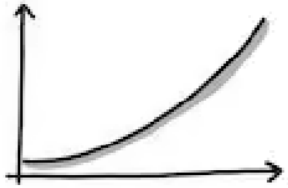


Simplicity—the art of maximizing the amount of work not done—is essential.

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



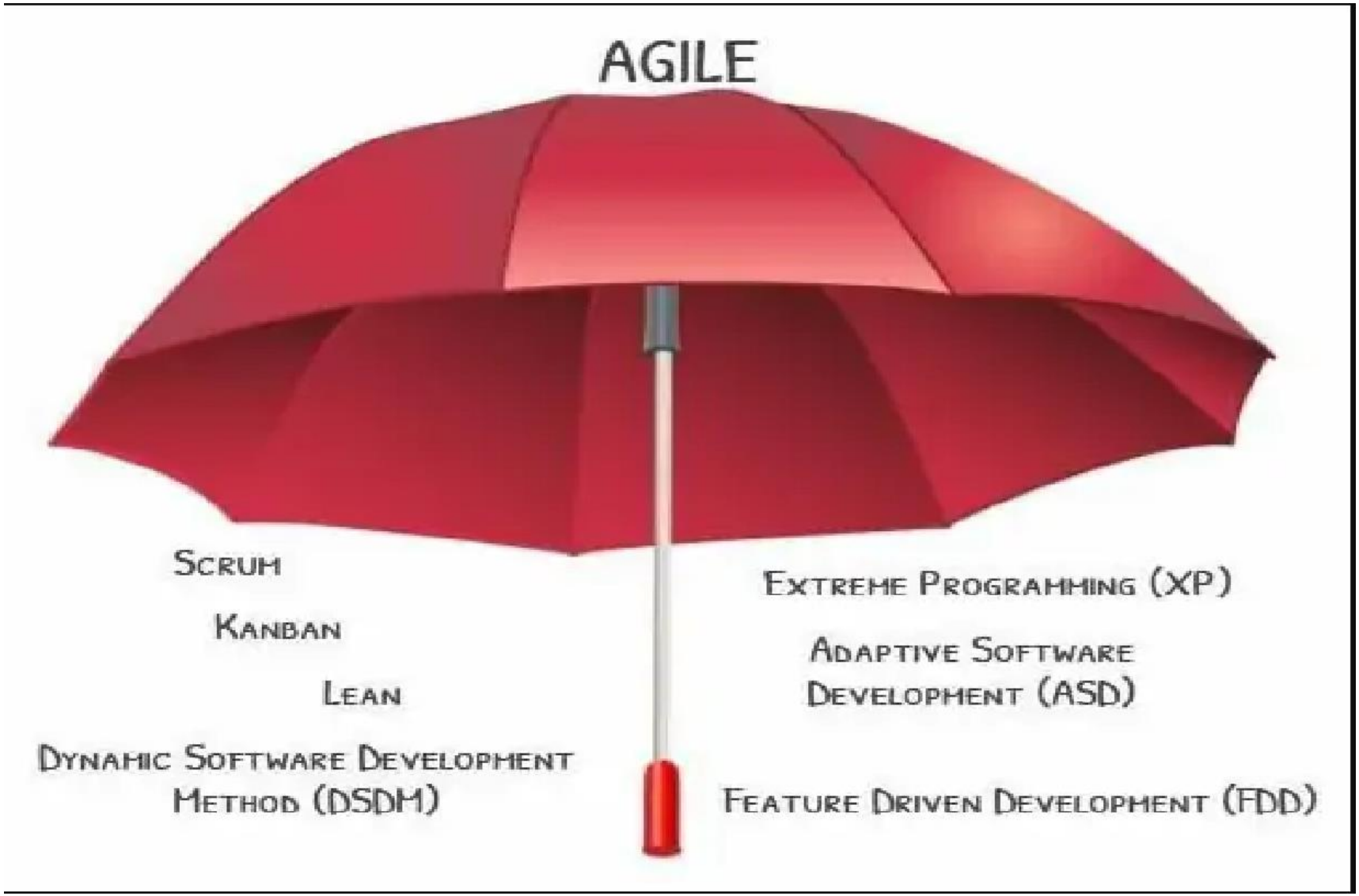
The team reflects on how to become more effective and adjusts its behavior accordingly.



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. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
5. Businesspeople and developers must work together daily throughout the project.
6. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
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.[3]

Agile Umberlla



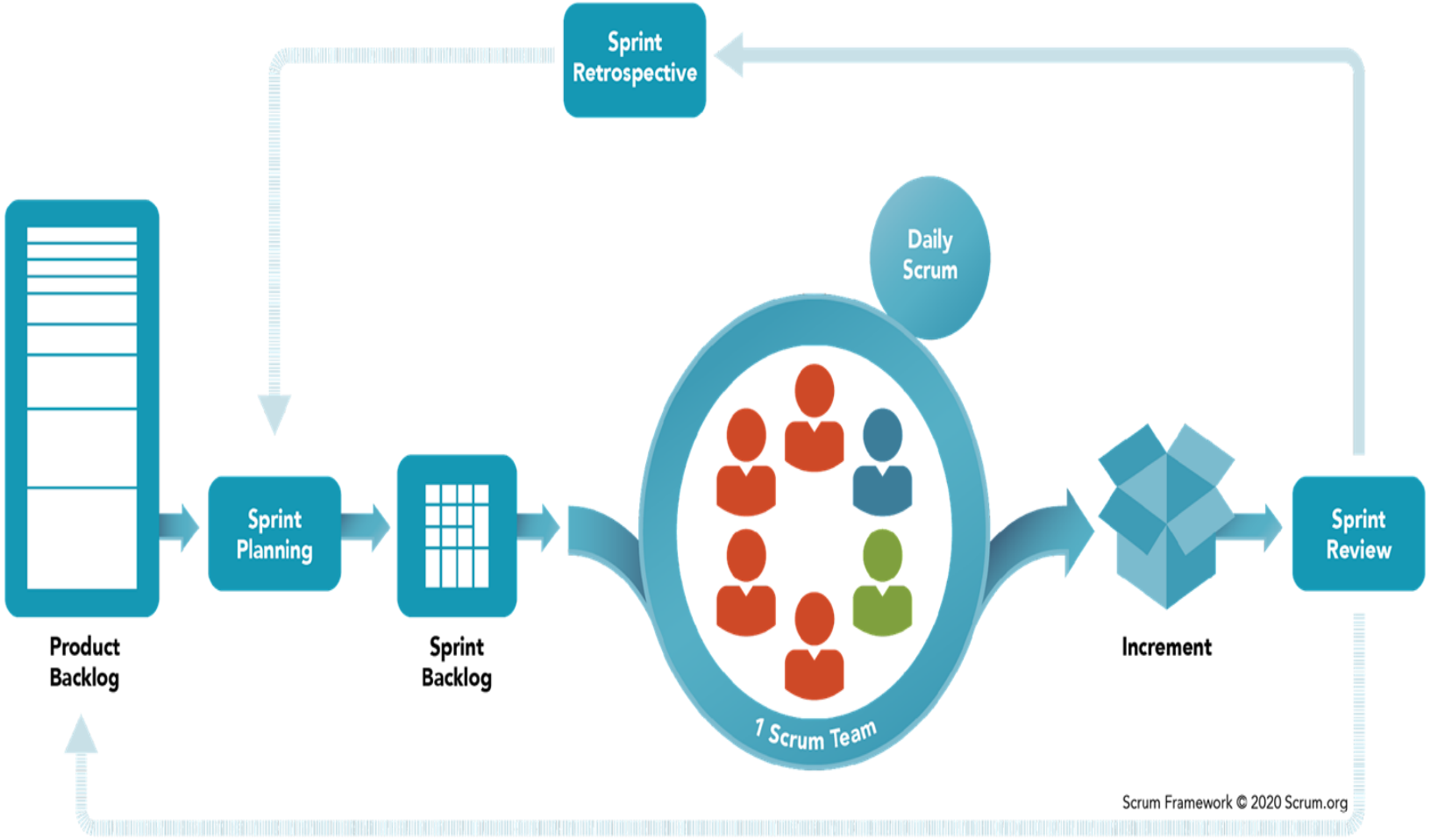
Scrum

- Agile is a mindset and philosophy that describes a set of principles in the Agile Manifesto.
- On the other hand, Scrum is a framework that prescribes roles, events, artifacts, and rules/guidelines to implement that mindset.
- In other words, Agile is the mindset and Scrum is the framework that prescribe a process for implementing the agile philosophy.

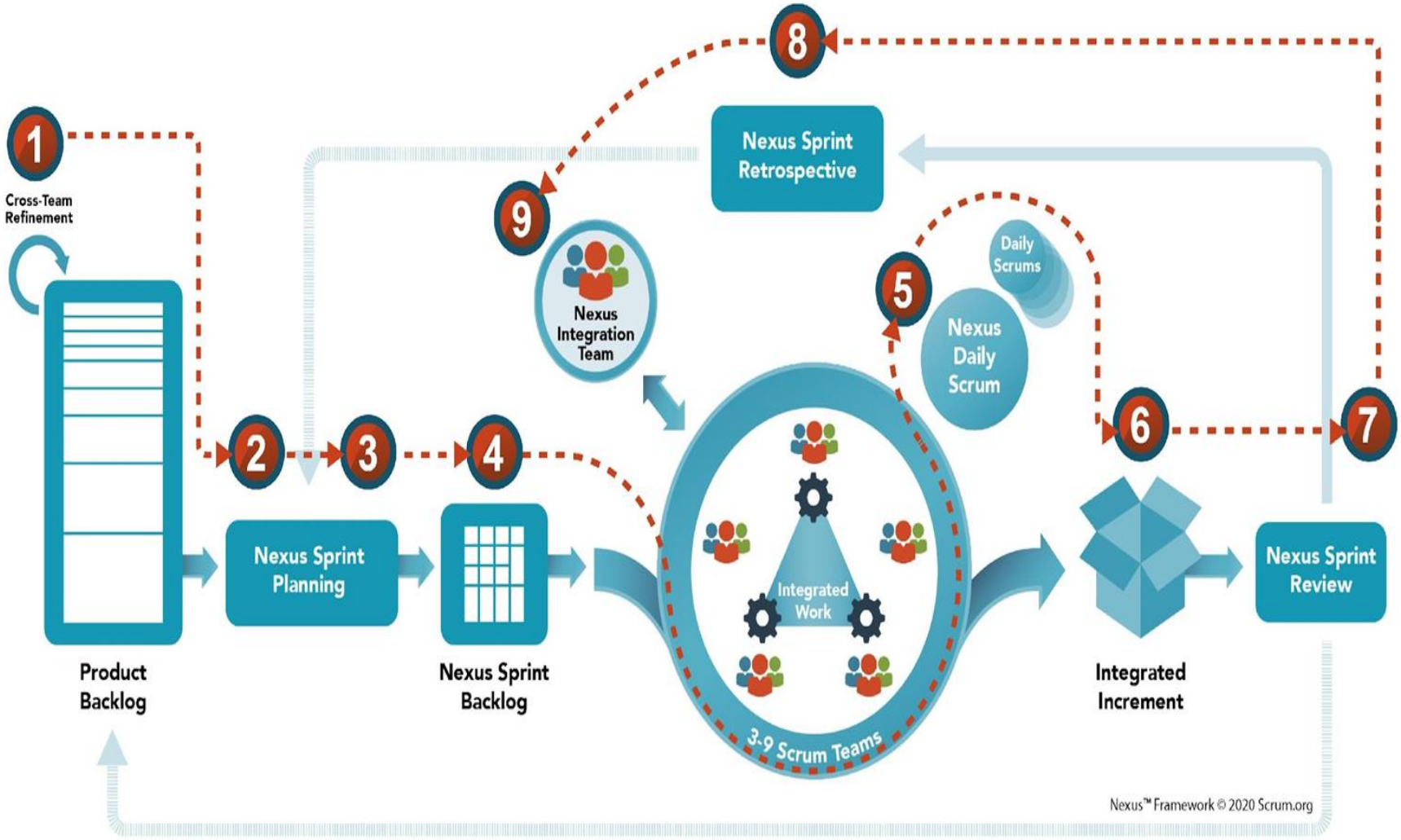
Scrum

- It is a **light weight framework**, which is very easy to learn and adapt, which enables people, teams and the organization at large generate value.
- Remember **scrum and agile are not same**, but scrum is one of the agile processes.

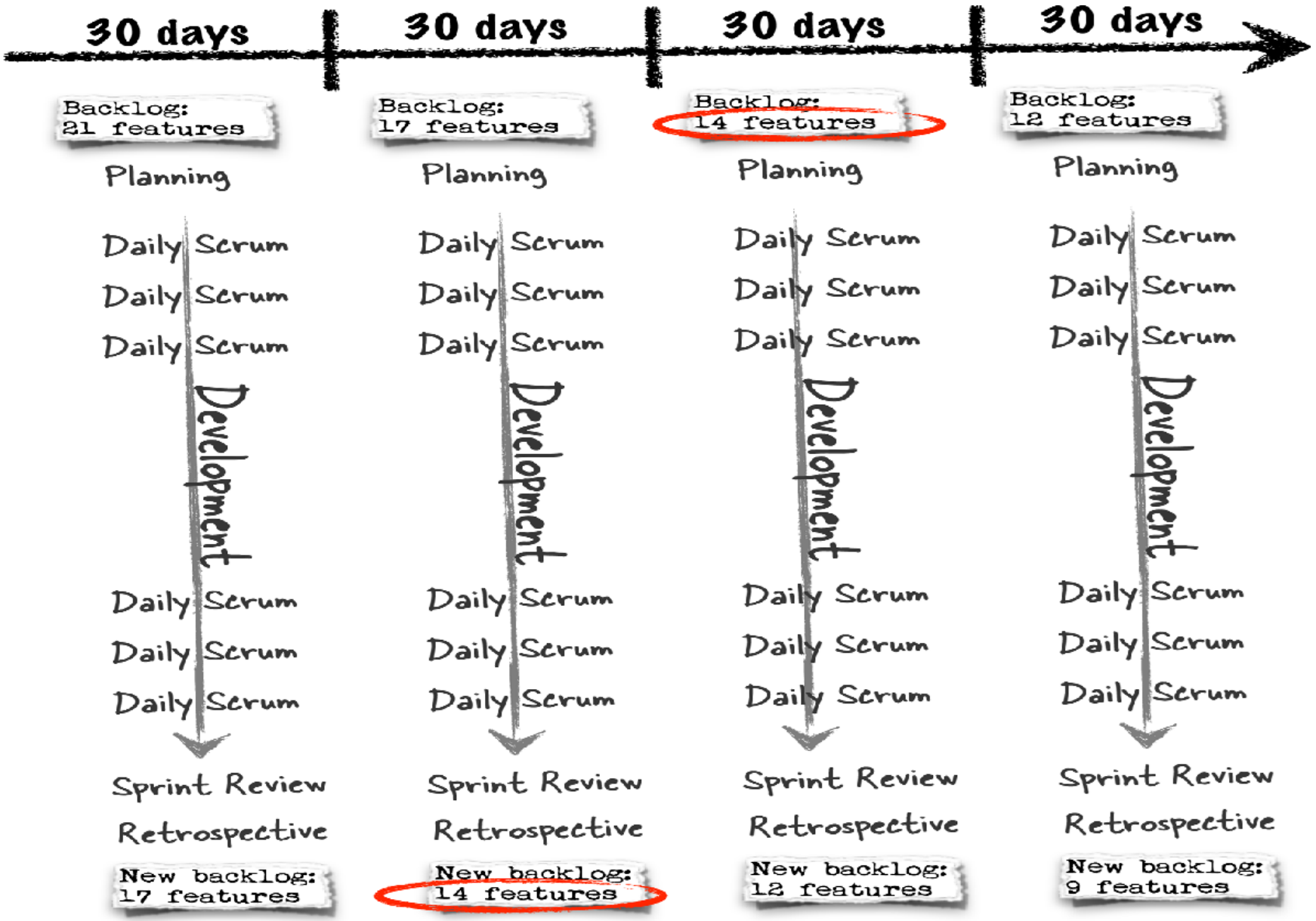
Scrum Process



Scrum Process



Basic Scrum Patterns



Basic Scrum Patterns

There are three main roles on a Scrum project:

- **Product Owner**
- **Development Team or Team Memebers**
- **Scrum Master**

Product Owner:

- **Owns the backlog**, strives to prioritize and takes all the key product decisions.
- This person turns all user requirements into actionable work for the development team

Basic Scrum Patterns

Development Team:

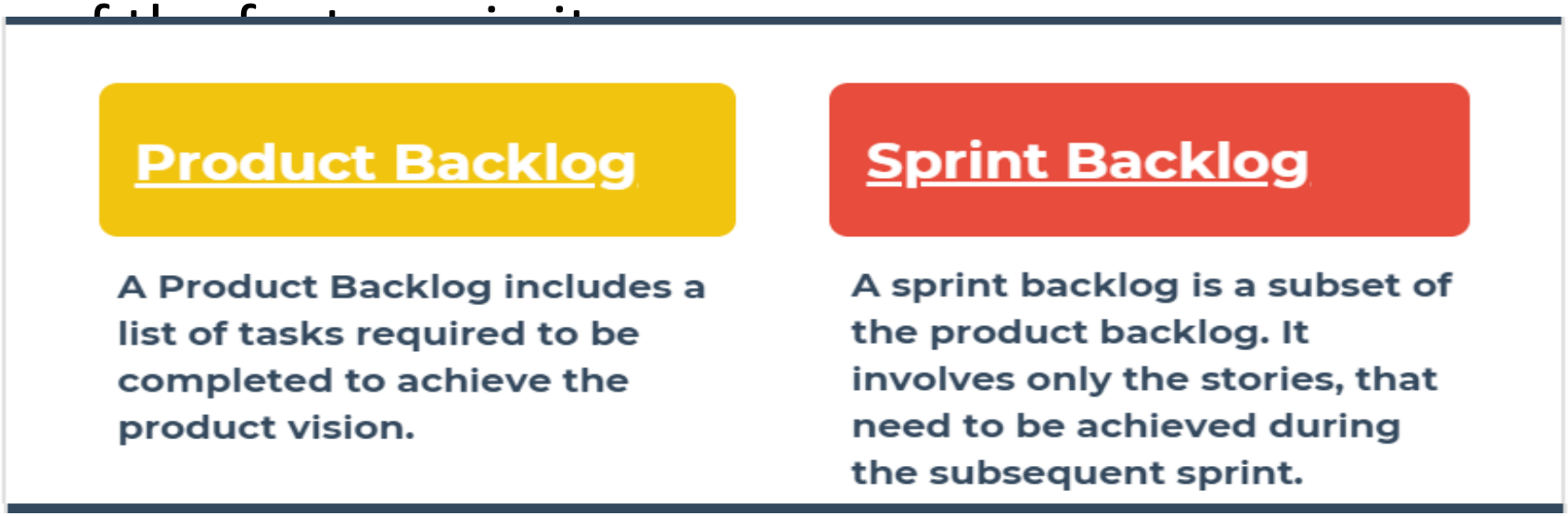
- Cross functional team of **developers, testers, designers, and other technical members** who are needed for the development of the product.

Scrum Master:

- A person who ensures the entire team has everything they need to ship a **user story on time**.
- This person communicates progress and ensures there aren't any roadblocks.

Basic Scrum Patterns

- The software is built using **timeboxed iterations** called **sprints**. At the start of each sprint, the team does sprint planning to determine which features from the backlog they will build. This is called the **sprint backlog**,
- and the team works throughout the sprint to build all



Basic Scrum Patterns

- Every day, the team holds a short face-to-face meeting called the **Daily Scrum** to update each other on the progress they've made, and to discuss the roadblocks ahead.
- Each person answers **three** questions:

✓ **What have I done since the last Daily Scrum?**

✓ **What will I do until the next Daily Scrum?**

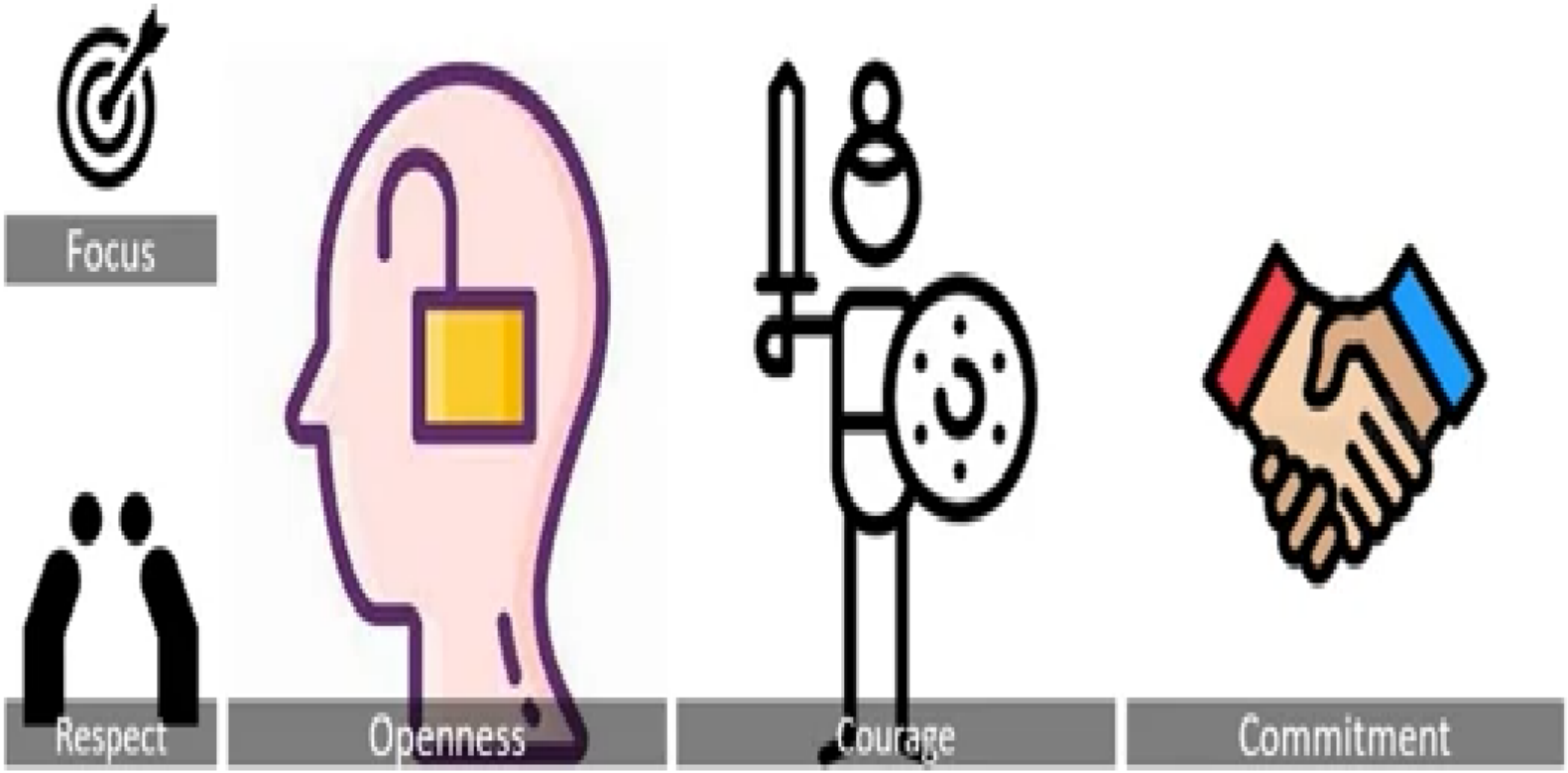
✓ **What roadblocks are in my way?**

- The **Scrum Master**, keeps the project rolling by working with the team to get past roadblocks that they've identified and asked for help with.
- At the end of the sprint, working software is demonstrated to the product owner and stakeholders in the **sprint review,** and the team holds a **retrospective** to figure out lessons they've learned, so they can improve the way they run their sprints and build software in the future.

Target

- ❑ 5 Values — Commitment, Focus, Openness, Respect, and Courage
- ❑ 3 Roles — Scrum Master, Developers and Product Owners.
- ❑ 5 Events — The Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective
- ❑ 3 Artifacts — Product backlog, Sprint backlog and Increments
- ❑ 1 Activity — Sprint Review

FIVE CORE SCRUM VALUES



Scrum Values

- Every company has its **own culture** that includes specific values.
- Self-organizing teams work differently than **command-and-control teams** because they have different values.
- In Agile Project Management with Scrum, Ken Schwaberdiscusses the five Scrum values: **courage, commitment, respect, focus, and openness**.

Scrum Values

- Everyone is **focused on the work**

Focus

The primary and core focus of the scrum team is towards the Sprint and to provide the best possible outcome.

Scrum Values

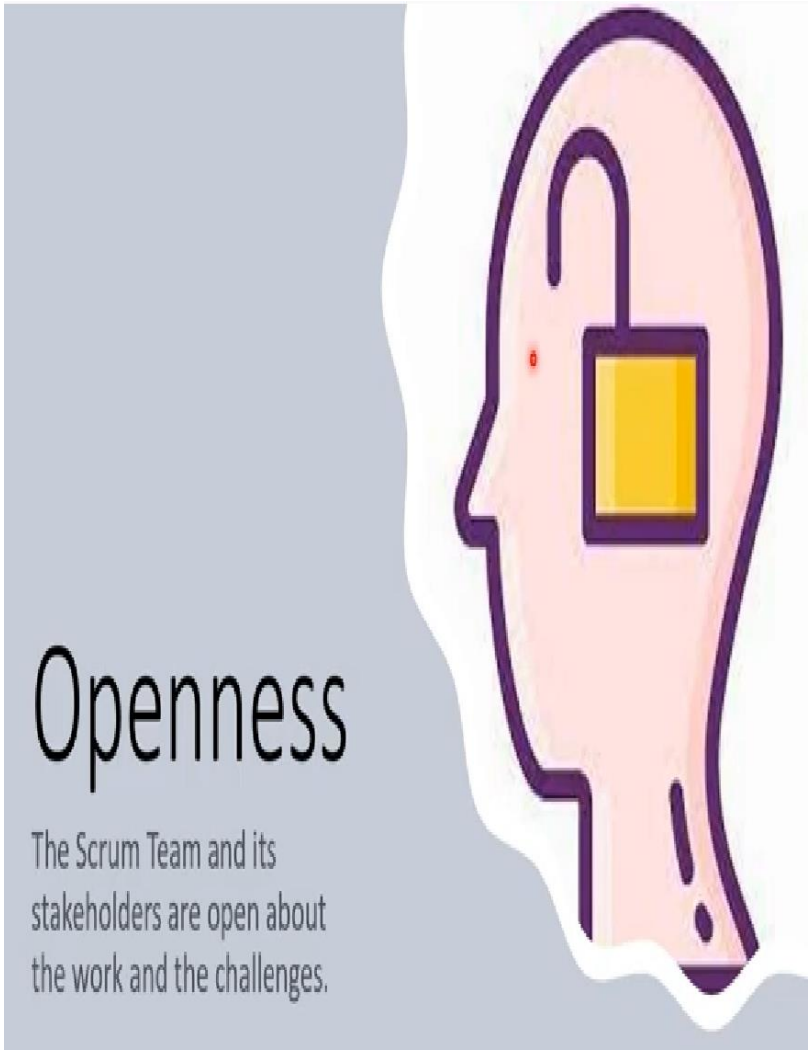
- Each person is **committed** to the project's goals
- That level of commitment can be achieved when the team has the authority to make decisions in order to meet those **goals**, and everyone has a say in how the project is planned and executed.

Commitment

The scrum team members, each and everyone is committed to achieving its goals and to support each other.

Scrum Values

- The teams value **openness**
- When you're working on a Scrum team, everyone else on the team should always be aware of what you're working on and how it moves the project toward its current goals.
- That's why the practices in the basic **Scrum pattern** are aimed at encouraging openness among the team members.
- **Task boards**, for example, allow everyone to see all of the work being done by each team member, and howmuch work is



Scrum Values

- Team members **respect each other**
- When team members have **mutual respect**, they're able to **trust** each other to do a good job with the work they've taken on.
- But that respect doesn't always come easily to programmers and other technical people.
- A good **Scrum Master** finds ways to increase the team members' mutual respect for each other.

Respect

This is very important. Respect each other in the team to be independent, capable.

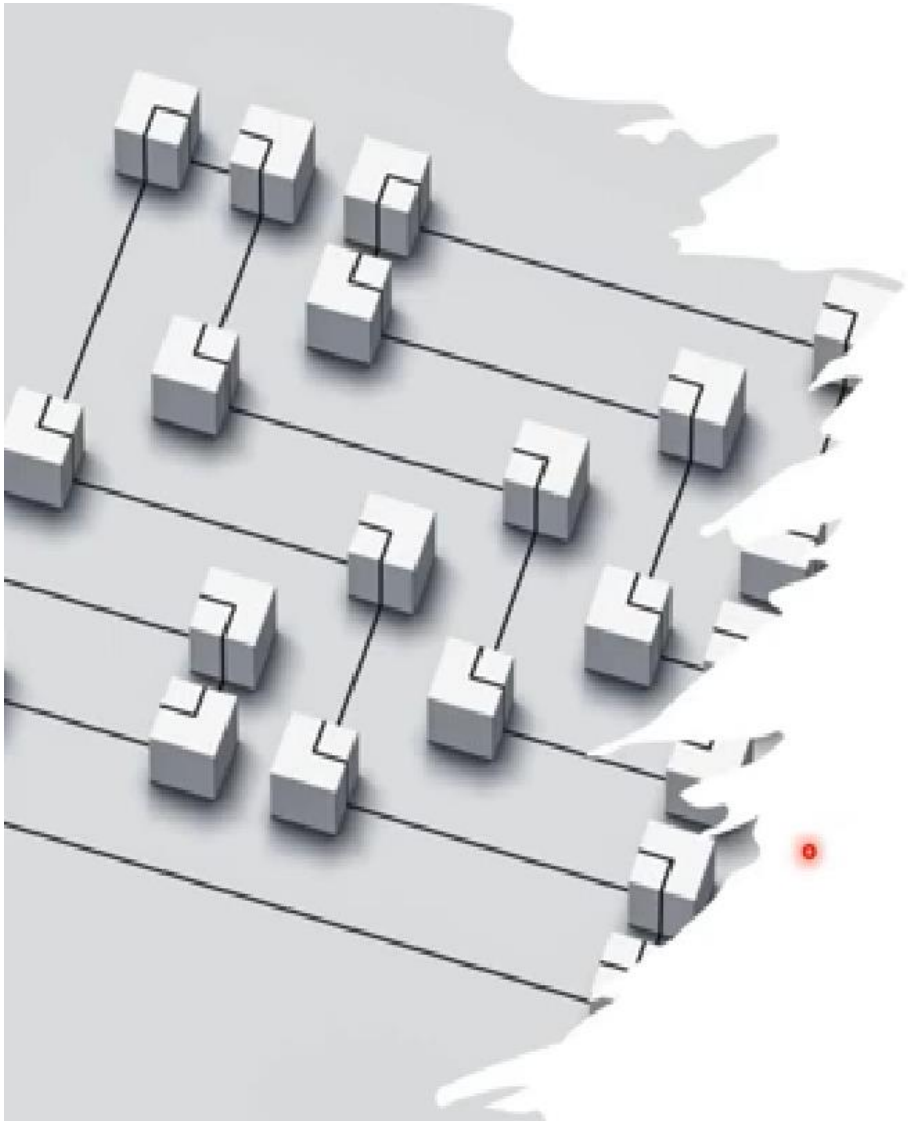
Scrum Values

- Team members have the **courage** to stand up for the project
- Scrum teams have the **courage** to live by values and principles that benefit the project.
- It takes courage to ward off the constant pushback from a company whose values clash with the Scrum and agile values.

Courage

- Well, this is important too.
- The team must have courage to do the right things and to take up challenging problems.

Scrum Roles



The team...

- The Scrum team has three core members – Scrum Master, the Product Owner and Developers. Yes, these three are the pillars.
- One of the very important aspects of the scrum team is (in fact, expectation is) the team being self managing.
- Scrum team shall be successful if they have cross functional members too with required skills.
- As discussed earlier, The primary and core focus of the scrum team is towards the Sprint and to provide the best possible outcome.
- Scrum team is expected to be intelligent enough to know:
  - Who will be doing what?
  - When to do?
  - How to do?

Scrum Roles

How much bigger could be scrum team?

- There is a saying, smaller the better! Scrum is so.
- It is preferred to have 10 or fewer members in the Scrum team who should be the size. Importantly, them team should be capable, efficient and competent to complete tasks.
- Too large teams are not suggested, if it is a case, it is preferred to break them into smaller cohesive scrum teams.
- We shall learn the SCRUM Master in this session followed by the rest in the next.

Scrum Roles

There are three main **roles on a Scrum** project:

- ❖ Product Owner
- ❖ Development Team or Team Memebers
- ❖ Scrum Master

Scrum Roles Product Owner

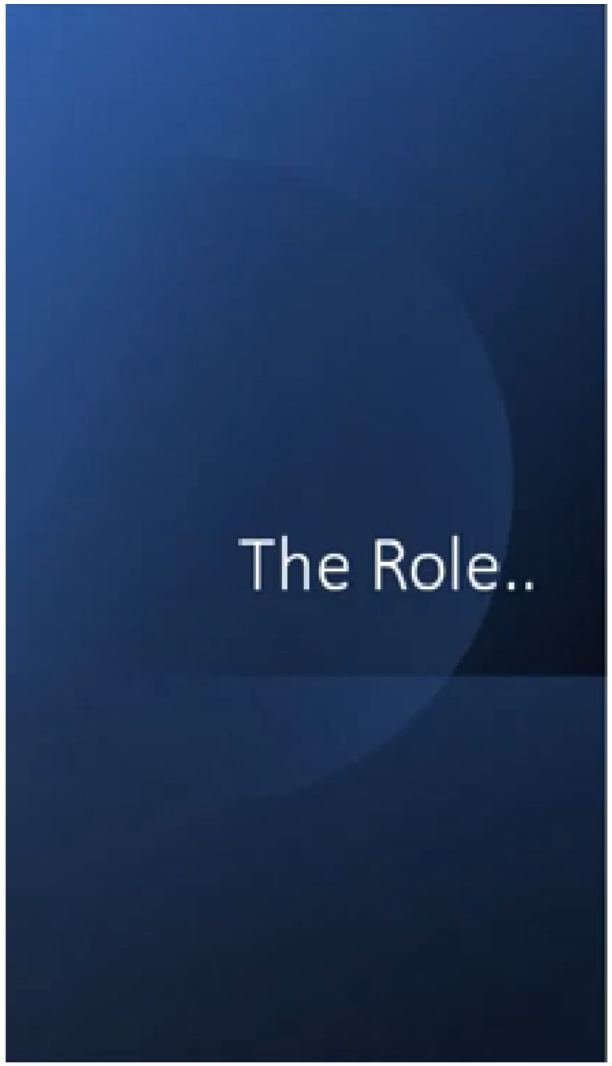
- PO Plays a major role in the cycle. His duty is to maximize the value of the product built by Scrum Team.
- PO is responsible for the PB which include the content, availability and order.
- PO is the guy responsible to keep PB intact and up to date. PO will include – Emerging Requirements, Demand, Customers feedback, all stake holders' feedback etc.
- This is Dynamic and keeps evolving with product. If you have product, you will have PB as well.
- PO is the owner and responsible for Product Backlogs.
  - Developing and communicating the product goal
  - Creating and communication the Product Backlog Items.
  - Ordering the PBI.
  - Making the PB very transparent and visible.

Scrum Roles Product Owner

Scrum Roles Developer

- Developers are the people who WORK ON THE PRODUCT building Increment (the output, i.e., the value)
- Different developers possess different skillsets.
- Developers are responsible for:
  - Creating a plan for the Sprint, the Sprint Backlog;
  - Teaching quality by adhering to a Definition of Done;
  - Adapting their plan each day toward the Sprint Goal; and,
  - Holding each other accountable as professionals.

Scrum Roles Scrum Master



- The Scrum Master is the owner of scrum. SM must establish scrum based on scrum guide.
- SM should enable everyone in the team to understand SCRUM theory, and to practice. (In fact, at the organization level, SM Has a role to play to make Scrum a practice)
- SM is answerable/responsible for the Scrum team's efficacy.
- SM contributes to Scrum Team, PO and the Org in many ways. We shall learn that too..
- **SM and Scrum Team – How Scrum Master Helps the Scrum Team?**
  - He is a coach, yes, he coaches the team on becoming more self managing and cross functional.
  - He facilitates the team towards delivering increments (output) which meets the Definition of Done requirements.
  - He removes the Impediments – Any be it.
  - SM ensures the scrum meetings, events everything take place and does not overshoot the time limits.

Scrum Roles Scrum Master

Scrum Roles Scrum Master

Scrum Artifacts

Why are artifacts important...

Artifacts are nothing but WORK or the VALUE.

The main idea behind the artifacts is to get increased transparency of the key information. This shall facilitate the inspection and adaptation.

This is designed in such a way that everyone involved shall have the same understanding of the Artifacts.

Scrum Artifacts

Product Backlog

- *Ordered list of what is needed to improve the product*
- *Long term goal (product goal)*
- *Broken down to smaller items for sprint*

Sprint Backlog

- *Ordered list of what is needed to be done in a sprint*
- *Short term goal (Sprint goal)*
- *Plan by and for the developers*

Increment

- *Concrete step towards product goal*
- *Should meet definition of done*
- *Delivered by the end of the sprint*

Scrum Artifacts

Commitment ...

- Each artifact has a commitment
  - For the Product Backlog it is the Product Goal.
  - For the Sprint Backlog it is the Sprint Goal.
  - For the Increment it is the Definition of Done.
- All these commitments shall reinforce empiricism and the Scrum values for the Scrum Team and their stakeholders.

Scrum Artifacts

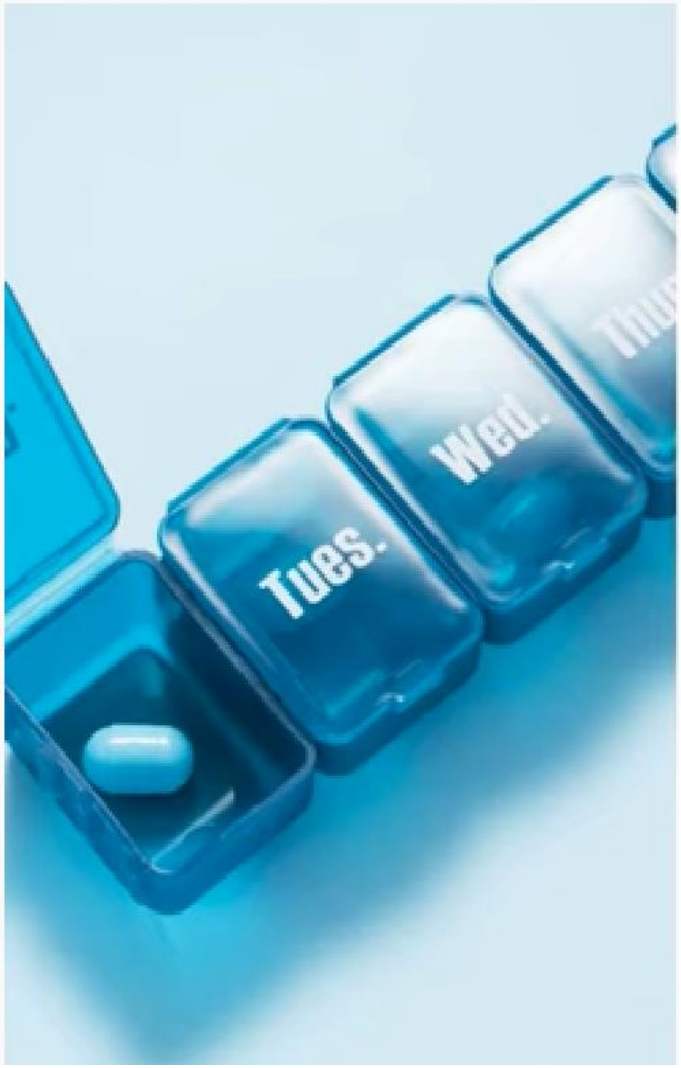
- Well, this is the to-do list. It has the list in a proper order of the items you must work towards the product.
- This is the reference or the only source that the SCRUM TEAM can refer to, to understand the requirements.
- Who owns this?, it is the product owner. The Product Owner is everything here. PO is responsible for the PB which include the content, availability and order.
- PO is the guy responsible to keep it intact and up to date. PO will include – Emerging Requirements, Demand, Customers feedback, all stake holders' feedback etc. This is Dynamic and keeps evolving with product. If you have product, you will have PB as well.
- Who can add the items to the Product Backlog? Well, you and I can. But PO is the one who can prioritize and reorders the list.
- Each item in a Product Backlog is called as a Product Backlog item (PBI).
- All these are towards enriching transparency.
- **Commitment : The Product Goal – Long term objective. Yes, you should complete/abandon one objective before taking next one.**

Scrum Artifacts

Sprint Backlog

- Definition: A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work. Sprints are at the very heart of scrum and agile methodologies and getting sprints right will help your agile team ship better software with fewer headaches.
- What is there in the Sprint Backlog? It has the Sprint Goal (Why??), **What** are the items shortlisted for the sprint? **How** is it planned to deliver the increment i.e., value?
- Who creates the Sprint Backlog? It is like democracy. For the developers, by the developers.
- Sprint Backlog is expected to be very visible and should reflect what the developers can do during the sprint.
- Sprint Backlog is consistently updated as well. Sprint backlog is expected to be discussed in the daily scrum too.
- **Commitment:** The Sprint Goal is the single objective for the Sprint. If the developers predict deviations during time, they will have to discuss with the PO, to see what can be done without affecting SPRINT GOAL.

Scrum Artifacts



Increment

- There can be multiple increments to reach the final product goal. Each increment is added to the previous one to reach the goal. It is additive.
- All increments are supposed to work together nicely. And increments must be USABLE!
- How many increments can be there within a sprint? Many is the answer. All of them together shall be presented for the Sprint review – This is called Empirism.
- If you complete the increment, it can be delivered after sprint review. (even earlier, is okay)
- **Commitment:** The Definition of Done is a formal description of the state of the Increment when it meets the quality measures required for the product. **THIS MUST BE MET. IF THERE ARE MANY SCRUM TEAMS, DEFINITION OF DONE WOULD BE DECIDED BY EVERYONE.**

Scrum Rules

- Each sprint starts with **sprint planning** done by the Scrum Master, Product Owner, and the rest of the team, consisting of a meeting divided into two parts, each timeboxed to four hours.
- The team holds a **Daily Scrum meeting every day**. All team members (including the **Scrum Master and Product Owner**) must attend, and interested stakeholders may attend as well (but must remain silent observers). The meeting is **timeboxed to 15 minutes**, so all team members must show up on time. Every team member answers **three questions**: **What have I done since the last Daily Scrum? What will I do between now and the next Daily Scrum? What obstacles and roadblocks are in my way?**
- Each sprint is timeboxed to a specific length decided during sprint planning: many teams use 30 calendar days, but this length can vary—some teams choose two-week sprints, some choose one month (and again, the planning timebox should vary accordingly). During the sprint, the team builds the items in the sprint backlog into working software.

Scrum Rules

- At the **end of the sprint**, the team holds a sprint review meeting where they demonstrate working software to users and stakeholders. The demo may only include items that are actually done.
- After the sprint, the **team holds a sprint retrospective** meeting to find specific ways to improve how they work. The team and Scrum Master (and optionally the Product Owner) attend.
- Each person answers two questions:
- **What went well during the sprint?**
- **What can improve in the future?**

Scrum Events

Scrum Events

All the events in the scrum are all contained in the container called the Sprint. So, remember it contains everything.

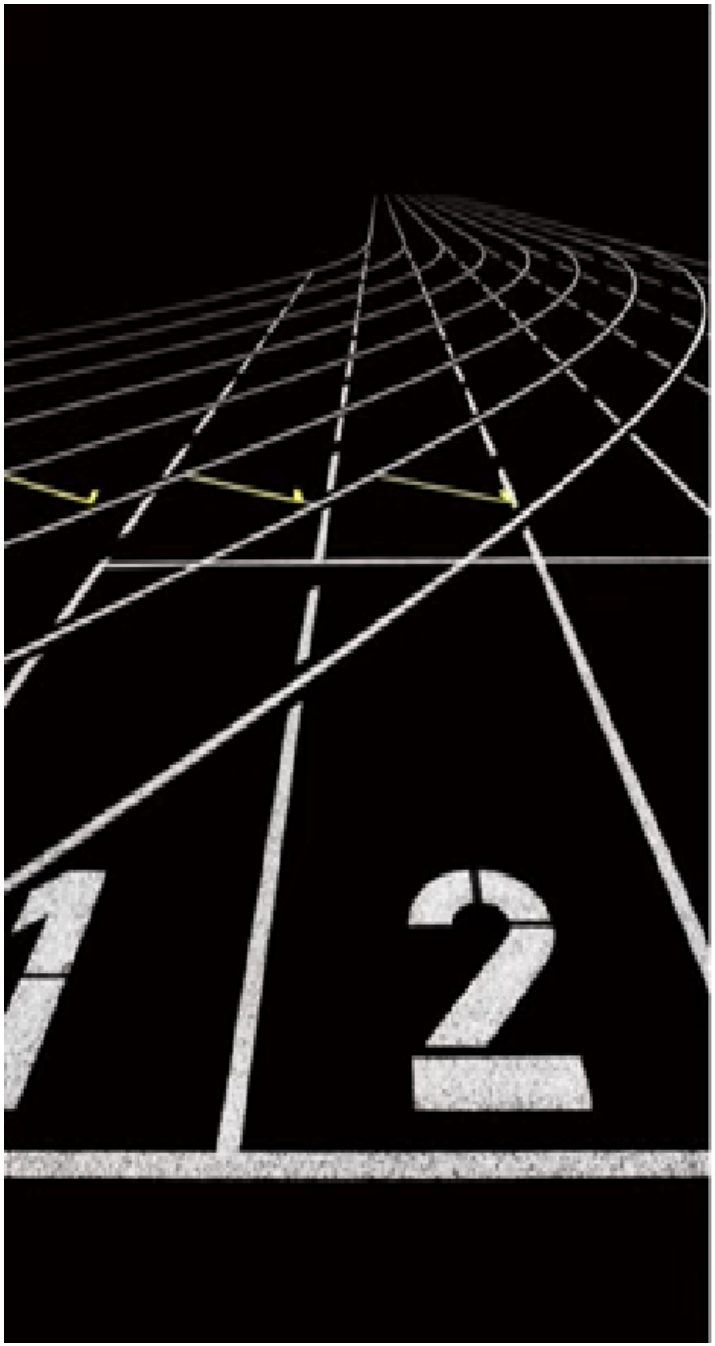
All the events in the scrum are very carefully designed with the core aim and target to inspect and adapt Scrum artifacts.

All these events are also focusing on the increased transparency.

All the events are designed to make sure things go with regularity and to reduce unnecessary meetings other than "Scrum Meeting" conducted every day.

All the events are time boxed; they have a start time and end time for sure! No meetings can go day long!

Scrum Events



Sprint

- Sprints are the "Core" of the scrum where ideas/thoughts are converted to "Values" – Values are the target!
- Definition: A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work. Sprints are at the very heart of scrum and agile methodologies and getting sprints right will help your agile team ship better software with fewer headaches.
- Mostly they are of for a period of month or less.
- When a new sprint can be started? Well, once you complete the previous one.
- All the work Starting with - **Sprint Planning, Daily Scrums, Sprint Review, and Sprint Retrospective are conducted within the Sprint.**

Sprint

When the sprint is ongoing…

Nothing that would endanger sprint goal can be done.

Quality is paramount, it should not be tampered.

Refinement of PB is okay.

Scope may be clarified and renegotiated with the Product Owner as more is learned.

- A Sprint could be cancelled if the Sprint Goal becomes obsolete. Only the Product Owner has the authority to cancel the Sprint.

Sprint Planning

Sprint Planning

- We can call this as a blueprint or the planning.
- Yes, here is where, the planning happens on what kind of work is to be performed for the sprint.
- The Scrum Team is the one who creates this collaborative plan.
- PO shall ensure the attendees are aware of the PBI (Product Backlog Items, i.e. the list of items in the Product Backlog)
- The Scrum Team may also invite other people to attend Sprint Planning to provide advice like the SMEs or Experienced People.

Sprint Planning is timeboxed to a maximum of 8 hours for a one month Sprint

How to Plan and Run an Effective Scrum Sprint

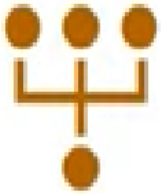
- **Starting with the backlog**—which means starting with the users
- Be **realistic** about what you can deliver
- **Change the plan** if it needs to change
- Get **everyone talking** about value

Sprint Planning

Why?

Why is this Sprint valuable?

Why? This is a good question. PO will analyze and proposes how the product will elevate the value during the current sprint.



Scrum team comes together to define a common sprint goal which directly is related to the sprint's value. The same gets communicated to the stakeholders.

Remember – Sprint Goal must be finalized before the Sprint Planning.

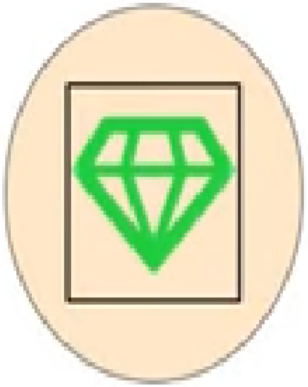
What?


What can be Done this Sprint?
REMEMBER – PRODUCT BACKLOG IS THE COLLECTION OF ITEMS TO BE WORKED ON.
PO AND THE DEVELOPERS DISCUSS AND SELECT THE ITEMS FROM PB AND INCLUDE THEM FOR THE CURRENT SPRINT.
THERE CAN BE REFINEMENT, IMPROVEMENT OF THESE ITEMS BY SCRUM TEAM.
WELL, THERE IS AN IMPORTANT THING TO BE REMEMBERED. HOW MUCH CAN BE COMPLETED IN THE TIME BOX? I MEAN, THE SPRINT. IT IS CHALLENGING TO SELECT.
DEVELOPERS ARE THE BEST TO DECIDE THIS – AS THEY HAVE PAST EXPERIENCES!

How?

How will the chosen work get done?

THE NEXT QUESTION IS HOW! DEVELOPERS ARE THE ONE WHO MUST PLAN AND WORK TOWARDS THE PLAN. REMEMBER, IT SHOULD MEET THE DEFINITION OF DONE!



SMALLER THE BETTER, DIVIDE AND RULE ARE GOLDEN WORDS! DO THE SAME HERE. BREAK THE ITEMS INTO SMALLER ITEMS AND DEVELOPERS ARE THE BEST TO DECIDE HOW MUCH AND HOW CAN THE WORK BE BROKEN.

FINAL GOAL IS – TURN THE PB TO INCREMENTS (VALUE, OUTPUT)

Daily Scrum

- The Daily Scrum is an **effective tool for visibility** because it helps with these communication problems. When those problems happen, they come out in the work.
- How to **Hold an Effective** Daily Scrum
- Take detailed **conversations offline**
- Take turns going first
- Don’t treat it like a **ritual**
- Everyone **participates**
- Don’t treat it like a **status meeting**
- **Inspect** every task
- **Change the plan** if it needs to be changed

Daily Scrum

We discussed this – Purpose is to **Inspect and adapt.** Yes, daily inspection which will lead to adaptation.

This enables adjustment, tuning in the planned work which will in turn lead to success.

How much time to be spent for the Scrum? – 15 mins a day. Maximum 10 members can be in the team. Smaller the better.

Normally, it is scheduled for the same time, place etc. to make it easy for everyone.

In some cases, if the scrum master himself or PO himself is actively working on the product backlog items, they can also participate as developers in the scrum meeting.

Daily Scrum is aimed @ improving communications, identification of impediments, quick decision making and **importantly to avoid other unnecessary meetings.**

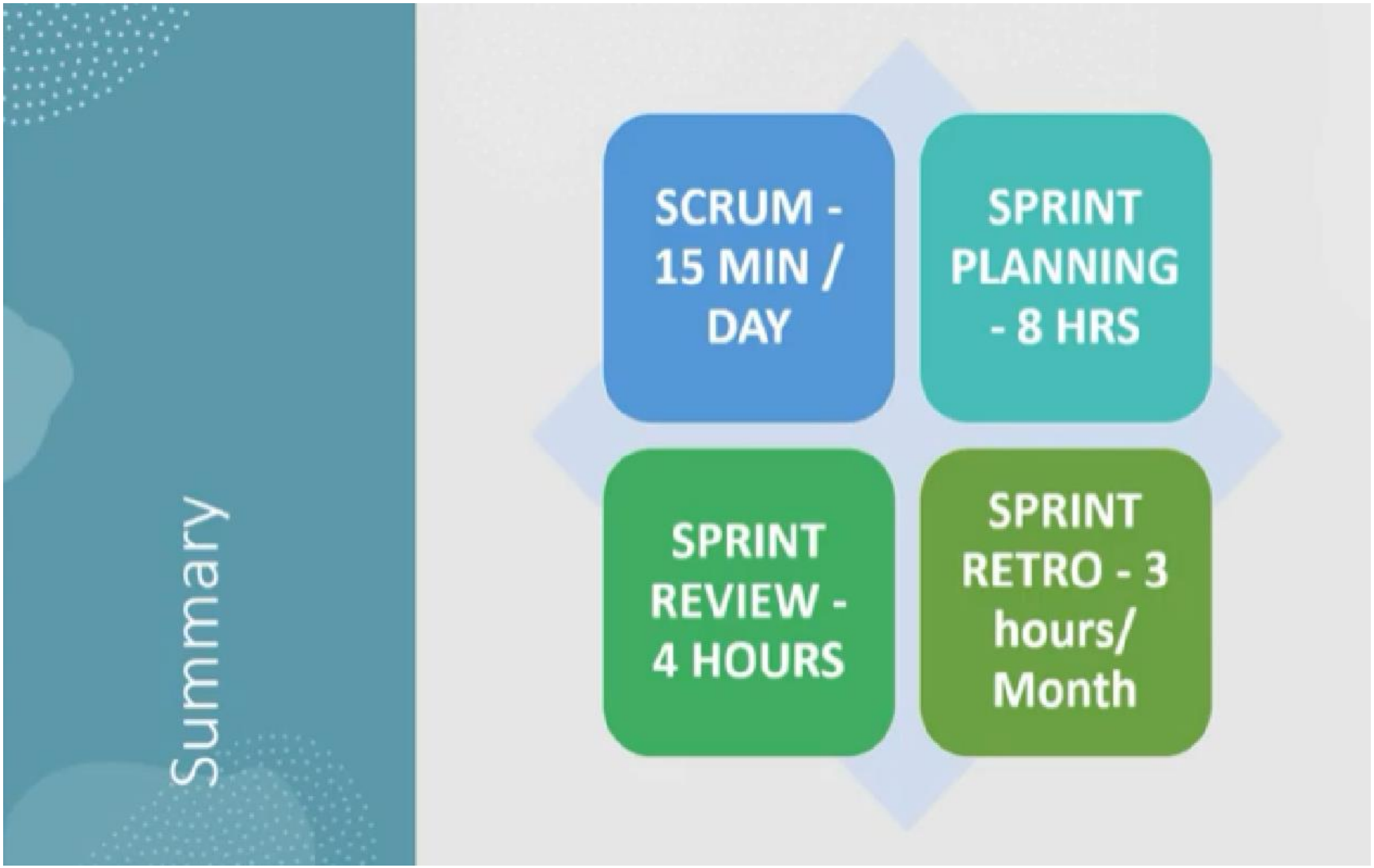
Developers can alter/adjust their plans not just during scrum. They are free to meet rest of the time as well.

Sprint Review

- Scrum Team presents the results of their work to key stakeholders at the end of the sprint
- Scrum Team and stakeholders review what was accomplished in the Sprint and what has changed in their environment
- Timeboxed to a maximum of 4 hours for a one-month Sprint

Sprint Retrospective

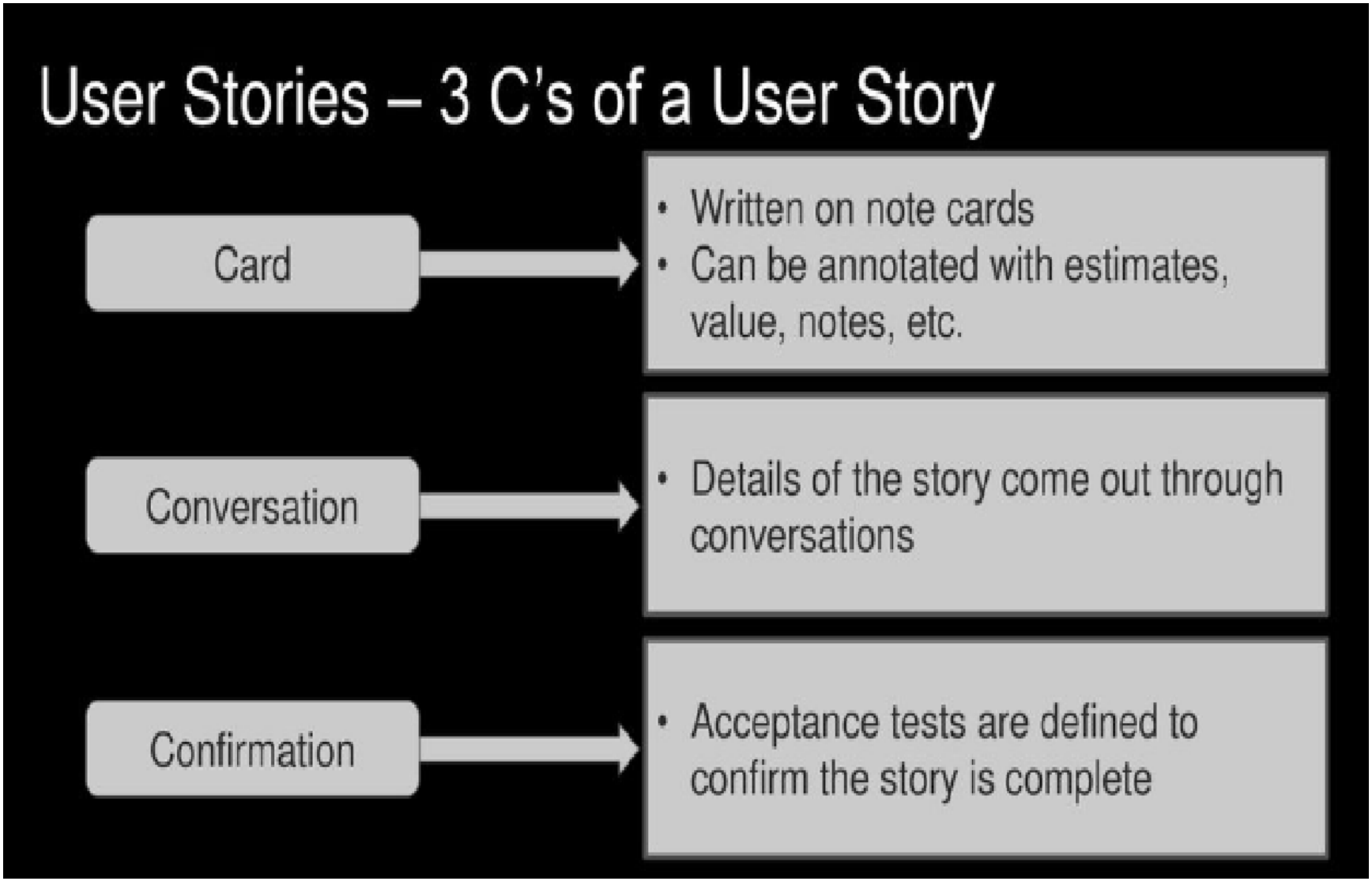
Summary


Summary
SCRUM - 15 MIN / DAY
SPRINT PLANNING - 8 HRS
SPRINT REVIEW - 4 HOURS
SPRINT RETRO - 3 hours/ Month

User Stories

- A user story is the smallest unit of work in an agile framework.
- It is the short requirements written from the perspective of an end user.
- It is mostly written in Non-Technical Language.
- It clearly defines the functionality which the End User expecting

User Stories



User Story Format

AS A (End User)

I WANT (Goal)

SO THAT (Benefit)

End User – User Type/Role

Goal – the functionality that the end user is expecting

Benefit – the benefit the end user gets from the developed Functionality

User Story is always written from the end user perspective

User Stories

- User Stories also give **teams an easy way to manage** their backlog
- User Story is **simple and written** from the user's perspective, it's easy for the Product Owner to review the stories with the users and Stakeholders – to figure out which stories are valuable
- Each **Story is small**, which make it easier to add new ones or change their order at any time
- To Start the sprint – **Product owner and team can pull** of the stories out of the backlog and designate them to be delivered in this sprint
- The team can discuss the stories with the **Product Owner** and get the clarity
- Most team will then **break down the stories** into task and start to estimate how long those tasks will take
- The task for each story would go into the **"ToDo" Column** of the taskboard
- Once the team member taken the task it moved to **"In-Progress"** Column of the taskboard

Conditions of Satisfaction

- **Condition of Satisfaction** – effective tool for helping **developer** to know what the software will look like when it is complete and to gauge how close they are to done
- Condition of Satisfaction – **simple but accomplish a complex goal**
- Condition of Satisfaction is **defined for each user story**
- Some team refer it to **"acceptance criteria"**
- Condition of satisfaction can fit on the back of the same **3X5 index card**
- These conditions are **valuable to developers** because it helps them avoid declaring victory too early
- Conditions of Satisfaction helps this by giving the team a concrete definition of **"Done"**
- **User Story is "Done"** only when the **conditions is satisfied** and it would be demonstrated in the sprint review

Example.,

Summary(Title) : Student Login page

User Story : AS A Student **I WANT** to have a Login functionality **SO THAT** I can successfully use Student Module

Scope (Expected output) :

- Build a Login Page
- Student validation

Pre – Condition : Student should have registered in School to use login

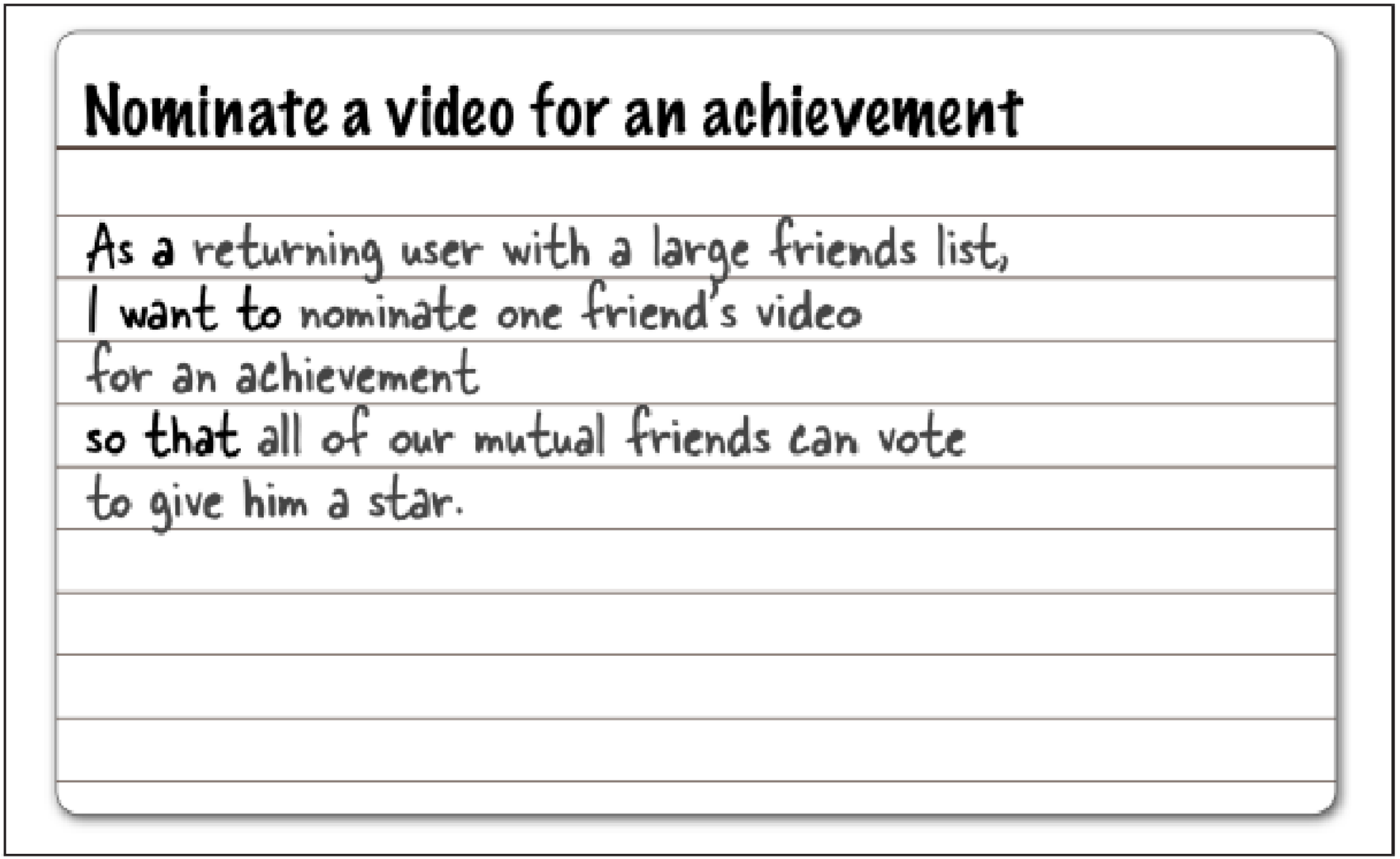
Non-Functional Requirement:

- It should work in all browsers
- Student should be able to login within 3 seconds

Acceptance Criteria : Conditions that a product must meet to be accepted by the user

Scenario 1 : Student can Login successfully	**Scenario 2 : Student can't Login successfully**
Given I am on Login Page	**Given** I am on Login Page
When I provide Valid Username and Password and I click on Sign In	**When** I Provide In-Valid Username and Password and I click on sign In
Then I will login successfully	**Then** I will get an error message "Please provide valid credentials

User Stories



A user story written on an index card.

User Stories

- This user story is effective because it makes three important things obvious:
- **Who the user is**: “a returning user with a large friends list”
- **What the user wants to do**: “nominate one friend’s video for an achievement”
- **Why the user wants to do it**: “so that all of our mutual friends can vote to give him a star”

Conditions of Satisfaction

Conditions of satisfaction written on the back of the user story index card.

User Stories

A good user story should fulfill this Checklist

- Keep them short
- Keep them simple
- Write from the perspective of the end user
- Make sure that the value/reason for the story is clear
- Story should have User Acceptance Criteria to test the functionality
- It should be testable
- Make sure that the story is linked to related Epic (Collection of user stories)

Story Points and Velocity

- **Story points** are units of measure for expressing an estimate of the **overall effort** required to fully implement a **product backlog** item or any other piece of work.
- Actual **velocity is calculated** by dividing the total Story Points completed by the team by the number of Sprints. For instance, if the Scrum Team has finished a total of **80 points over 4 Sprints** then the actual velocity of the team would be **20 points per Sprint.**

Story Points and Velocity

- Story points are a **way to understand how much effort** you need to build user story by assigning a number of points to it
- The teams comes up with those points by comparing the current user stories to other story that a team has build in the past
- Some team used the numbers between **1 and 5**, some use between **1 and 10**, some use **Fibonacci series, some use exponential numbers for story points**
- As a team assign points to all of the stories they work on , they start to see how many points worth of stories they can complete in a sprint
- **Velocity in Scrum** is a **measure of how much work** is completed by the Developers within a specific Sprint.
- If your team complete an average of **25 point worth of story** in a sprint , then their *project velocity is* 25 points/sprint
- Teams that tends to work on **constant velocity** at each sprint
- You can use the velocity from past sprints to help you plan for the next one

Story Points and Velocity

- A **sprint planning** session using story points might go like
- Start with **most valuable user stories** from the product backlog
- Take a story in that list – **ideally the smallest one** , because that makes a good baseline for comparison – find a similarly sized story from a previous sprint, and assign it the same number of points
- Discuss with the team whether that estimate is accurate- additional problems , work, or technical changes – **increases the estimate** ; **reusing existing code** – decreases the estimate
- Keep going through the stories until you have accumulated enough points to fill the sprint

Why Story Points work

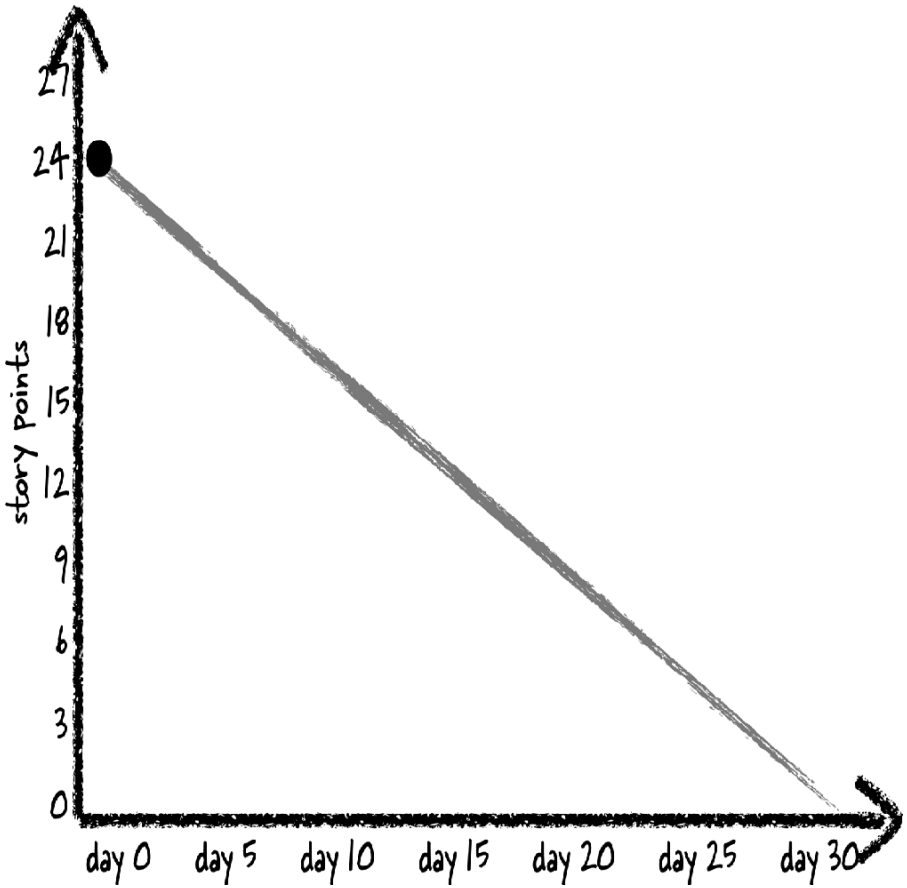
- They are **simple**
- They are **not magic**
- The team is control of them
- They get your team talking about estimates
- Developers **aren't scare** of them
- They help the team discover **exactly what a story means**
- They help every one on the team become genuinely committed

Burndown Charts

- It is a **visual representation** of remaining amount of work for any specific sprint at Scrum team level.
- A burndown chart is a way for anyone to see, **at a glance,** how the sprint is actually progressing when compared with the team's past velocity.
- A burndown chart is almost a **"must"** have tool for a Scrum team for the following main reasons:
- **Monitoring** the project scope creep
- Keeping the **team running on schedule**
- Comparing the planned work against the team progression

Points to build a burn down chart:

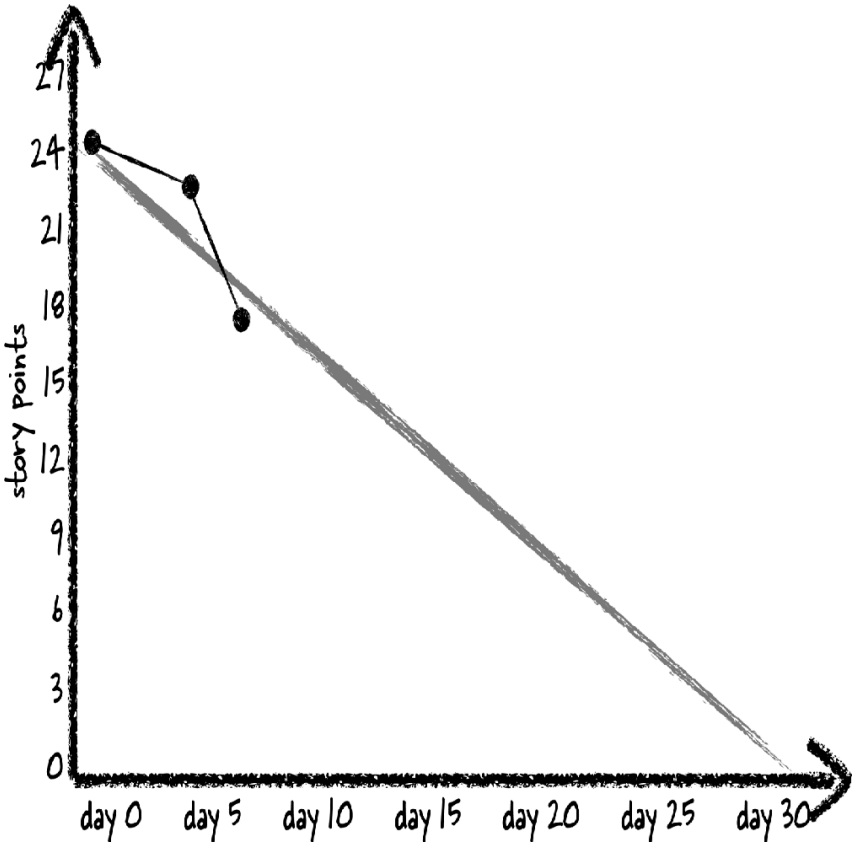
- Start with empty line chart. **X- axis** has dates, starting with first day of the sprint and ending with last day.
- **Y – axis** has story points, ranging from **0 to 20%** more than the total number of points in the sprint backlog
- Draw the first dot on the chart: the number of points in the sprint, at day 0
- Draw a straight line – called **"guideline"** from the first point to the end of the project – **zero points left when the sprint is complete**
- This burndown chart is created using **story points**, but you can also use hours, days, or another unit of measurement.



Burndown chart when the sprint starts.

Burn Down Chart

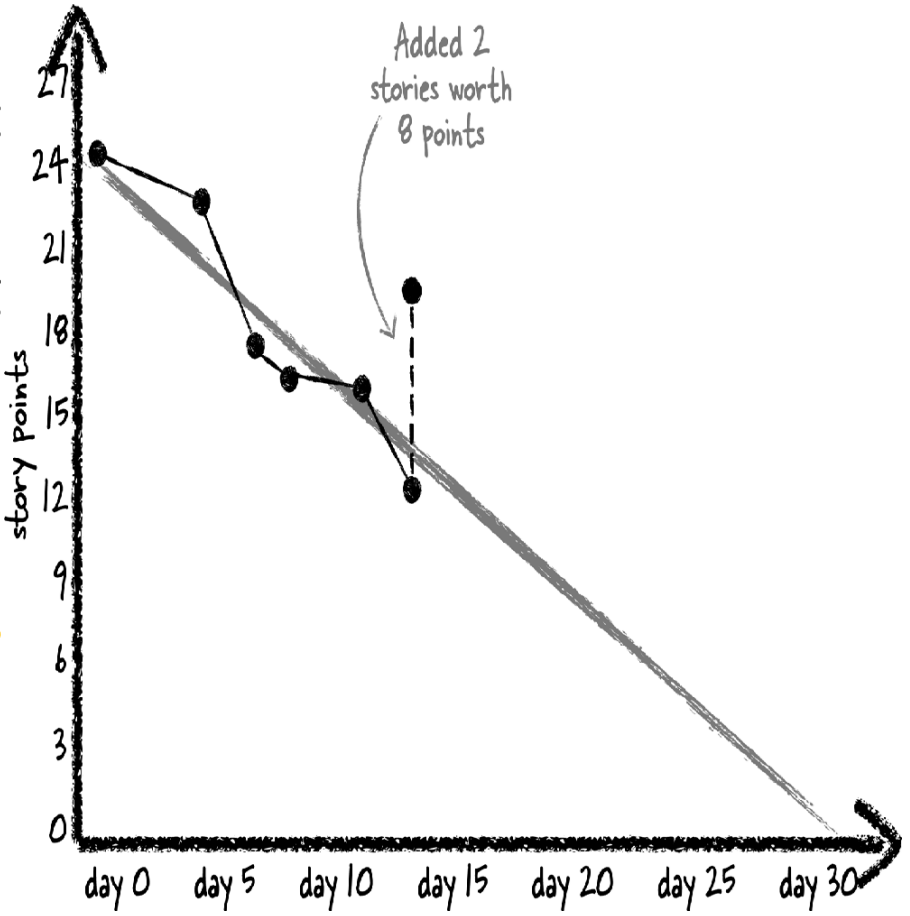
- As soon as the first user story is **finished** and moved to the **"Done"** column of the **task board**, draw the next dot on the chart: the number of points left in the sprint, at the current day.
- As you finish more stories and **burn** more points off of the **backlog**, fill in more days in the **burndown chart**.



Two stories worth 7 points burned off

Burn Down Chart

- Some stories are **added halfway** through the sprint by the product owner
- You might discover during a **Daily Scrum** that more work needs to be added.
- Sometimes you'll do this because the **team is burning** more points than they expected, and they'll finish early as a result.

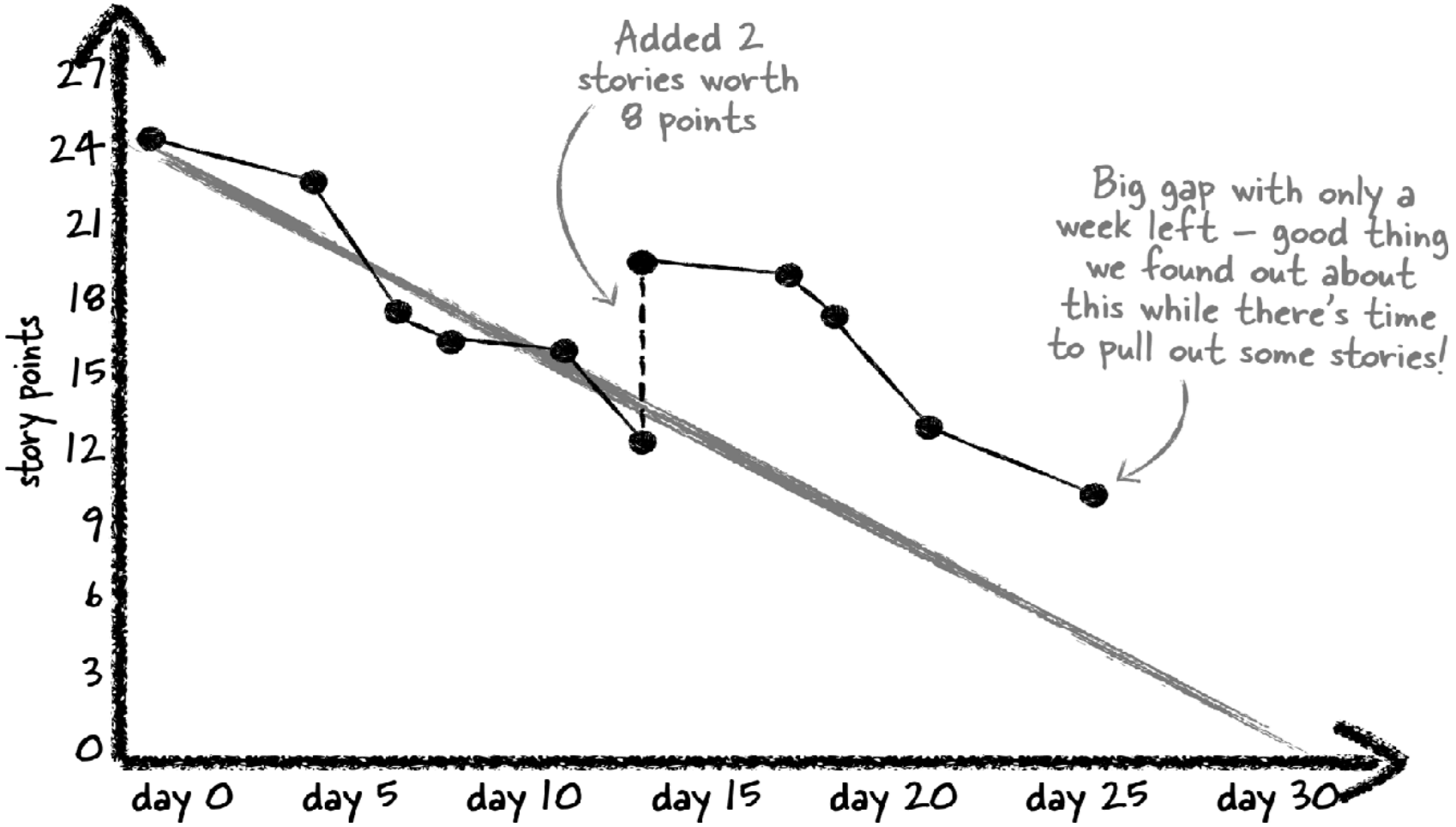


Burn Down Chart

- As you get close to the **end of the sprint**, more and more points burn off the chart.
- Keep an eye out for a **gap between the guideline** and your actual burndown, because that could mean you've got too many points **left in the sprint** and need to remove a user story.

Burn Down Chart

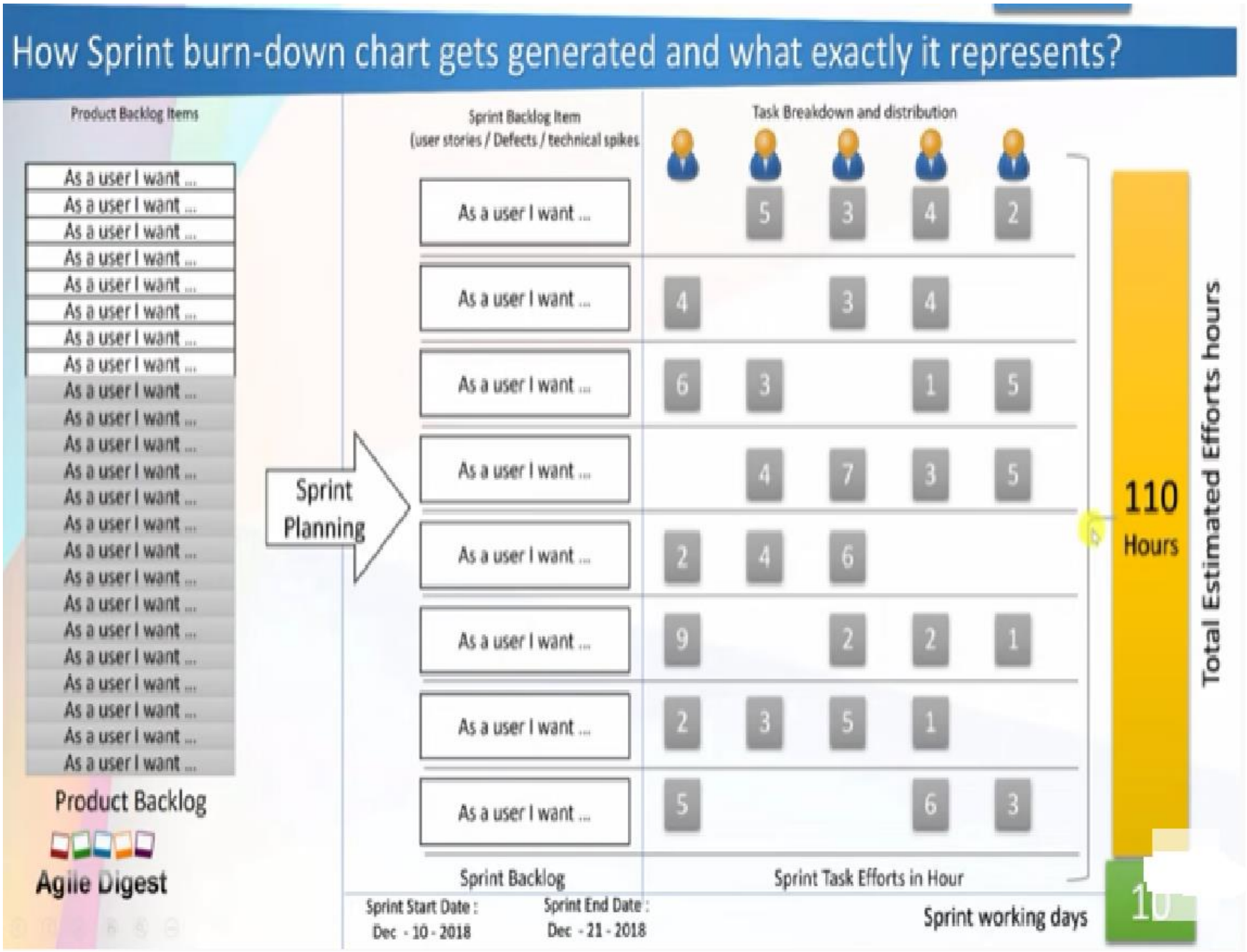
A gap between the **burndown chart and the guideline** tells that there's a good chance you won't finish all of the stories by the end of the sprint.



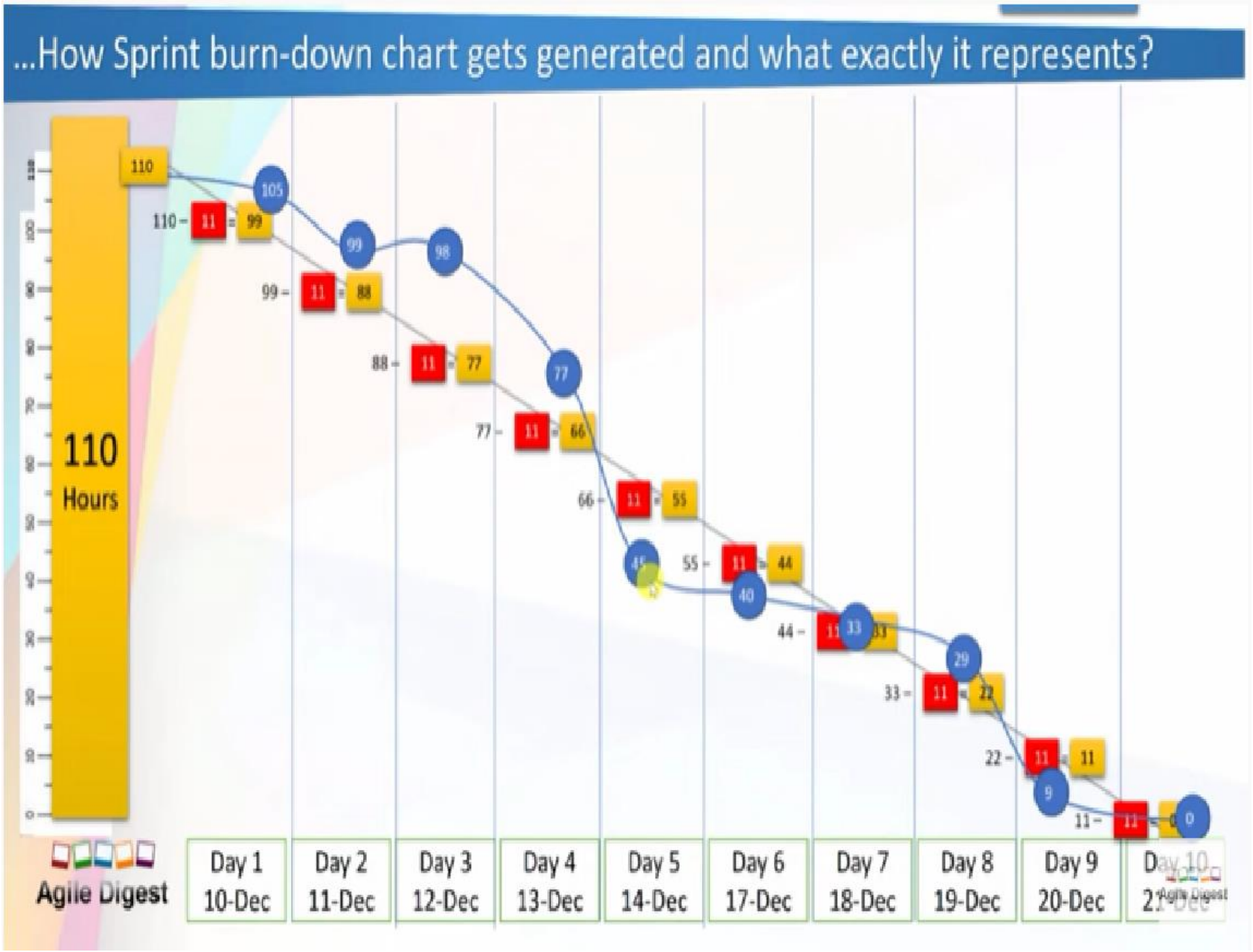
Burn Down Chart

- There are lots of great software packages that let you **manage your backlog, stories, and story points, and draw burndown charts** automatically.
- However, many teams prefer to **draw burndown charts by hand**, and keep the chart on the same wall (often on the same whiteboard) as the task board.
- This lets everyone see exactly **how the project is doing at any time**.
- It's especially satisfying for the team if each developer gets to update the burndown chart after he or she completes a **user story** and moves the card to the **"Done" column**.

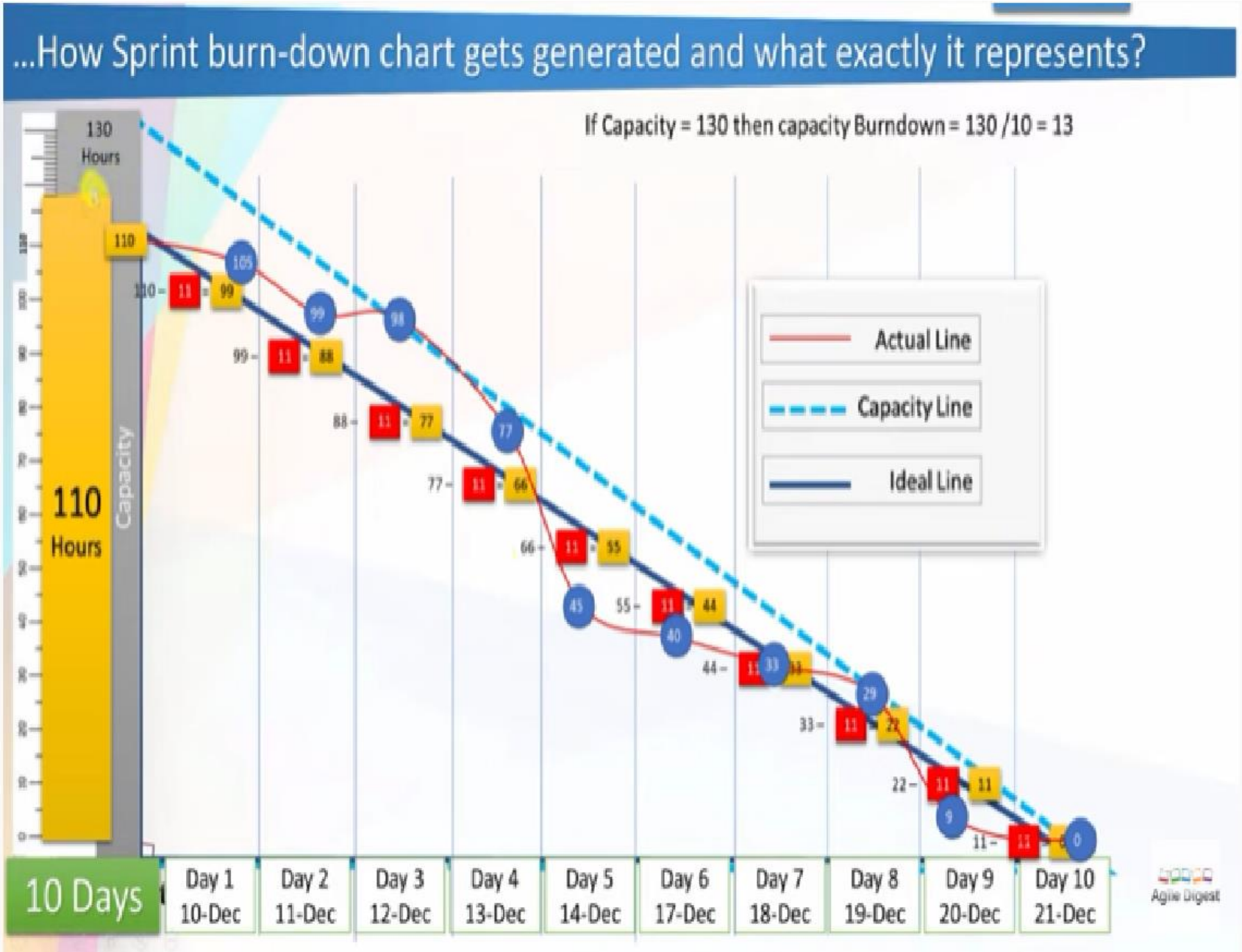
How Sprint burn-down chart gets generated and what exactly it represents?



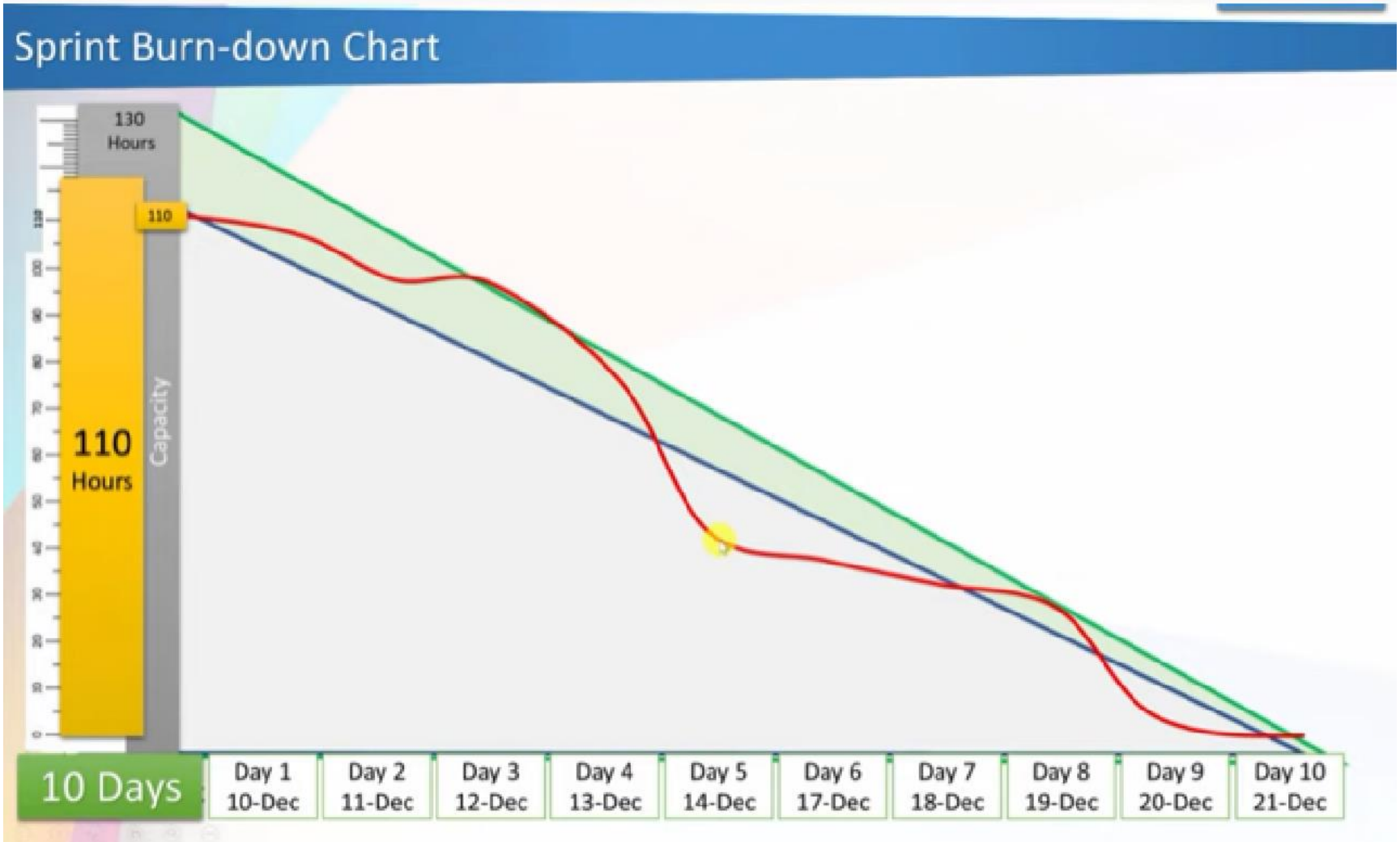
...How Sprint burn-down chart gets generated and what exactly it represents?



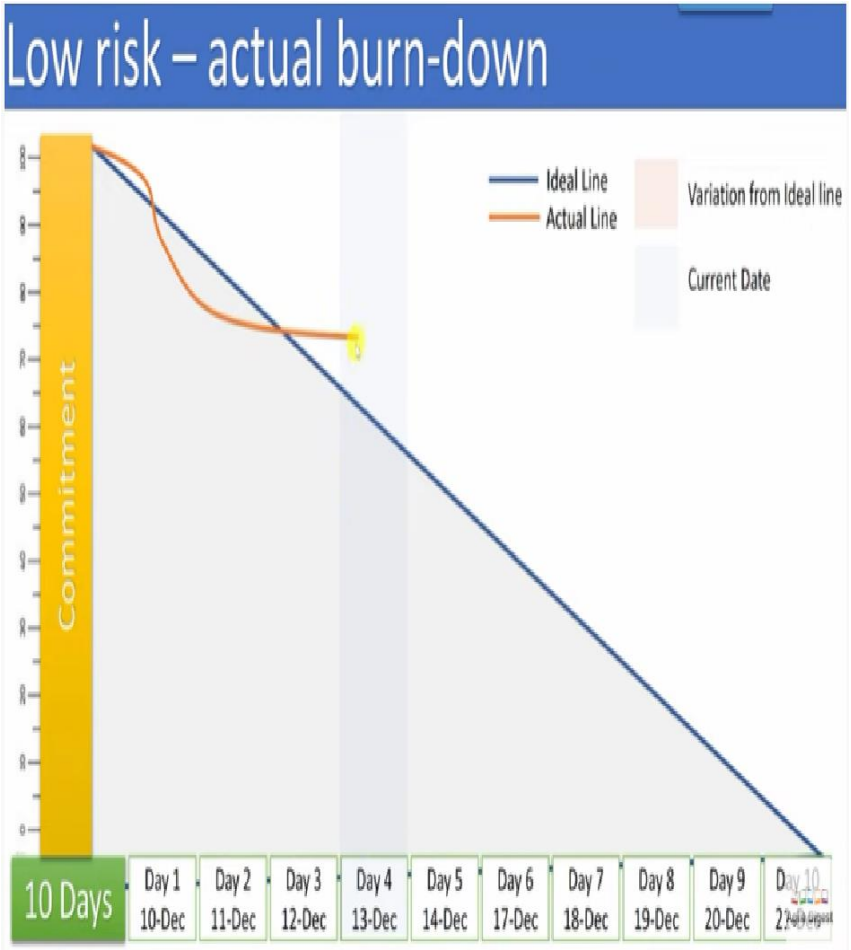
...How Sprint burn-down chart gets generated and what exactly it represents?

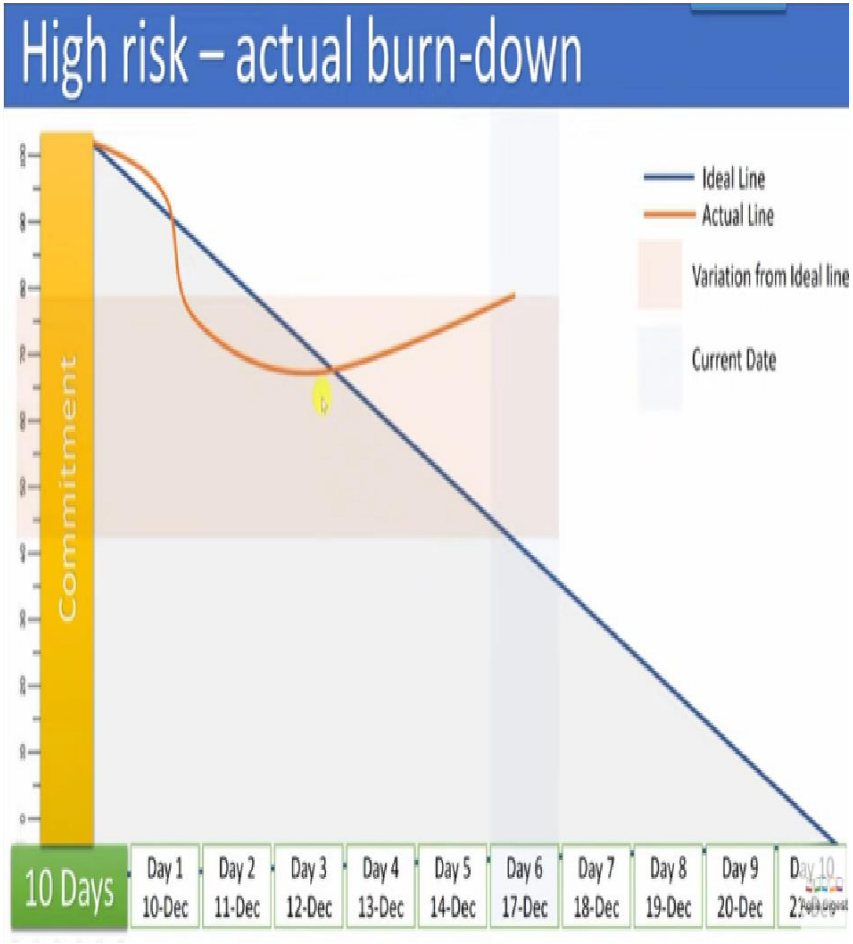


Burn Down Chart

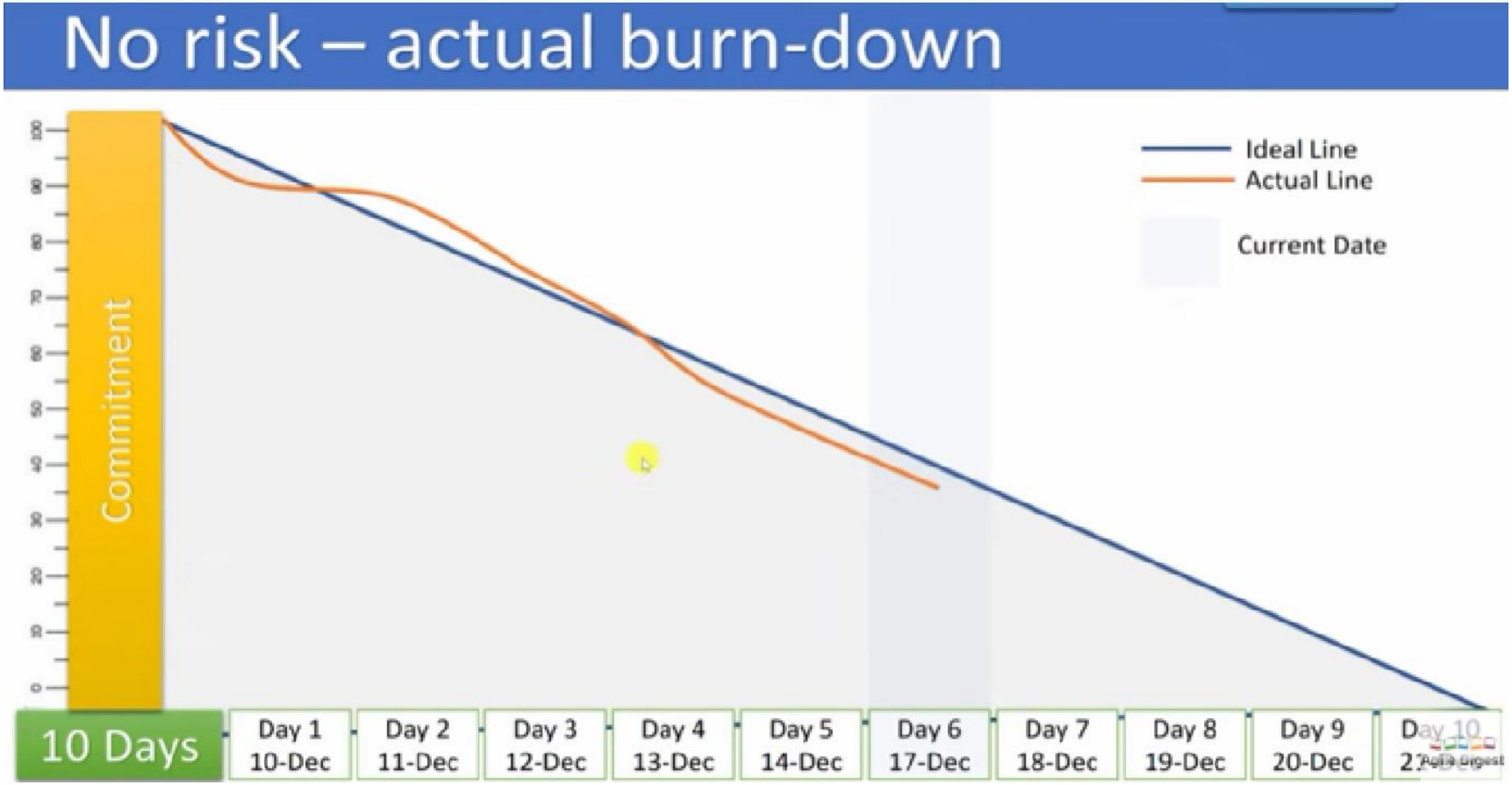


Burn Down Chart

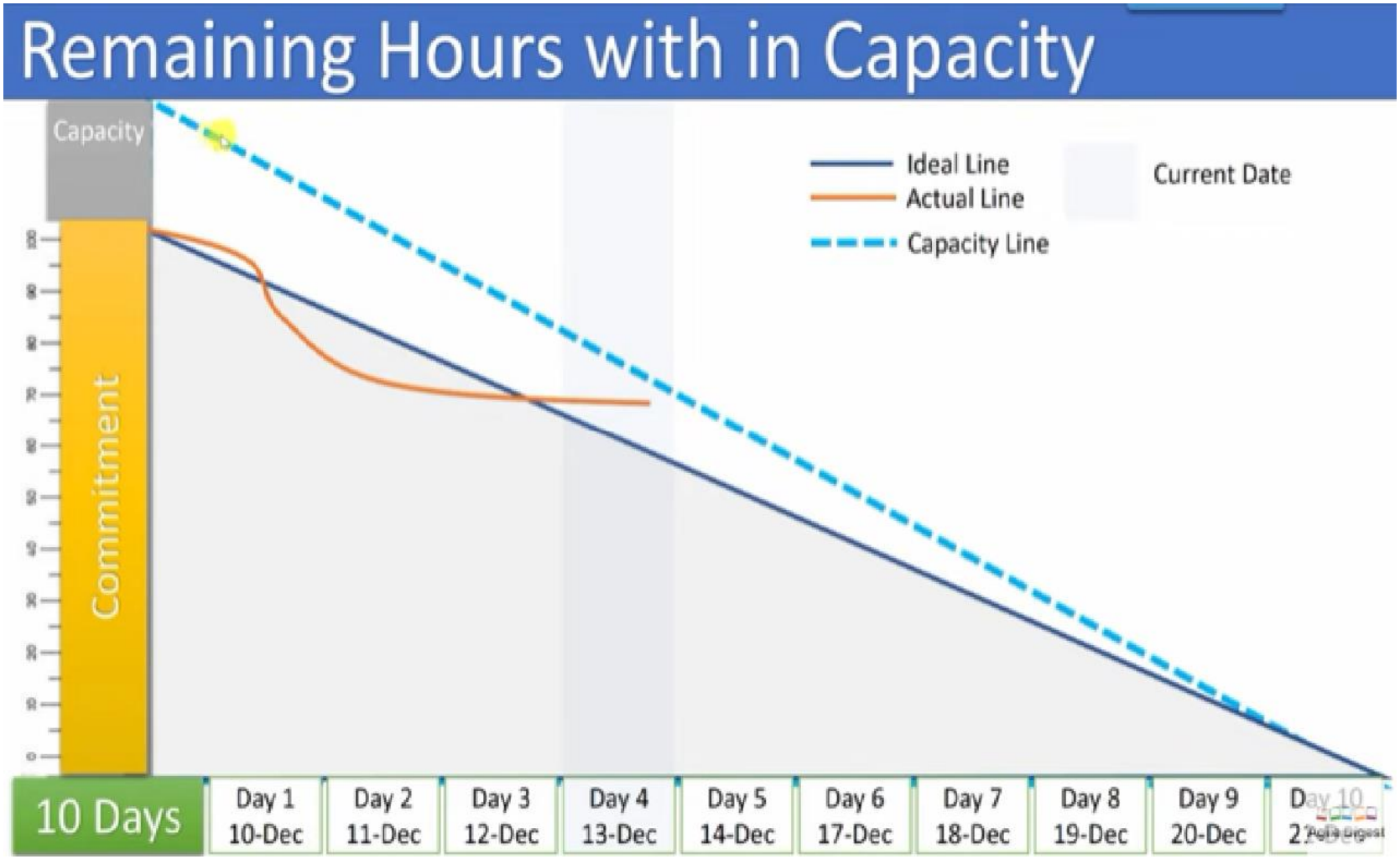




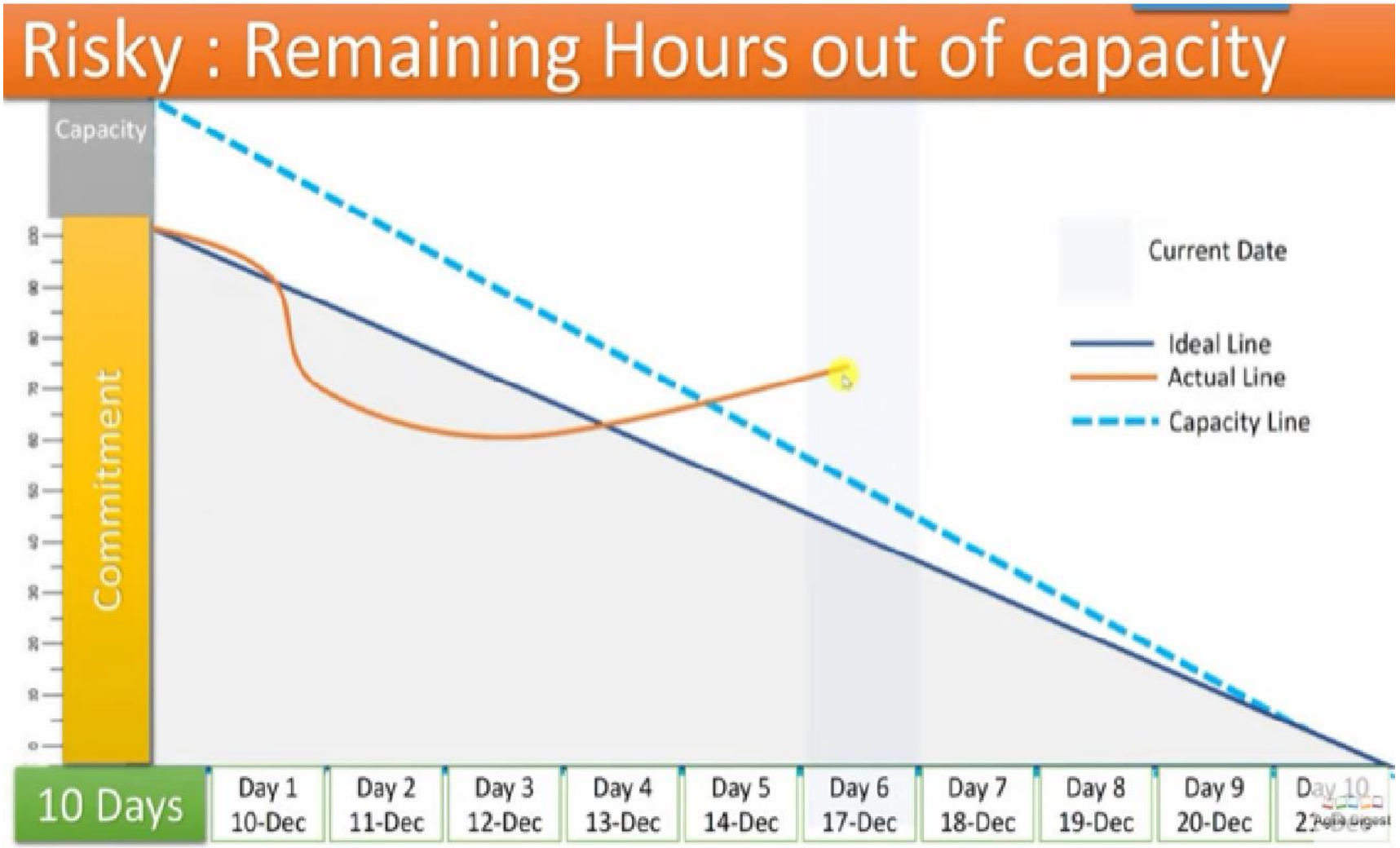
Burn Down Chart



Burn Down Chart



Burn Down Chart

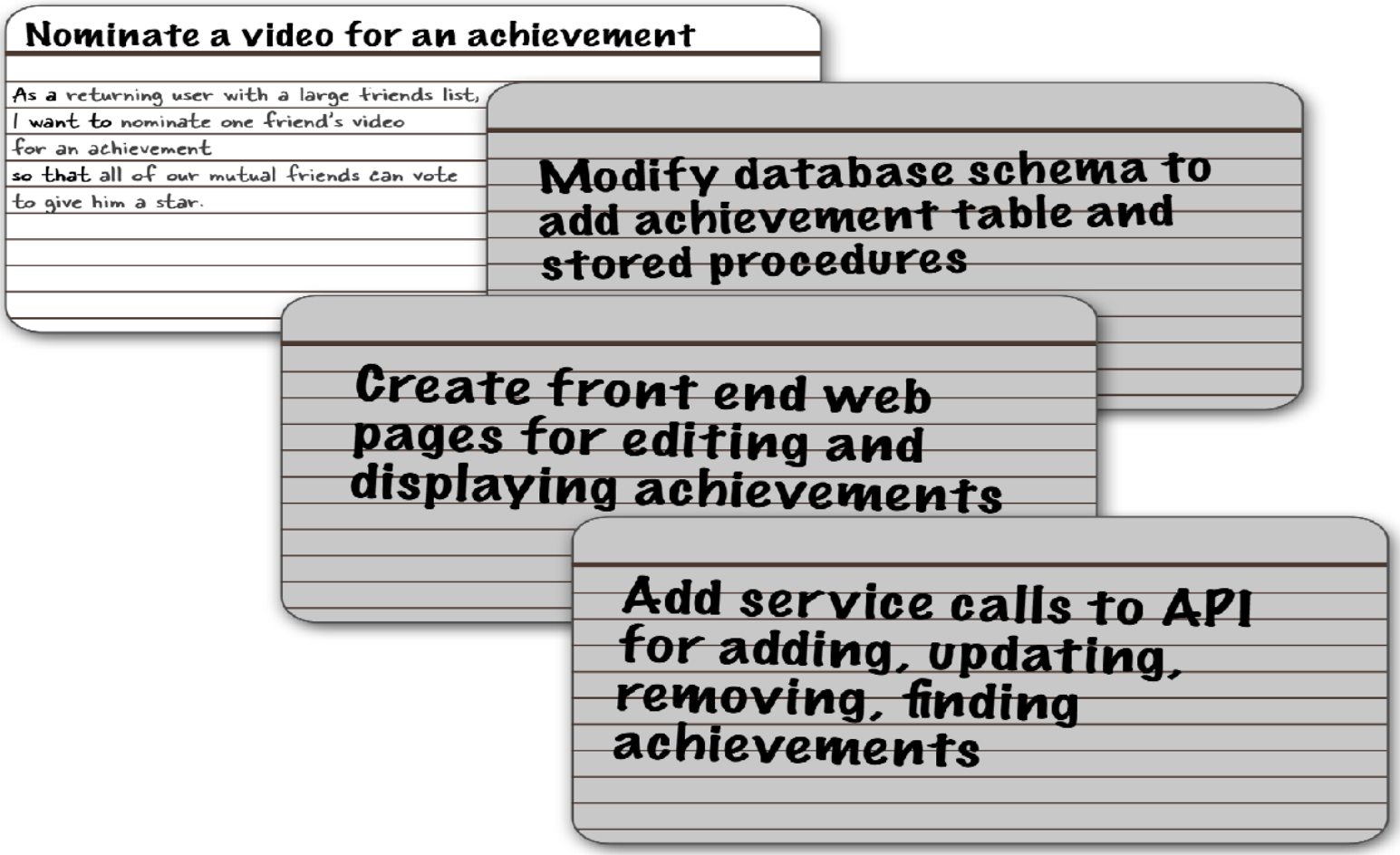


Planning and Running a Sprint

- One of the most common ways for **Scrum teams to plan** out the actual work for the team is to add cards for individual development tasks.
- The team holds the second **sprint planning meeting**.
- The **Scrum Master** starts with the first story, and leads a discussion with the team about exactly what they need to do to build it. Everyone **works together to come** up with a list of individual tasks that they think will each take no more than a day to complete.
- Each task is written on a separate card—some teams use different colored cards for stories and their tasks to make it easier to tell them apart—and the story card is **grouped together with its task cards**.

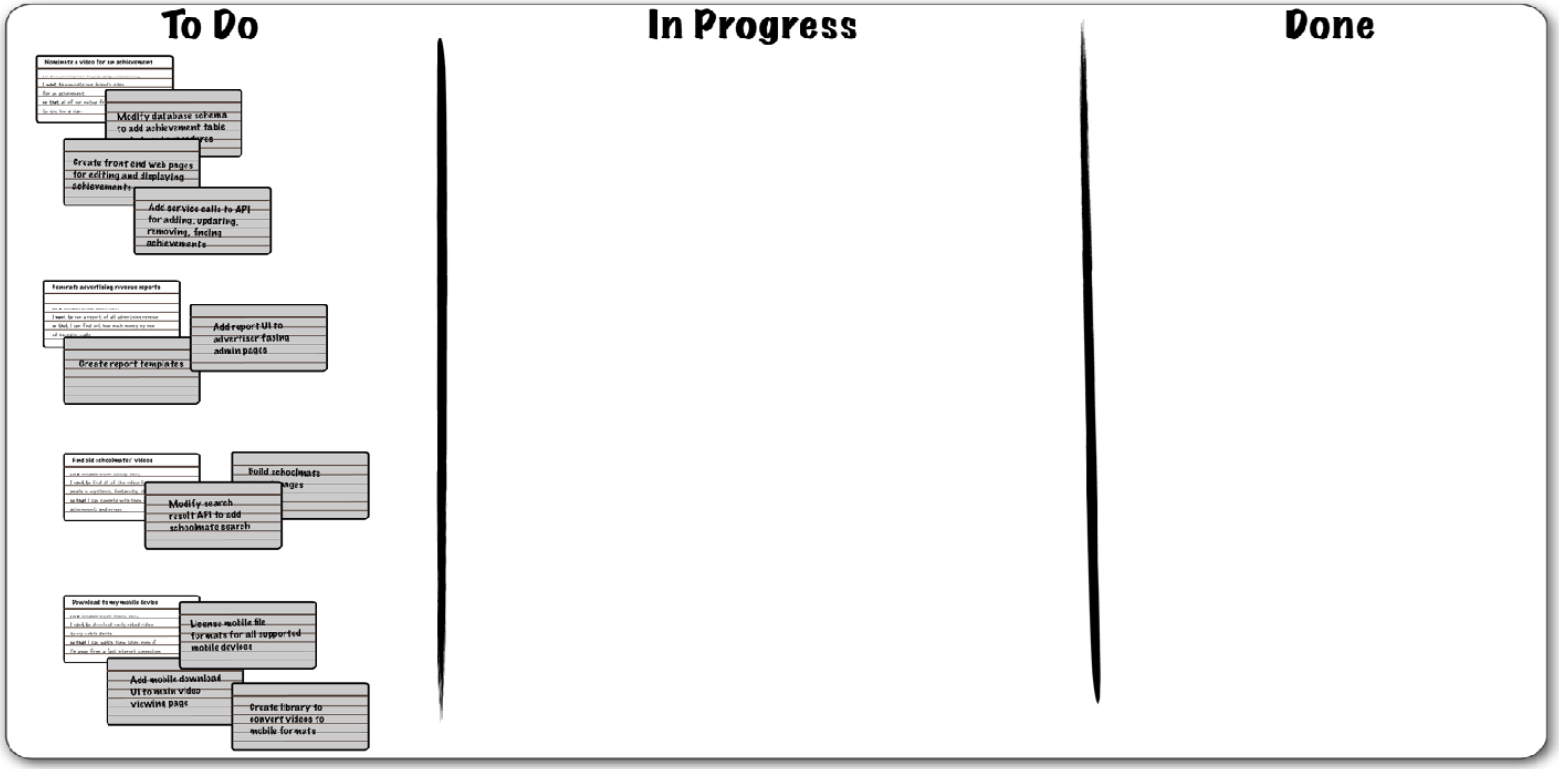
Planning and Running a Sprint

The second half of the **sprint planning session** is all about breaking the stories down into tasks that the team members will do during the sprint



Planning and Running a Sprint

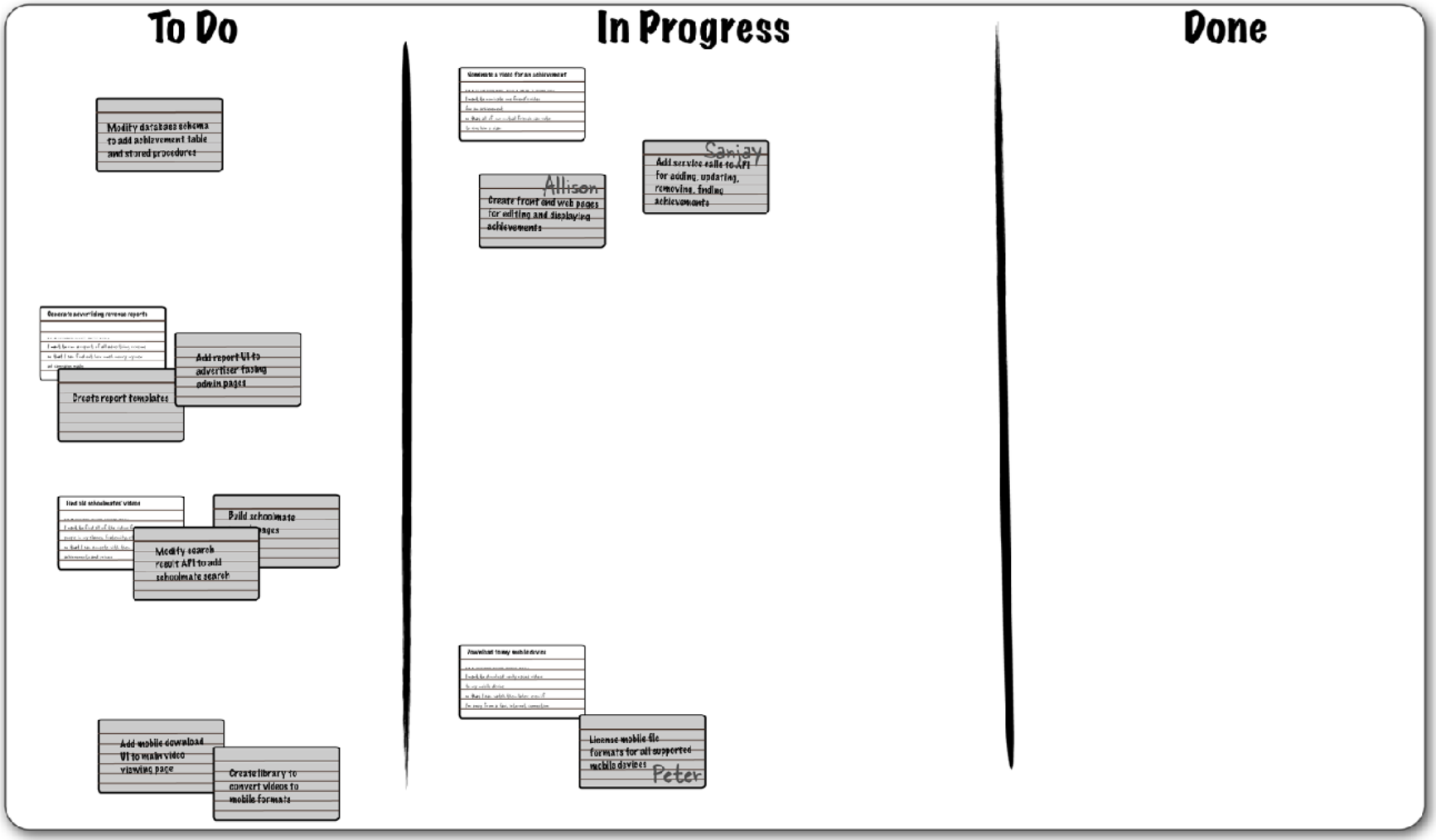
- The stories and their tasks are grouped together and added to the **"To Do" column** of the **task board**.



Planning and Running a Sprint

- When a team member finishes one task and is ready to move on to the next, he moves the task card for the completed work to the **“Done” column**.
- Then he pulls the task card for the next piece of work out of the **“To Do” column**, writes hisname on it, and then puts it back on the board in the “In Progress” column.
- If the story is still in the **“To Do” column**, he moves it to **“In Progress”** as well

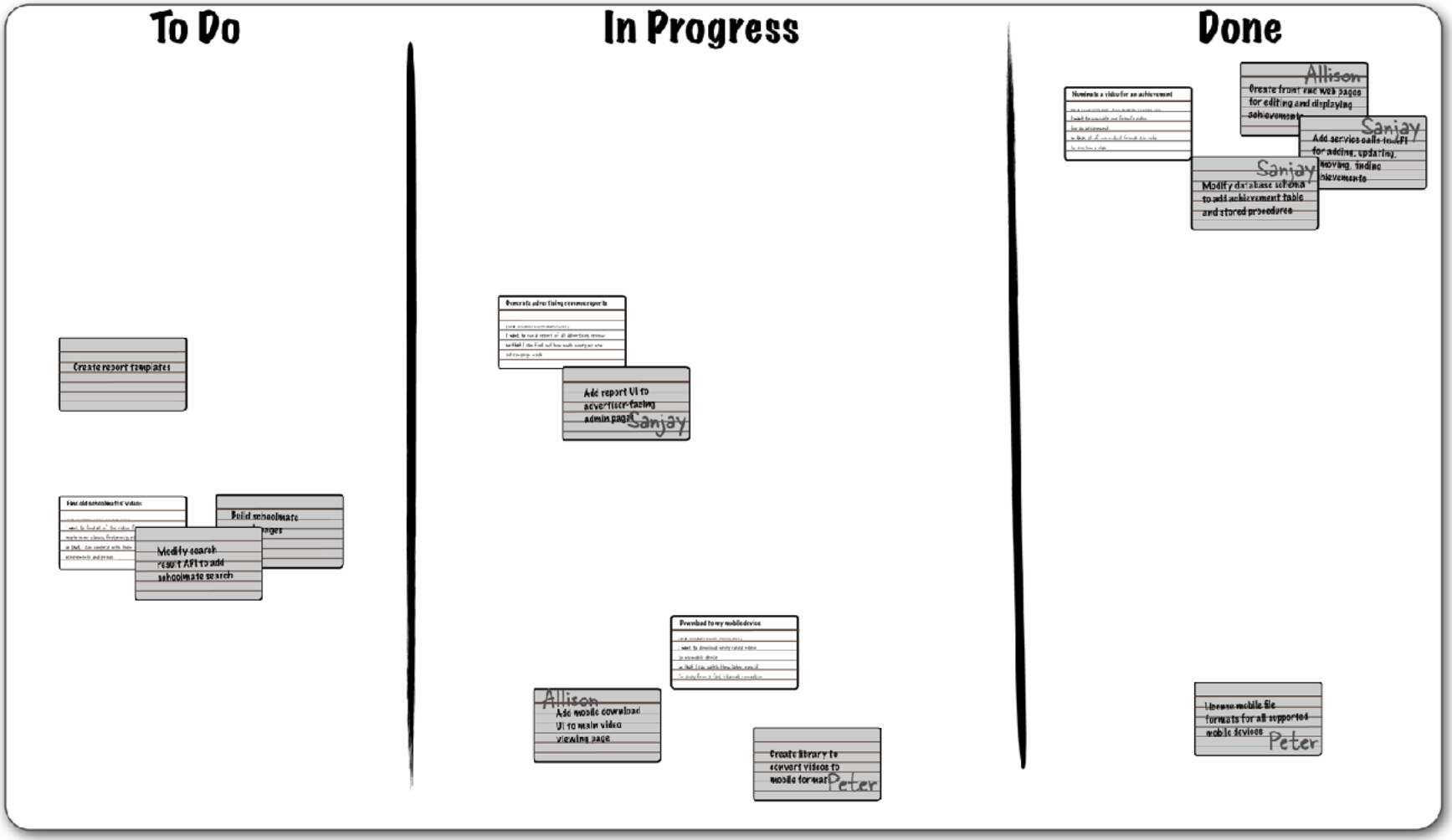
Planning and Running a Sprint



Planning and Running a Sprint

- As the sprint rolls along, the team moves tasks from **"To Do" to "In Progress" to the "Done" column**.
- It's pretty common for team members to discover that they need to do additional **tasks to finish a story**.
- Once a team member **finishes the final task for a story**, he pulls the story card off of the **task board**, verifies that all of the conditions of satisfaction are completed, and moves it to the **"Done" column** so that it's grouped with all of its tasks.
- If he discovers that one of the conditions of satisfaction hasn't been met, then there's still work to be done—so he moves the story back to the **"In Progress" column** and adds tasks to complete that work to the **"To Do" column**.

Planning and Running a Sprint



Planning and Running a Sprint

- The sprint is done when the **timebox expires**.
- This means that there might still be story and task cards left in the **"To Do" and "In Progress"** columns.
- The stories go back to the **product backlog** so they can be prioritized in the next **sprint planning session**, but the team can reduce the number of points based on the tasks that remain.
- The team does not get credit for completing a story until its card and all of its task cards are moved to the **"Done" column**.

Generally Accepted Scrum Practices

- There are many **great practices** that teams use to improve how they use Scrum.
- Cohn termed **Generally Accepted Scrum Practices** (GASPs)
- Many teams find that their **Daily Scrum meetings** are more effective when held as a standup meeting, where everyone stands until the meeting is **complete**. This is not a core Scrum practice, but it's **accepted by many Scrum teams**.