# BUILD GREAT TEAMS
# THAT ALWAYS DELIVER
# AWESOME PRODUCTS

## Certified Scrum Master (CSM®) Workshop

**Version: 4.3 (January 2021)**

**NAME:**

**Trainer**


# VIJAY BANDARU

# Meet Your Trainer


**MY VISION**

Make Learning SPECIAL

- **S** – Simple
- **P** – Practical
- **E** – Effective
- **C** – Collaborative
- **I** – Innovative
- **A** – Adaptive
- **L** – Long-lasting

- ✓ International Trainer
- ✓ Training from the Back of the Room (TBR) certified
- ✓ Certified Agile Coach
- ✓ Experience in Large Scale Agile Transformation
- ✓ Received "Best Trainer" multiple times
- ✓ Volunteer, Speaker, and Organizer
- ✓ Published Several Articles
- ✓ Created "WONDER" Coaching Model

- My previous Trainings Videos, Photos and Feedback at: [www.facebook.com/learnovative](http://www.facebook.com/learnovative)

www.vijaybandaru.com
www.learnovative.com
www.linkedin.com/in/vijaybandaru
vkbandaru@yahoo.com
+91 – 98480-32144

Training is not just my **PROFESSION,** it is my **PASSION**

I hope you enjoy the learning in my class and wish you good luck !!

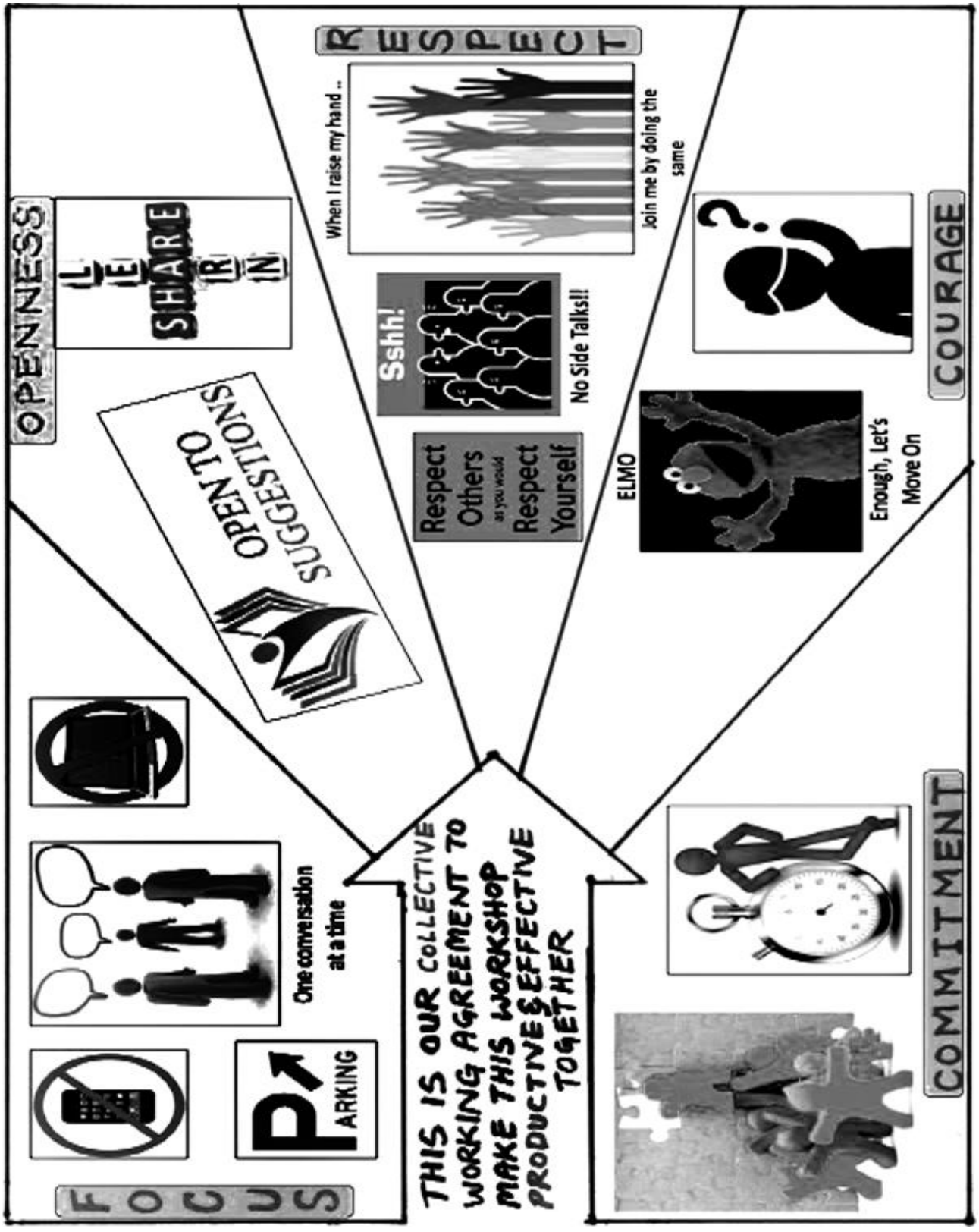
## WHAT'S OUR PLAN FOR THESE TWO DAYS?

# *Planning is Everything, Plans are nothing*

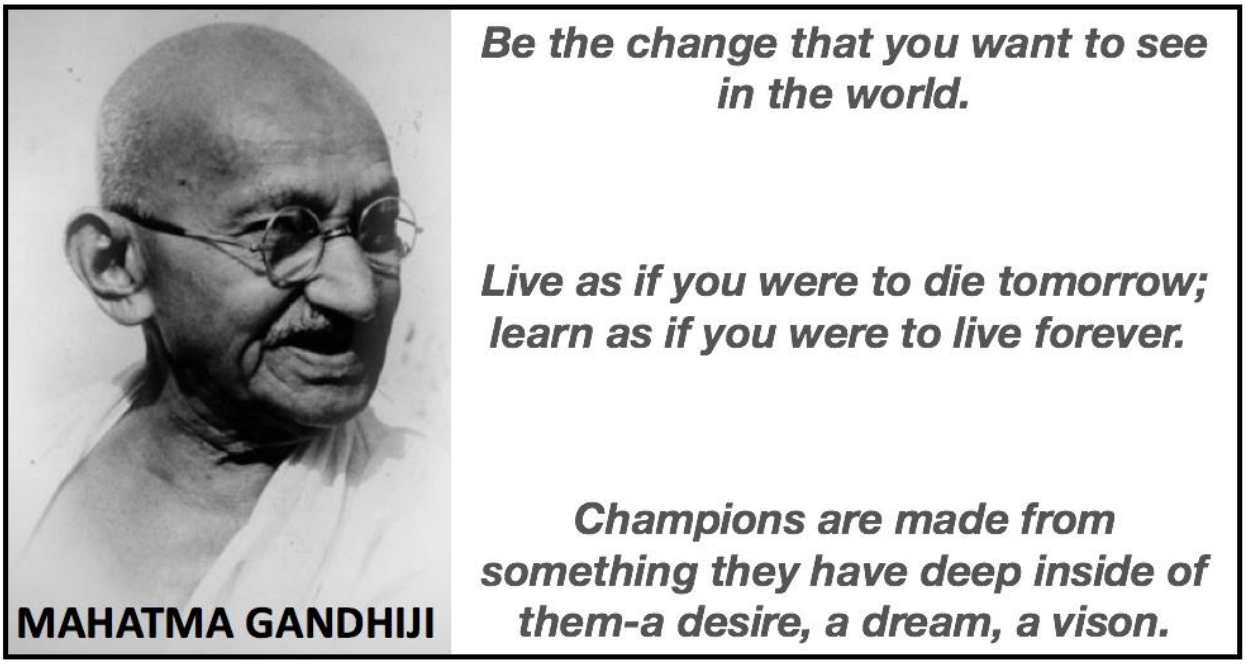
Let us learn "Goal" oriented,

with periodic inspect and adapt ☺

# Our working agreement



# Do you agree?




"Coming together is a beginning, staying together is progress, and working together is success." - *Henry Ford*

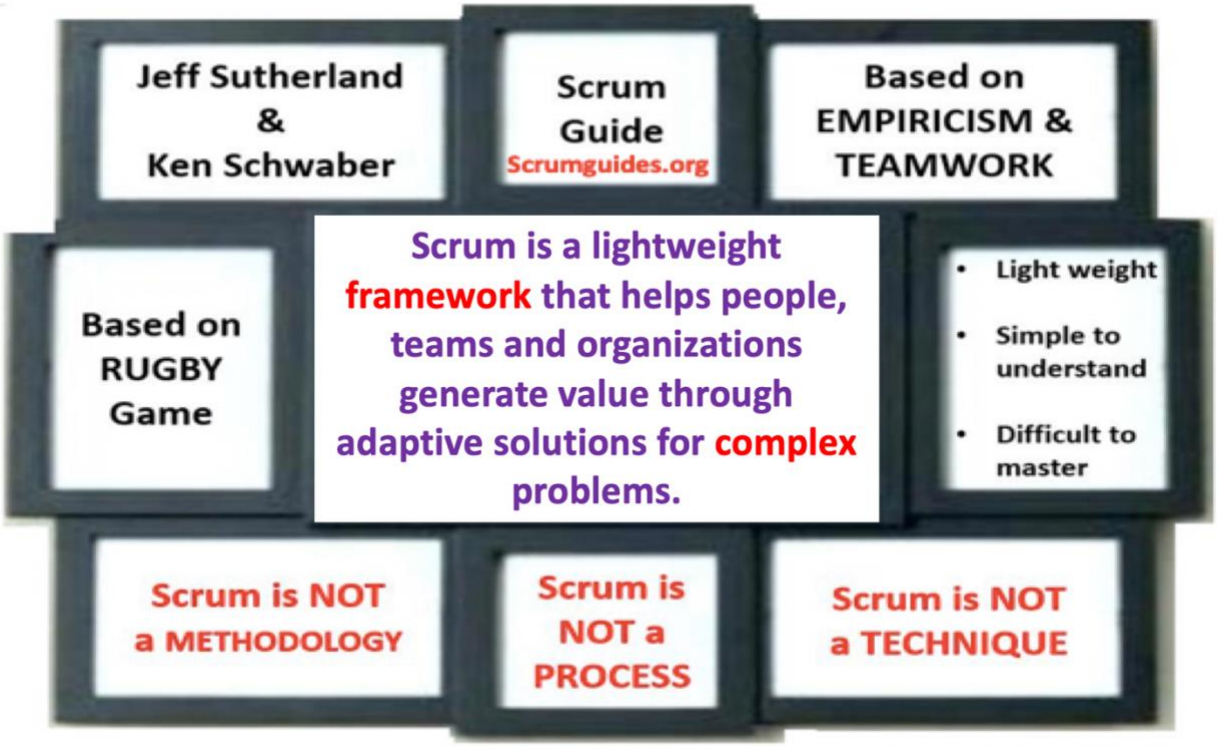
"Talent wins games, but teamwork and intelligence win championships." - *Michael Jordan*

"Teamwork is the ability to work together toward a common vision. The ability to direct individual accomplishments toward organizational objectives. It is the fuel that allows common people to attain uncommon results." - *Andrew Carnegie*

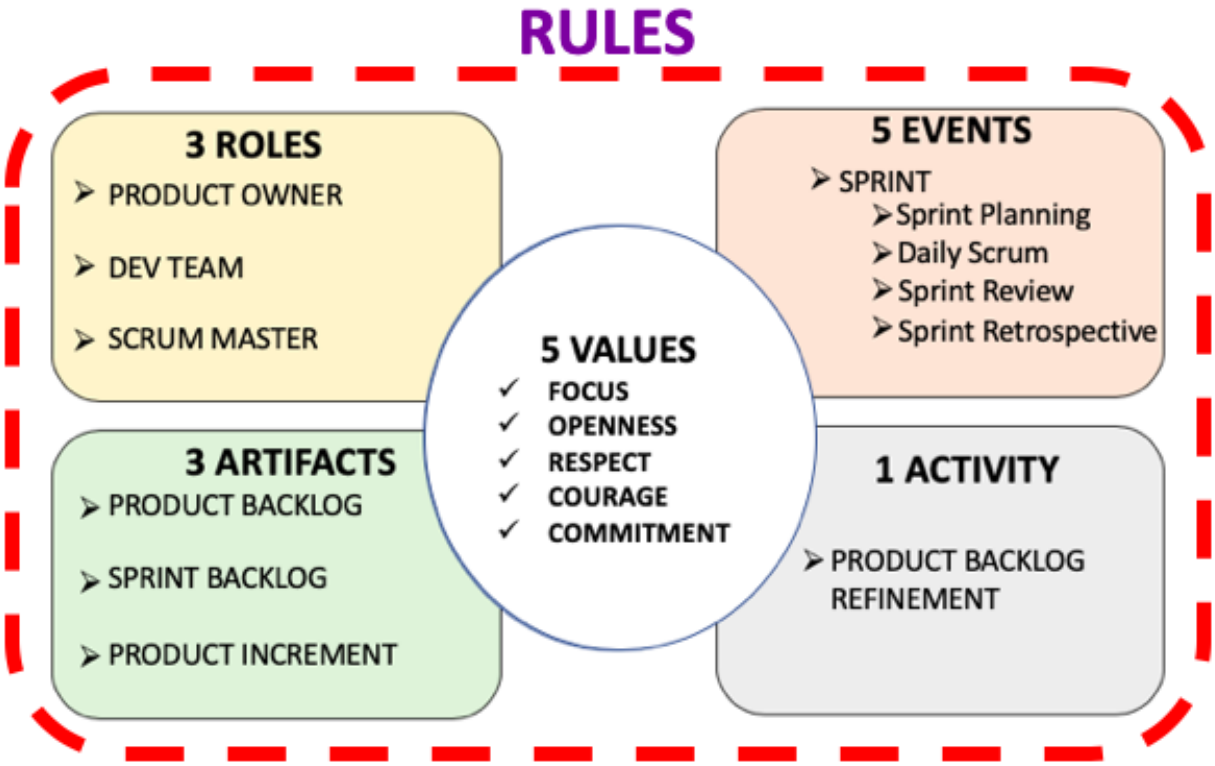
# Introduction To Scrum

*An iterative & incremental framework*

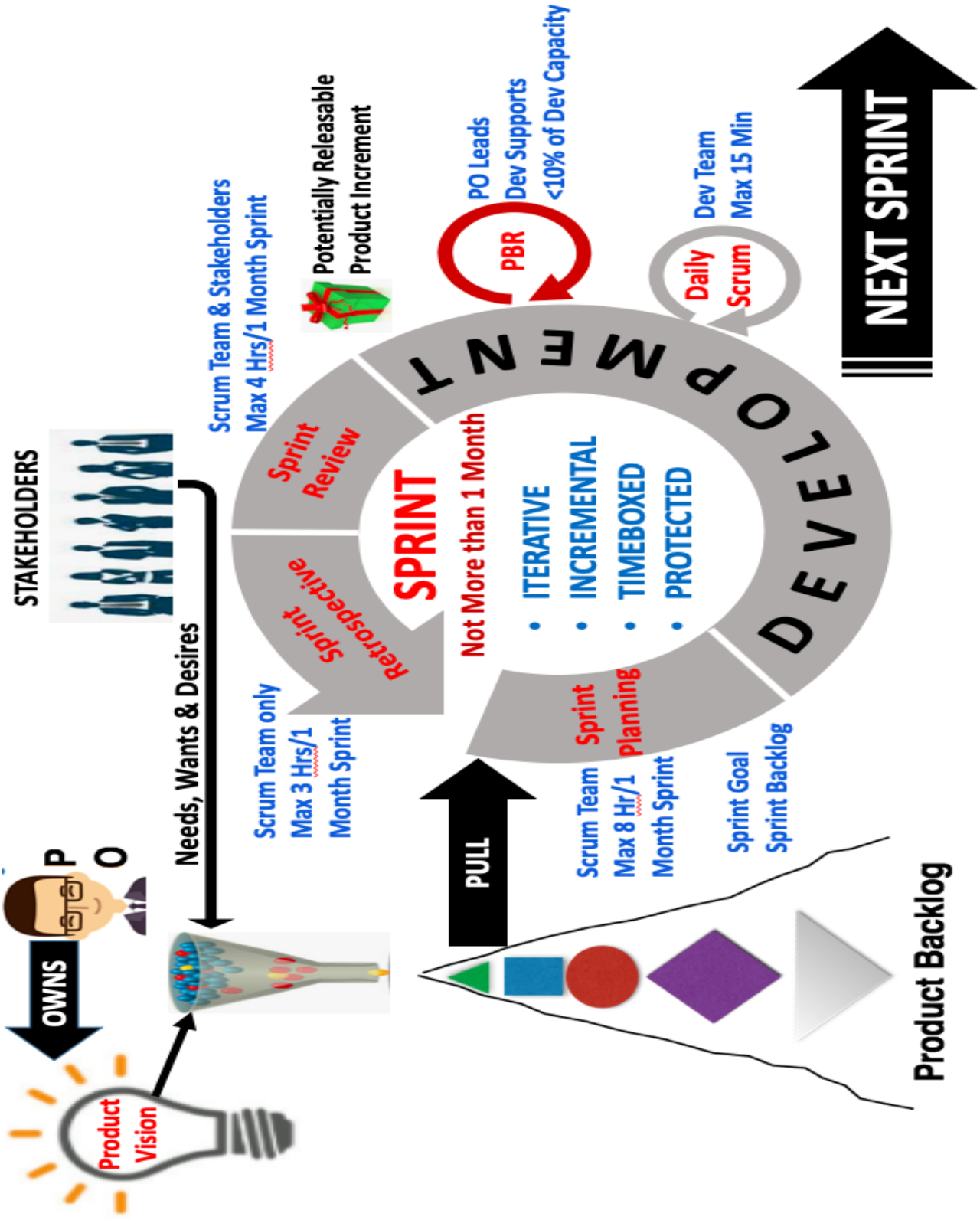
## Scrum - Introduction



## Scrum framework - Overview



## Scrum Framework Flow



## Tracking the Workshop Progress using Burndown/up

**Burndown helps to track the work remaining and the Sprint level Burndown will be updated by Development Team as and when they complete an item during the Sprint.**

**Y – Axis is work pulled into the Sprint and X – Axis is the Sprint duration**

### Sprint – 1

### Sprint – 2

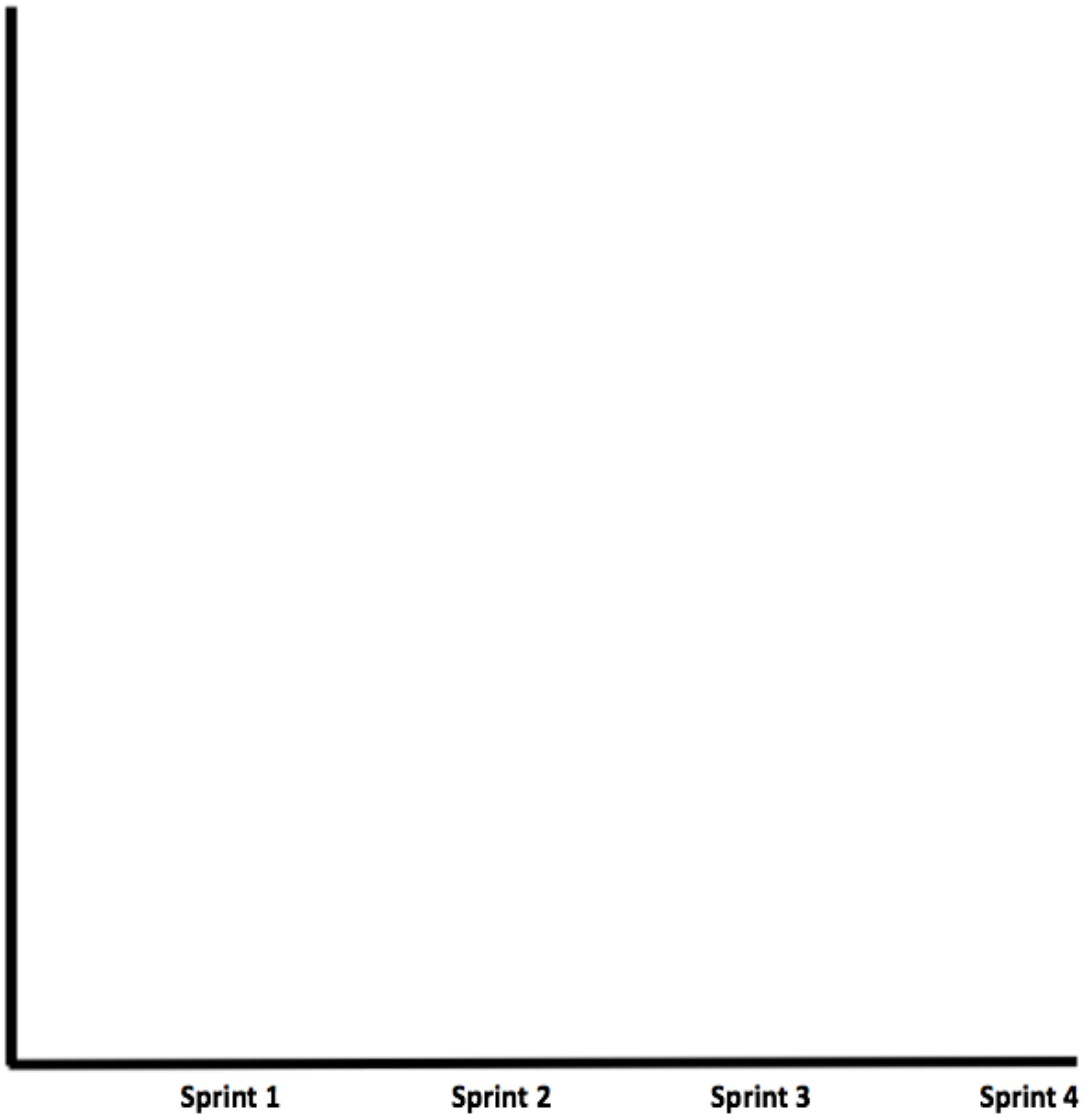
## Sprint – 3

## Sprint – 4

## Overall progress tracking of workshop using Burnup chart

**Burnup chart helps to track the work done and the Release level Burnup chart will be updated by Product Owner at the end of every Sprint.**

**Y – Axis is the Total work of the Project and X – Axis is the Project Duration (# of Sprints)**




| Sprint # | Best Velocity | Average Velocity | Worst Velocity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

# Agile Introduction

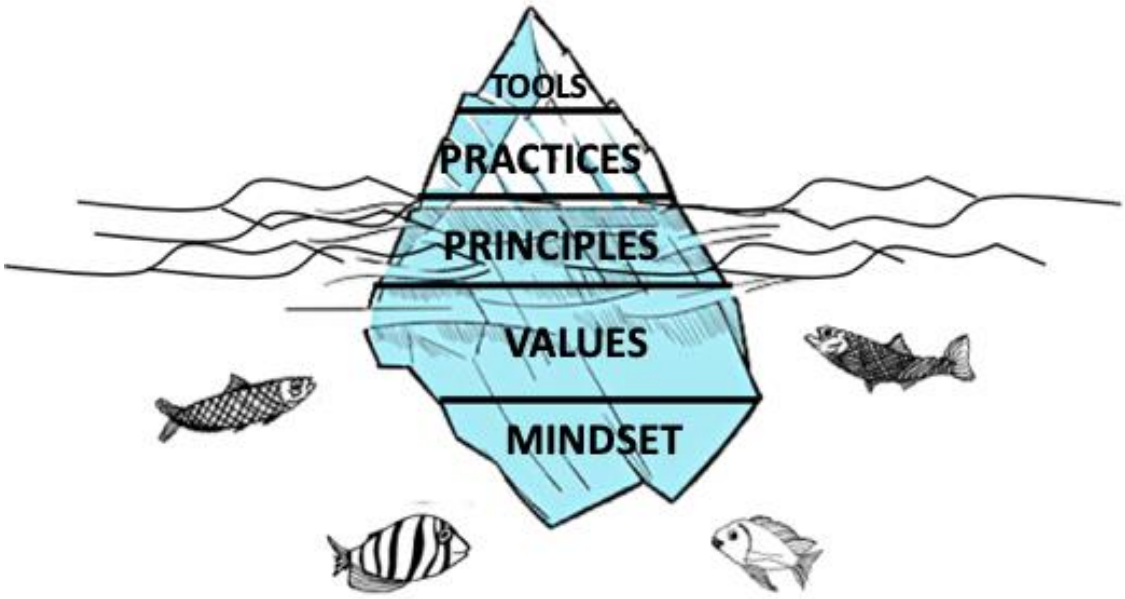
*What & Why*

## Agile Introduction

### Activity:

**Based on your experience so far with waterfall projects, recollect what were all the challenges you faced during the water project. List all the challenges at your table.**

## Agile is:



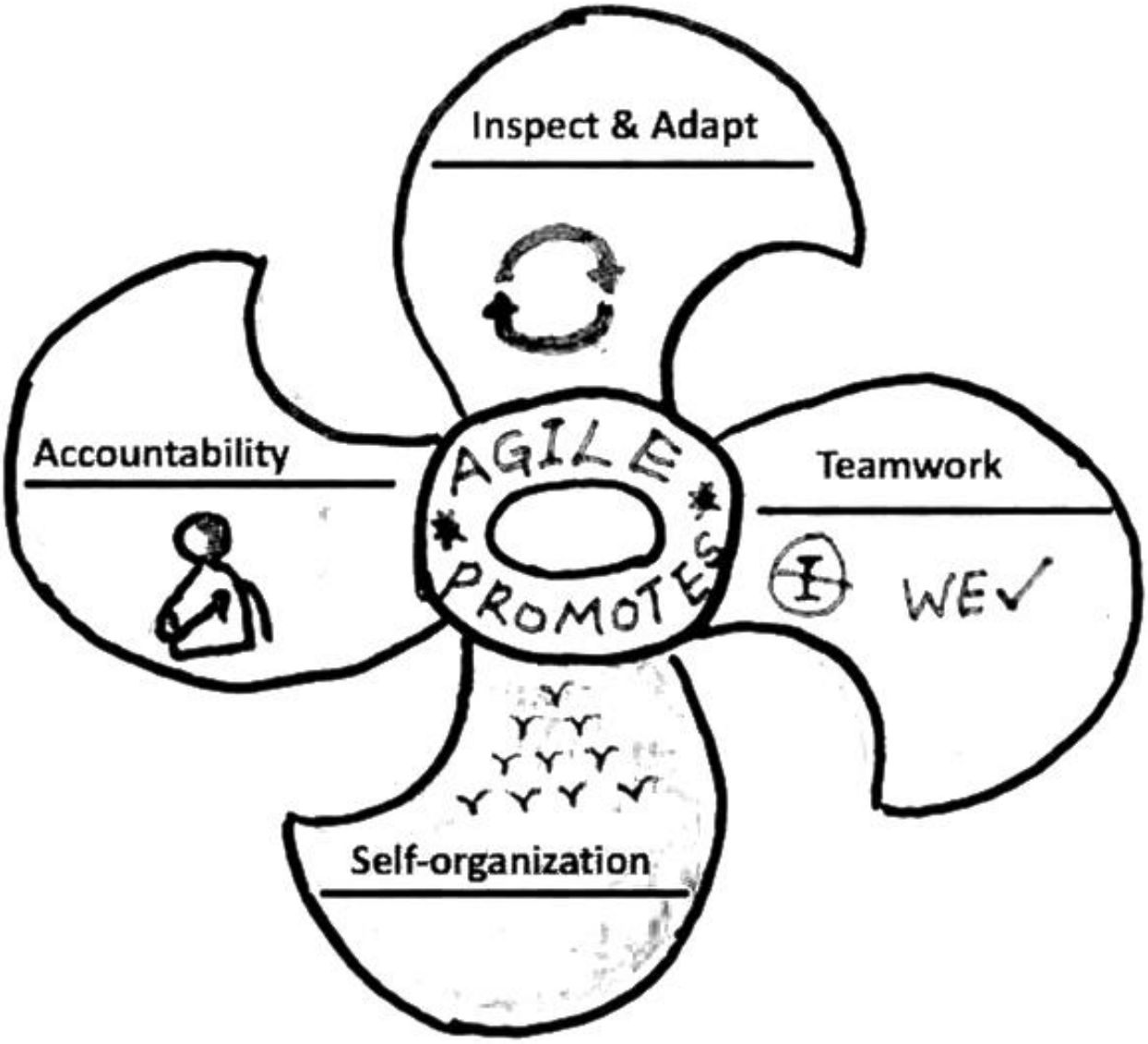

SPACE FOR NOTES

Map these words to the above iceberg into the 5 sections separated by a line.

***PRACTICES, VALUES, MINDSET, TOOLS, PRINCIPLES***

### Activity: Value or Waste?

## Agile generally promotes:




***Self-organization is the emergence of PATTERN and ORDER in a system by INTERNAL processes, rather than EXTERNAL constraints or forces*** [Encyclopedia of Ecology 2008]

SPACE FOR NOTES

**Agility is the ability to deliver customer ____VALUE____ frequently and consistently, while dealing with inherent project unpredictability and dynamism by recognizing and adapting to ____CHANGE____**

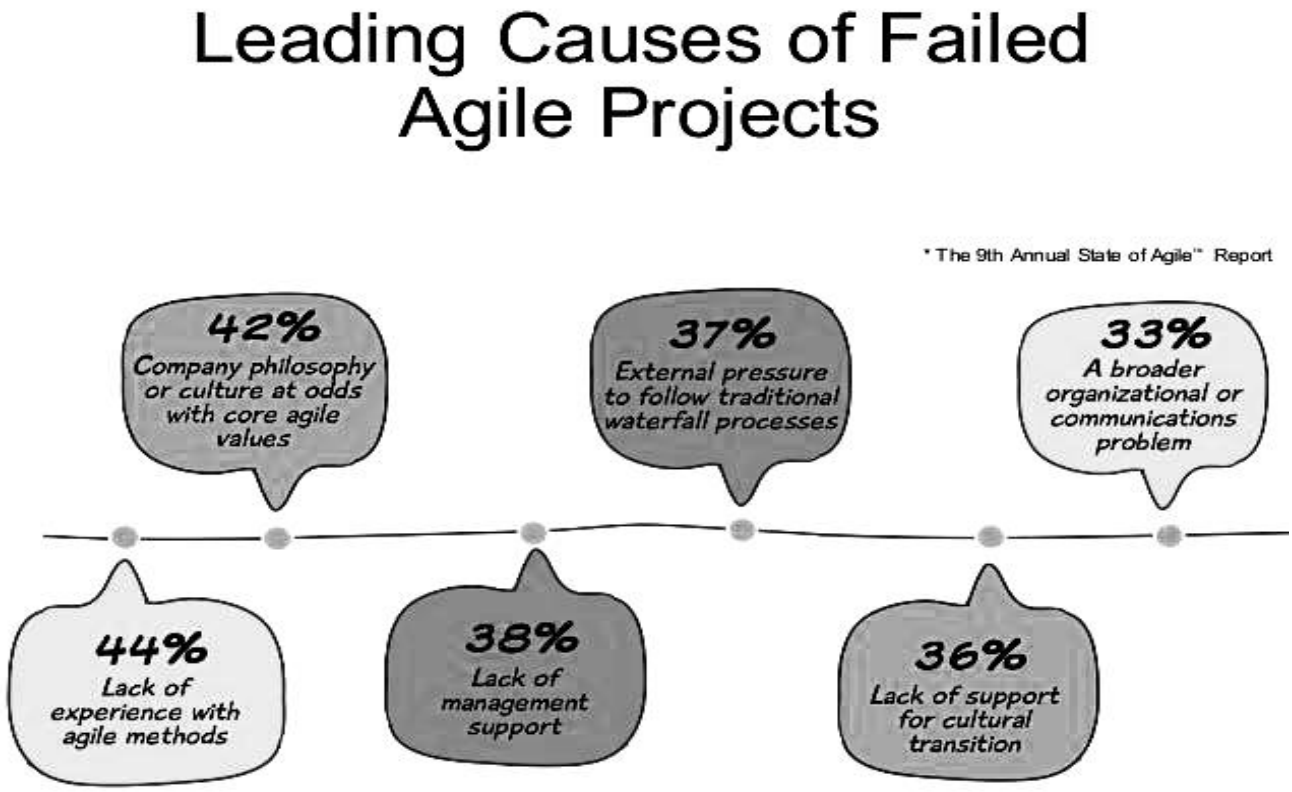
Frequent delivery helps you to receive early *FEEDBACK (which helps you to reduce cost of change)*

Consistent delivery helps you to increase *ROI*

Iterative (repetitive) way of delivery improves the *PROCESS*

Incremental (add on to) way of delivery improves the *PRODUCT*

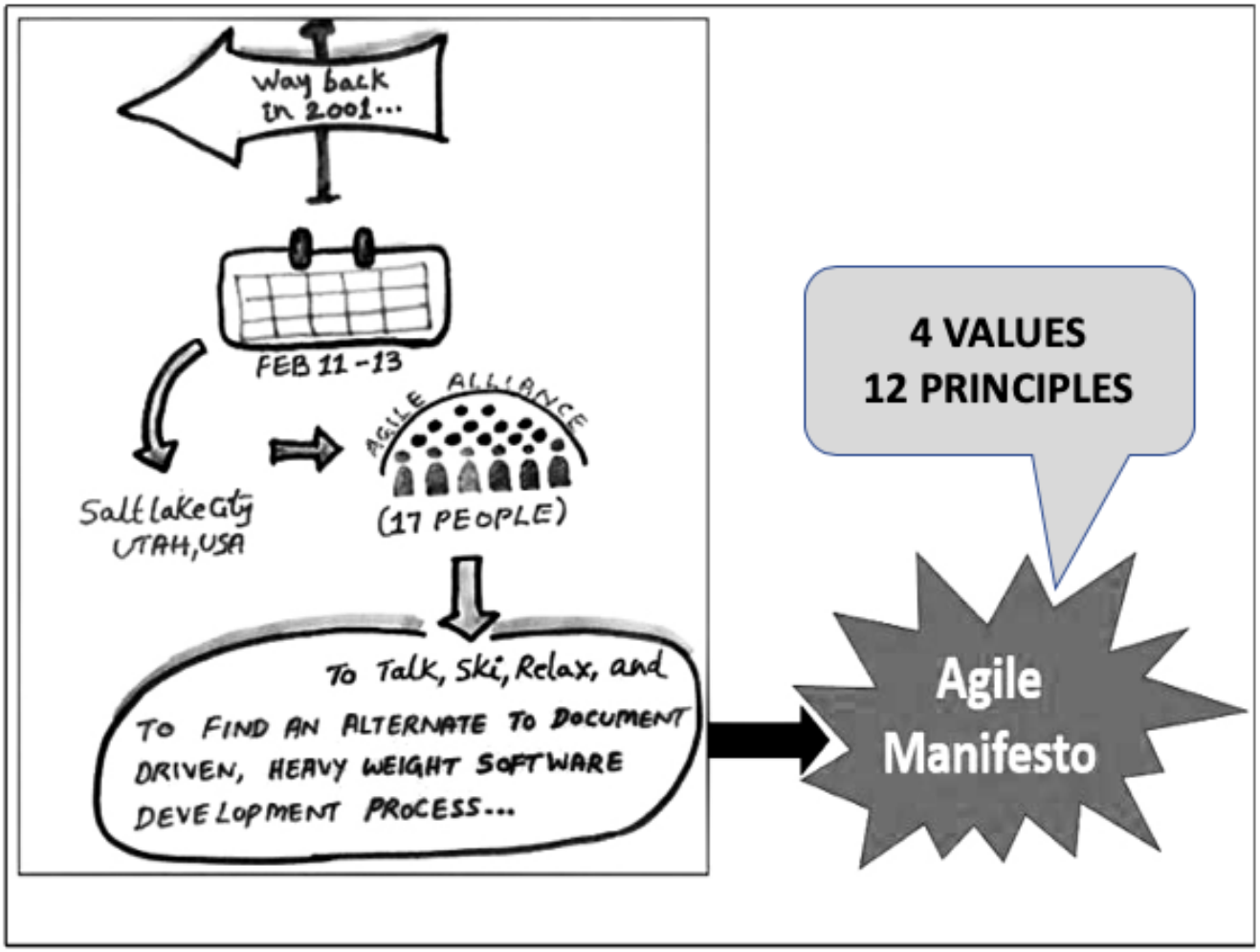
***Agile is the MEANS, not the END !***



# Agile Manifesto

## *Agile Values & Principles*

## Agile Manifesto:




SPACE FOR NOTES

## Agile Manifesto 4 Values

**We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:**

Individuals ***and Interactions*** ***OVER*** ***Processes and Tools***

Working ***Software*** ***OVER*** ***Comprehensive documentation***

Customer ***collaboration*** ***OVER*** ***Contract negotiation***

Responding ***to Change*** ***OVER*** ***Following a plan***

**While there is a VALUE in the items on the right side,**

**we VALUE the items on the left more!**

**Read the Blog:** What level of "Documentation" is required for Agile projects?

## Most of the time Processes beat the commonsense !!

# 12 Agile Principles

## Activity: Pair learn

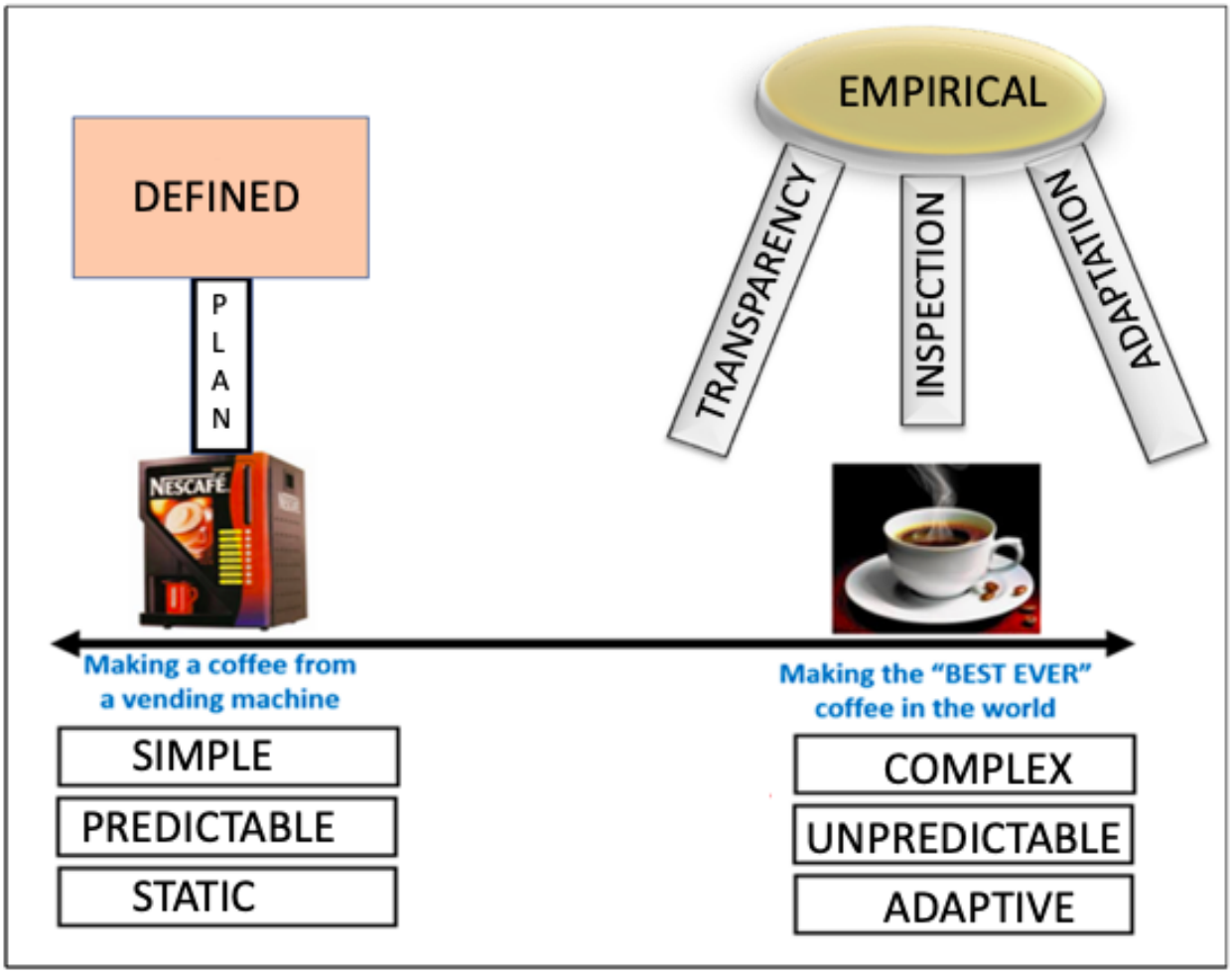
| Agile Principle | Explanation |
|---|---|
| 1. **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.** | - Early delivery helps receiving early feedback<br>- Continuous delivery helps customer receive ROI |
| 2. **Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.** | - Helps to improve flexibility of scope changes during the project so customer will be happy (Business agility) |
| 3. **Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.** | - "Time is the enemy of Transparency"<br>- Shorter duration reduces cost of change through receiving early feedback |
| 4. **Business people and developers must work together daily throughout the project.** | - Helps short circuiting feedback loops between business and development teams<br>- Improves collaboration, trust and transparency between business and development teams |
| 5. **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.** | - Helps create "Safe to fail" environment so that the people will have higher level of accountability<br>- Teams can do more experiments and learn more |
| 6. **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.** | - Helps reducing to-and-fro loops of information exchange<br>- Shared documentation does not guarantee shared understanding |
| 7. **Working software is the primary measure of progress.** | - Helps measure progress based on value instead of milestone like in waterfall<br>- It helps to come up with better forecast |
| 8. **Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.** | - Maintaining constant pace of delivery throughout will help creating regularity<br>- Sustainable development helps achieving predictability |
| 9. **Continuous attention to technical excellence and good design enhances agility.** | - Helps attaining technical agility so that it helps to achieve business agility (Principle #2)<br>- Helps to create emerging architecture |
| 10. **Simplicity--the art of maximizing the amount of work not done--is essential.** | - Focus on OUTCOME not OUTPUT<br>- Means, work on only high value, high importance, high priority items |
| 11. **The best architectures, requirements, and designs emerge from self-organizing teams.** | - Helps decision making decentralized so that decisions can be made faster<br>- It also improves the accountability at the teams who make those decisions |
| 12. **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.** | - Inspect and adapt periodically to come up with improvements to become better |

## Discuss: How these principles address the waterfall challenges

# Process Controls

## *Defined VS Empirical*

## Process Controls:

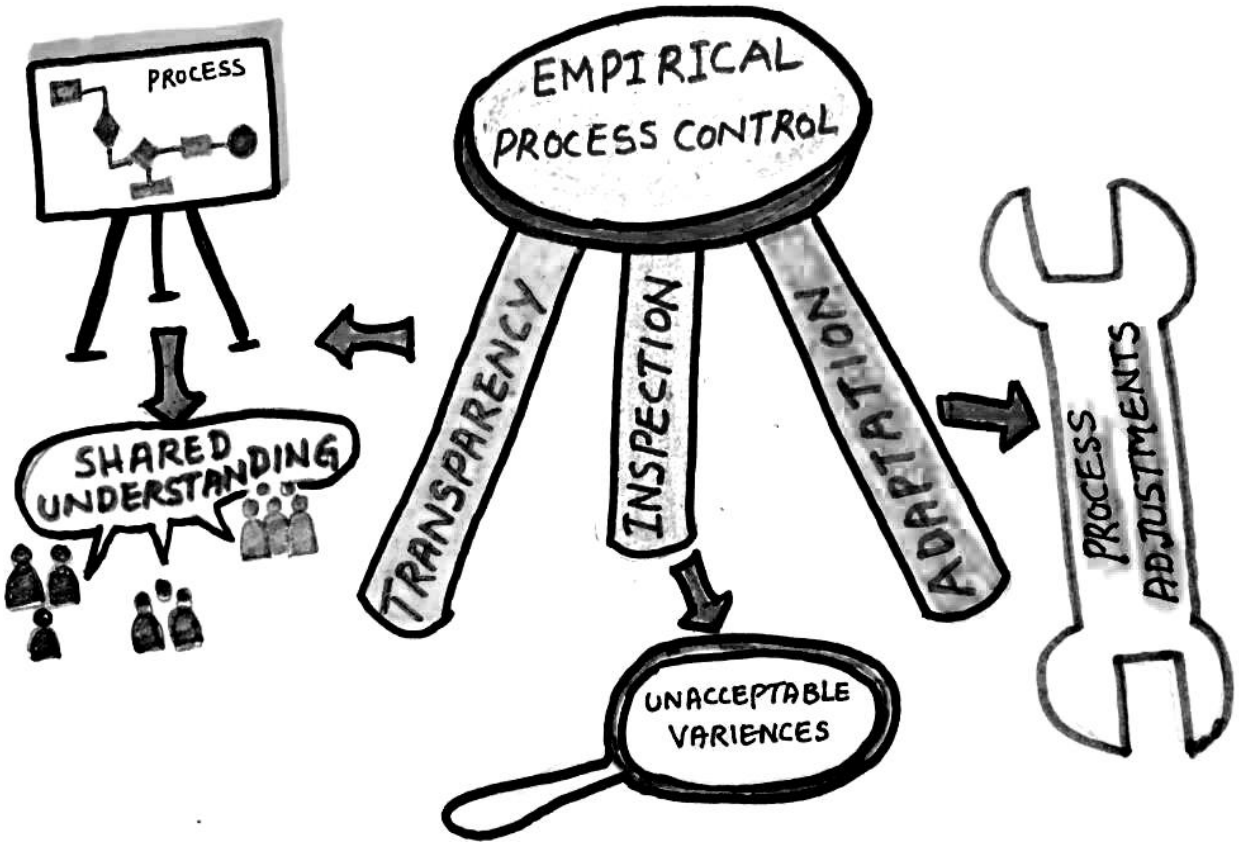



*Map the items in the center to the boxes on both sides based on the problem statements given*

**Simple Environment: KNOWN is more than UNKNOWN**

**Complex Environment: UNKNOWN is more than KNOWN**

SPACE FOR NOTES




## Empiricism asserts that "KNOWLEDGE" comes from "EXPERIENCE". The "DECISIONS" are made based on what is "KNOWN"

**Transparency**: Visibility/Openness. The various aspects of process that affect the 'outcome' must be visible to those controlling the process. What is done is visible!

**Inspection**: The various aspects of the process must be inspected frequently enough that unacceptable variances in the process can be detected.

**Adaptation**: If the inspection reveals that one or more aspects of the process are outside the acceptable limits and that the resulting product will be unacceptable, the inspector must adjust the process, or the material being processed.

The projects will become highly reliable if the team's 'ability to adapt to the environment' gets better (rather than follow a prescribed path).

**Exercise: Identify which of the following are based on Defined / Empiricism and why:**

**DEFINED**

**Baking a Cake**

**EMPIRICAL**

**Cooking (Biryani)**

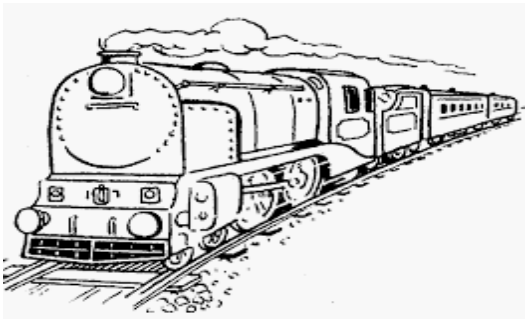
**EMPIRICAL**

**Bringing up children**

**Depends (Based on Simple/Complex)**

**Trekking up a hill**

**DEFINED**



**Train journey**

**EMPIRICAL**

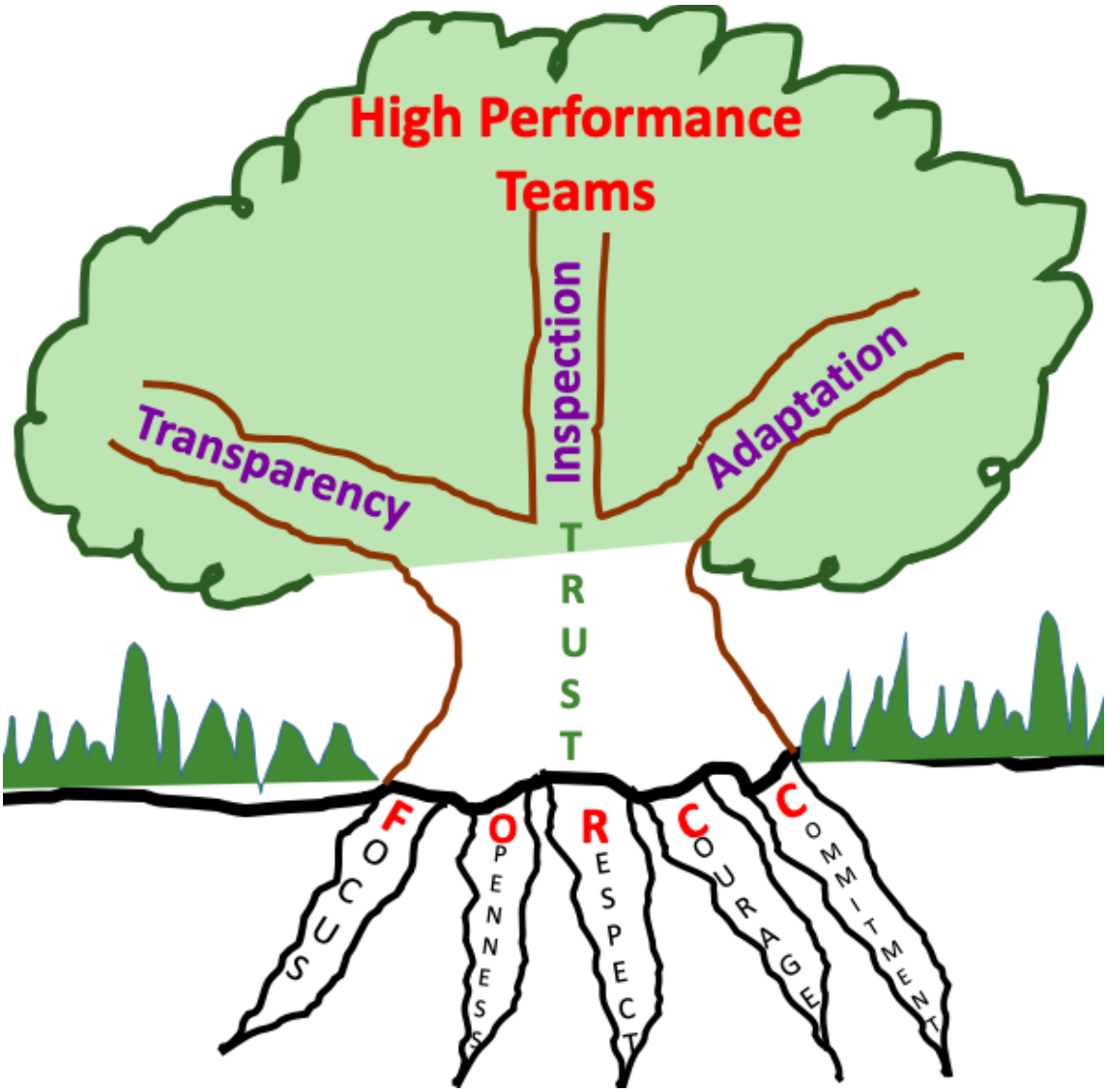
**Helicopter journey**

## *What about Product Development?*

# Scrum Values

***The ROOTS that Build TRUST***

## Scrum Values:

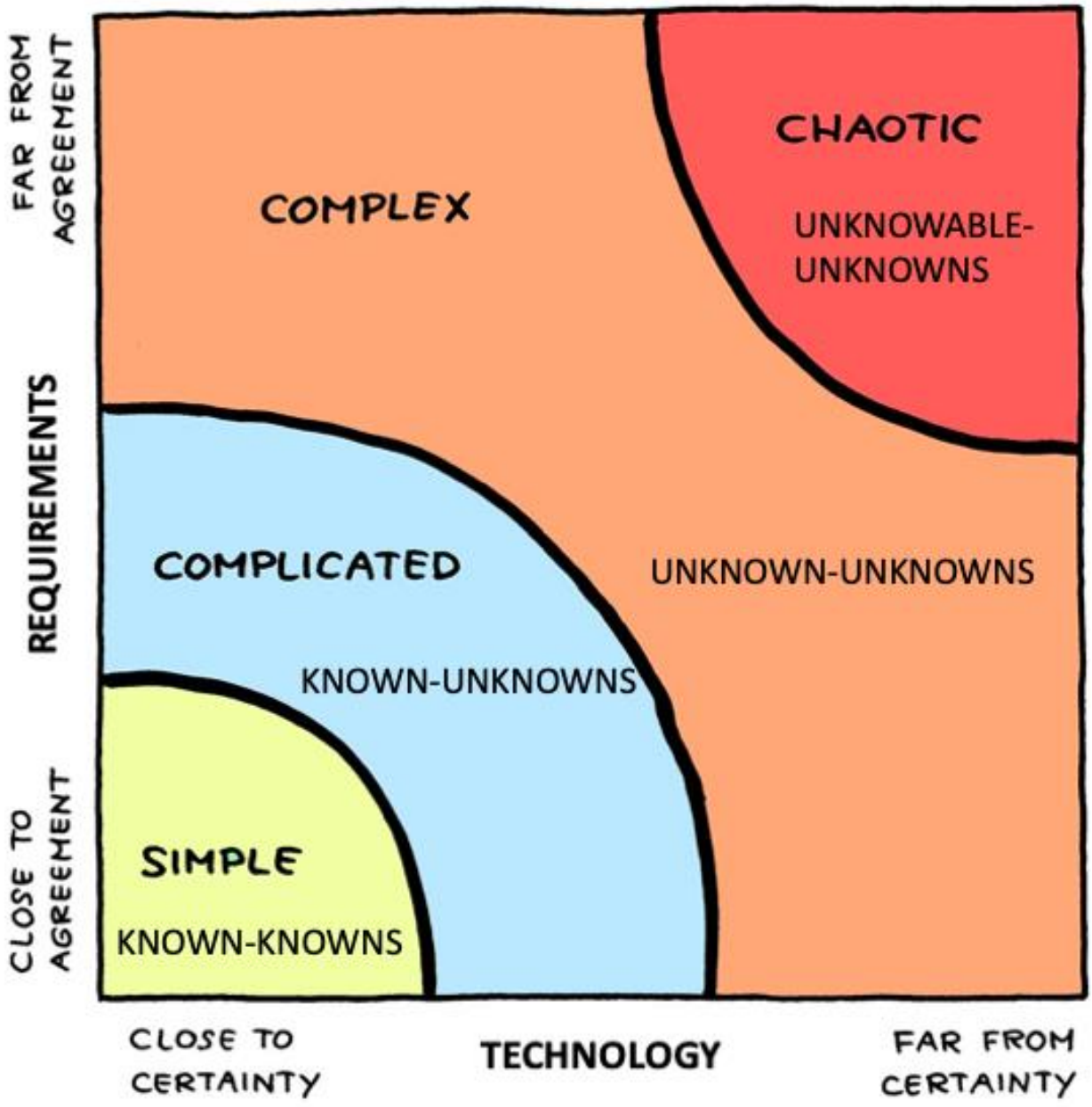



*It asks you to **COMMIT** to a goal and then provides you with the authority to meet those commitments. Scrum insists that you **FOCUS** all your efforts on the work you're committed to and ignore anything else. **OPENNESS** is promoted by the fact that everything about a Scrum project is visible to everyone. Scrum tenets acknowledge that the diversity of team members' background and experience adds value to your project. Finally, Scrum asks you to have the **COURAGE** to commit, to act, to be open and to expect **RESPECT**.*

**Activity: Identify the suitable Scrum values for the scenarios given below**

| SNo | Scenario description | Suitable Scrum Value |
|---|---|---|
| **1** | You are a development team and you just finished the Sprint planning. You started implementing the items in the Sprint backlog in such a way that you are working on one item at a time as a team to complete it before moving to the next item. | **Focus** |
| **2** | You are a product owner, you have received a new requirement from a key stakeholder which is not at all in line with the Product vision and does not add any value to the Product. You are saying "No" to the stakeholder. | **Courage** |
| **3** | You are a team and you are always keeping the Sprint backlog, burn down chart up-to-date always throughout the Sprint. | **Openness** |
| **4** | You are the development team and you are in a Sprint planning. Your Product Owner is trying to push more work than what you can achieve in the Sprint. You want to tell the same to the Product Owner. | **Courage** |
| **5** | You are a development team with a combination of senior experienced members and also a couple of college graduates. Your team is willing to explore and discuss the suggestions from the college graduates whenever they suggest any new ideas. | **Respect** |
| **6** | You are the Scrum master of a team and you identified that your team and Product owner are doing something not in line with the Scrum framework. You want to tell them to avoid that practice. | **Courage** |
| **7** | You are the Scrum master of a team. You have observed that your team is giving their best efforts to achieve the Sprint goal throughout the Sprint. | **Commitment** |
| **8** | A Development team tells during the Sprint that they are unaware of something and they need some help. | **Courage** |

## Where can you use Scrum?




**UNKNOWN-UNKNOWNS I KNOWN-KNOWNS I UNKNOWABLE-UNKNOWNS I KNOWN-UNKNOWNS**

Scrum works effectively in ***Complex*** environment. The reason being, Enhanced transparency, frequent inspect and adaption and validated learning helps to manage complexity.

## Agile Vs Scrum

Scrum is Agile but Agile is ***not*** ONLY Scrum.

Agile is an umbrella of frameworks as shown below.


**Agile is a state of "Being"**

**Scrum is a state of "Doing"**

SPACE FOR NOTES

# Sprint

*The HEART of the SCRUM*

## Sprint – The "Heart" of the Scrum

- A basic unit of development is called a "Sprint"
- Team produces Potentially Releasable Product Increment at the end of every Sprint
- Each Sprint has the Four Events in the order of occurrence, they are also timeboxed
- Sprint is timeboxed for not more than one month
- No gap between Sprints (as it impacts the regularity and focus)
- Fixed (same) duration improves predictability
- Each Sprint has a GOAL and a Design and PLAN
- During the Development:
    - Changes that impact the GOAL are not accepted
    - Quality goals cannot be changed
    - Development team composition cannot be changed

**"Scrum Team" collaborates to decide the Sprint Duration**

**Factors that help deciding the Sprint Duration:**

Development Team Size, Skill set, stability of Product Backlog, Overall Project Duration, Organizational barriers or influencers

### Sprint Cancellation

- **Abnormal Termination of Sprint before Timebox expires is called as Sprint Cancellation**
- **Only Product Owner can make the decision to cancel a Sprint**
- **Only reason for Sprint cancellation is, Sprint Goal becomes obsolete**

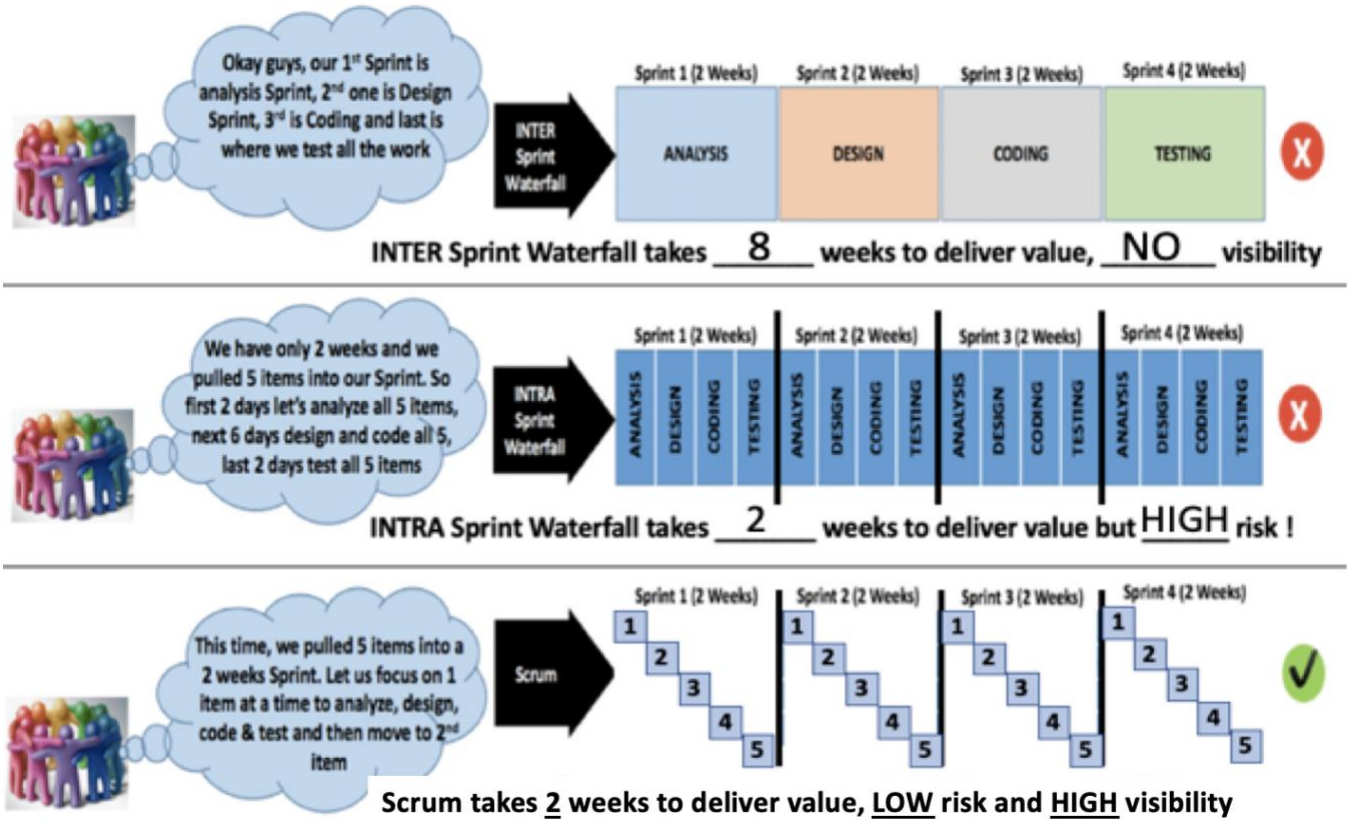
**What happens when a Sprint is cancelled?**

1. Review of the items (if there are any by the time of cancellation) that have value
2. Retrospective
3. Next Sprint planning

## Why iterative and incremental delivery?

- ***Minimizes*** **risk**
- ***Manageable*** **complexity**
- Clear ***visibility*** of working **increments**
- Constant ***collaboration*** with **customers**
- ***Useful & usable*** product **increment**
- ***Reduces*** **waste**
- ***Flexibility*** in managing **changes**
- ***Predictable*** pace of **delivery**
- ***Better*** **quality**
- ***Frequent*** and regular **feedback**
- ***High Value*** delivered **early**

## Beware of the way your Sprinting!



# Scrum Team

*The "Players" of Scrum*

## Scrum Team:

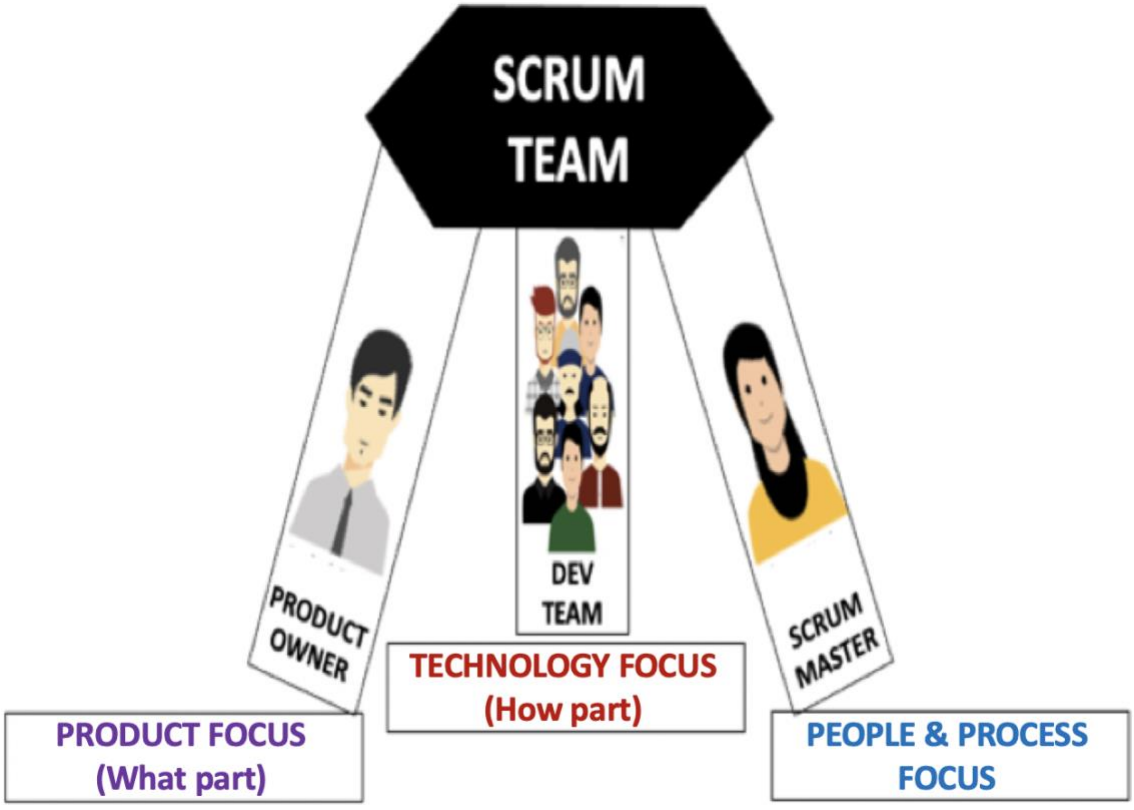
**"Scrum Team" consists of:**

- **Product Owner**
- **Developers** *(Formerly called as Development Team till 2017 Scrum Guide)*
- **Scrum Master**

**Each role has a Specific:** Purpose, Rights and Responsibilities

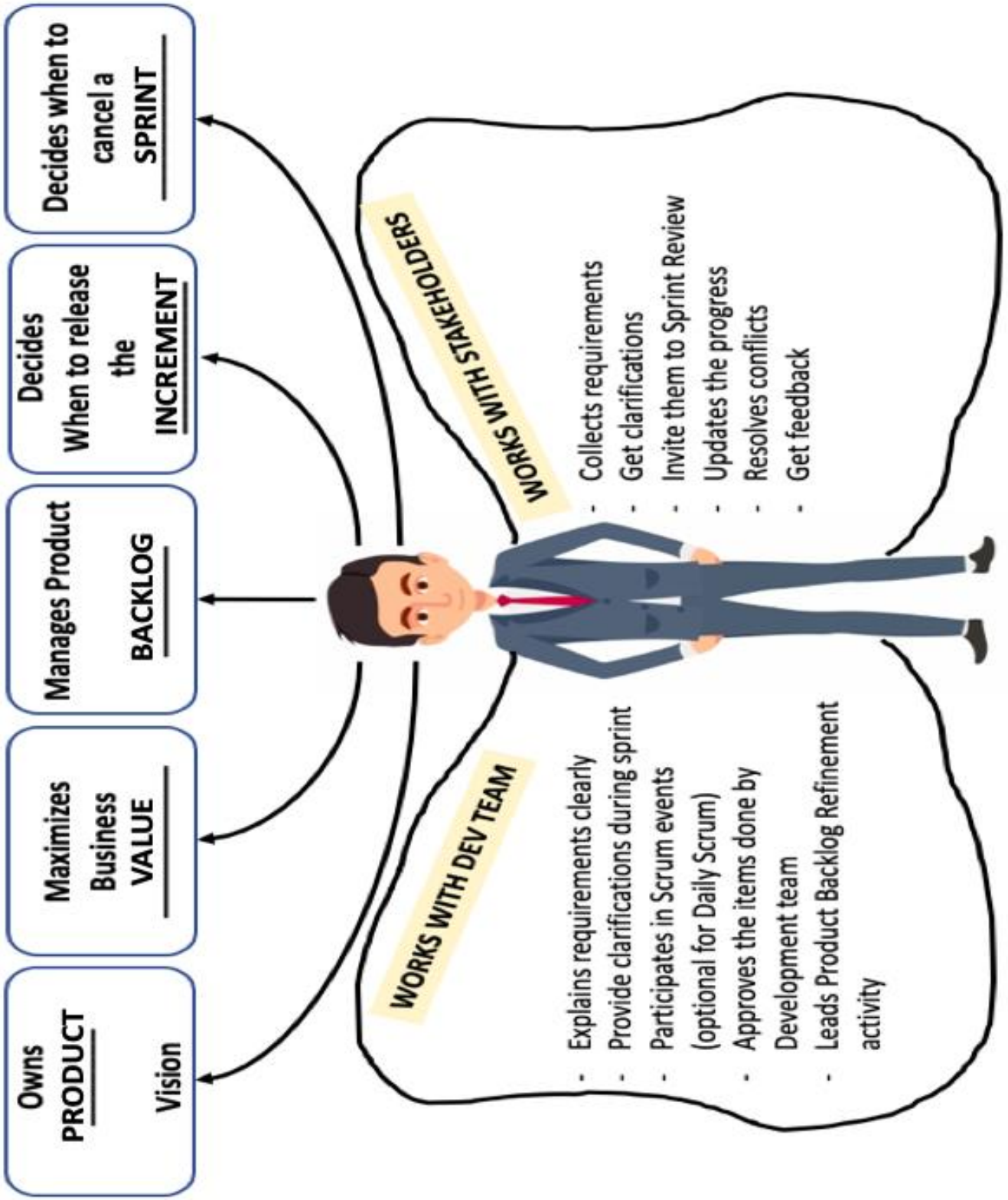
**Helps to optimize:** Productivity, Flexibility and Creativity

## Scrum Team Focus Areas:




**Scrum Team Size: Typically 10 or less members inclusive of Scrum Master and Product Owner**

## PRODUCT OWNER:



**Product Owner Rights and Responsibilities:**

- ONE person, not a COMMITTEE or GROUP
- Owns the Product VISION (represents ***WHY*** we build product and ***WHAT*** the desired end state is)
- Manages Product BACKLOG (Content, Availability & Ordering)
- Clearly expresses the Product Backlog items
- Ensure the Product backlog is visible and accessible to all
- Ensures Return on investment
- Works closely with Development team and Stakeholders
- Represents the Customer, not always the end customer
- Tracks overall product progress
- Reviews product backlog items and provide acceptance/feedback to Development team
- Provides clarifications to the Development team
- Leads Product Backlog Refinement
- Decides when to release the Product increment

**Activity: Circle the incorrect statements of the Product owner role in the below. (Answer: All Odd numbered boxes are incorrect)**

1. The Product Owner must tell the Development Team members how they must do their work

2. The Product Owner should decide when to release the product increment

3. The Product Owner must always be the end customer himself/herself

4. The Product Owner is the only one that can change the order of the product backlog items

5. The BEST Product Owners already worked as Project Managers earlier

6. The Product Owner must balance the needs and desires of different stakeholders

7. The Product Owner should decide the architecture of the product

8. Only the Product Owner can cancel the Sprint in case the Sprint goal is obsolete

9. The presence of Product Owner is <u>not</u> mandatory for Sprint Planning

10. To be more effective, the Product Owner collaborates with Development team and Stakeholders

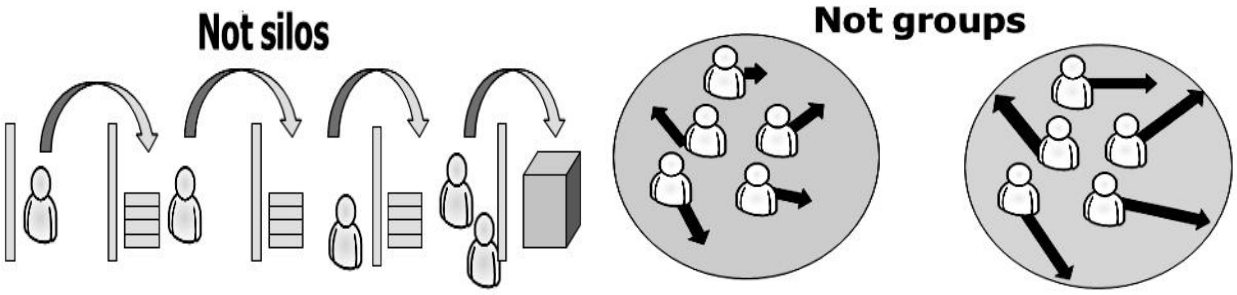
11. The Product Owner can blame the Development team in case he/she could not achieve the ROI

12. The Product Owner must not estimate the product backlog items and leave it to the Development team

13. The Product Owner can be a committee to share the responsibilities collectively

14. The Product Owner is responsible for defining what is the right product and what are the right features of that

## DEVELOPERS:


Not silos
Not groups

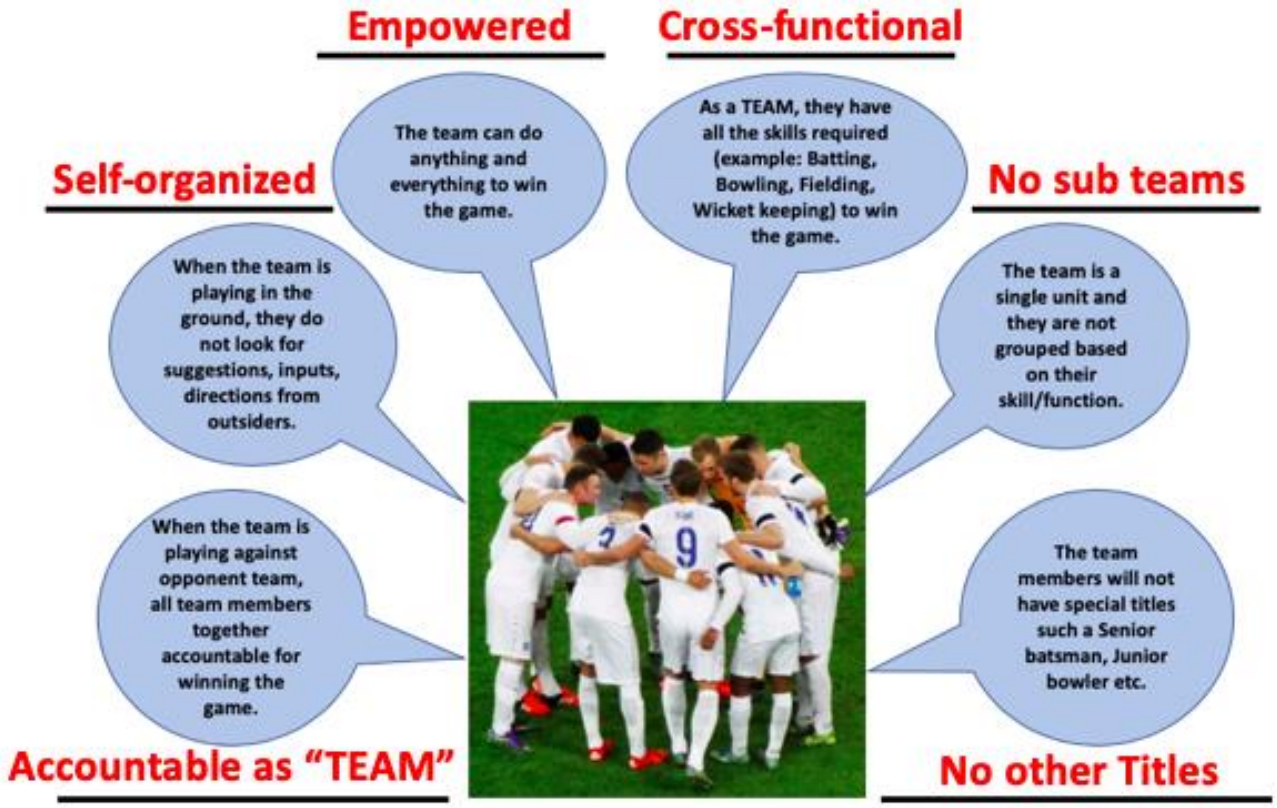

Together
Everyone
Achieves
More


Teams!

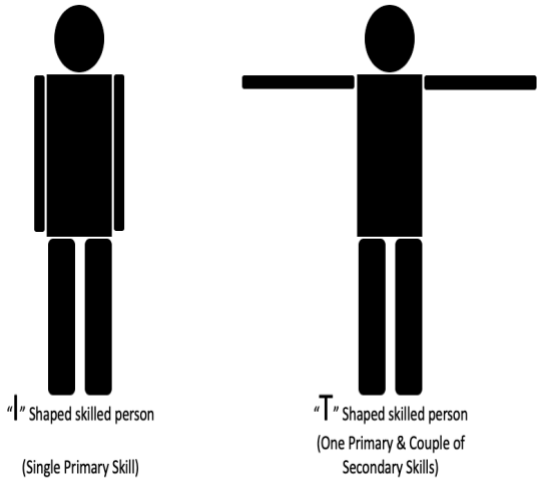

ME
influencers.w0rld
A TEAM IS NOT A GROUP OF PEOPLE
THAT WORK TOGETHER.
A TEAM IS A GROUP OF PEOPLE THAT
TRUST EACH OTHER.
WE

### Developers Characteristics:

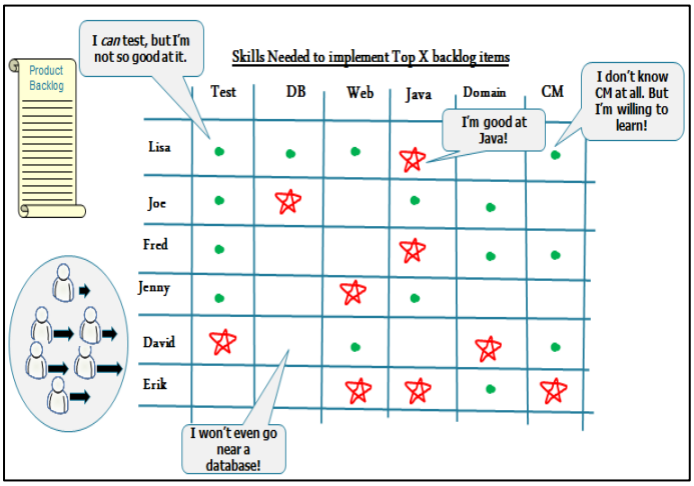
Let us consider Any Sports team (Example Cricket, Football).




**"T" shaped team members**



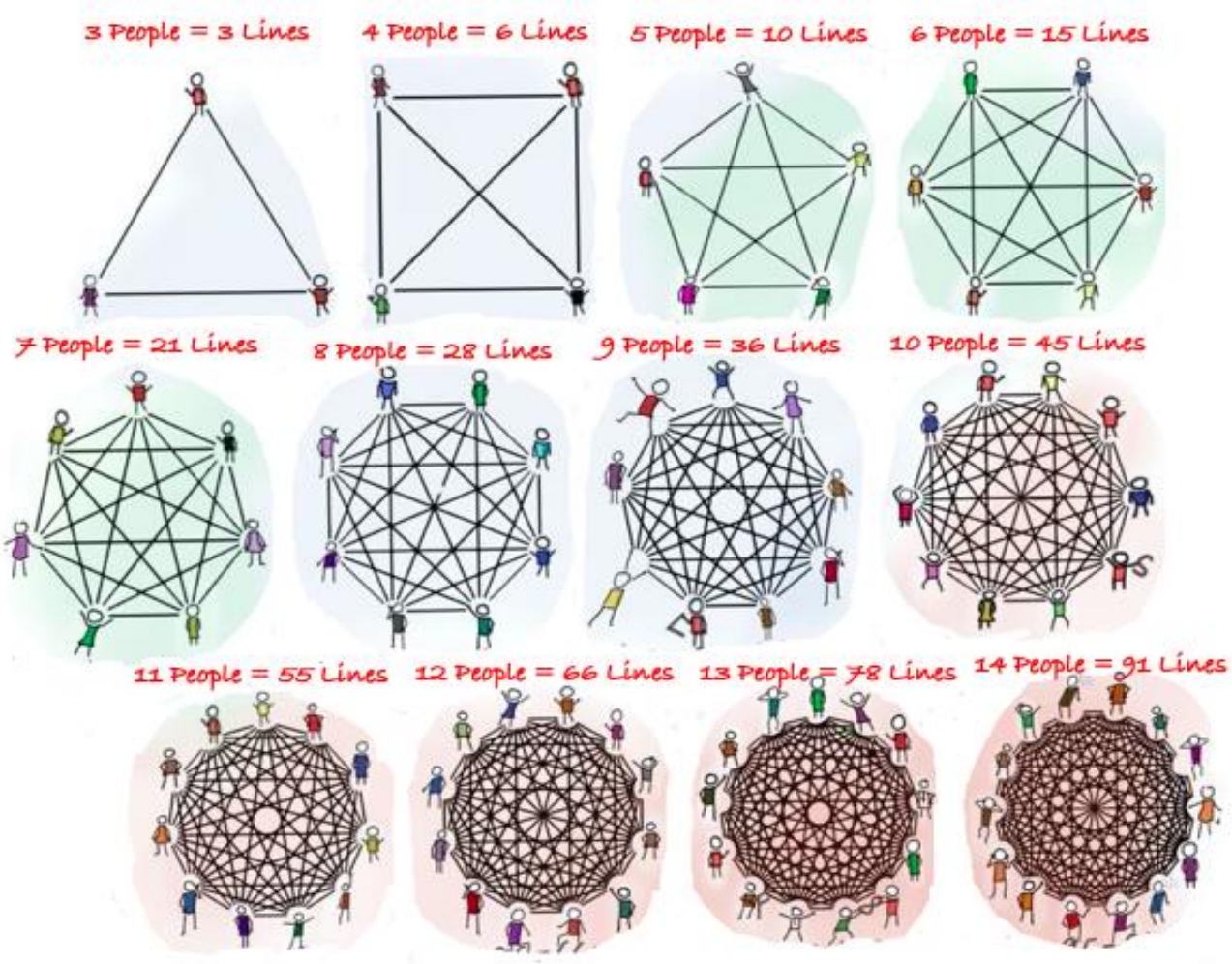

**How can you develop cross-functional team?**



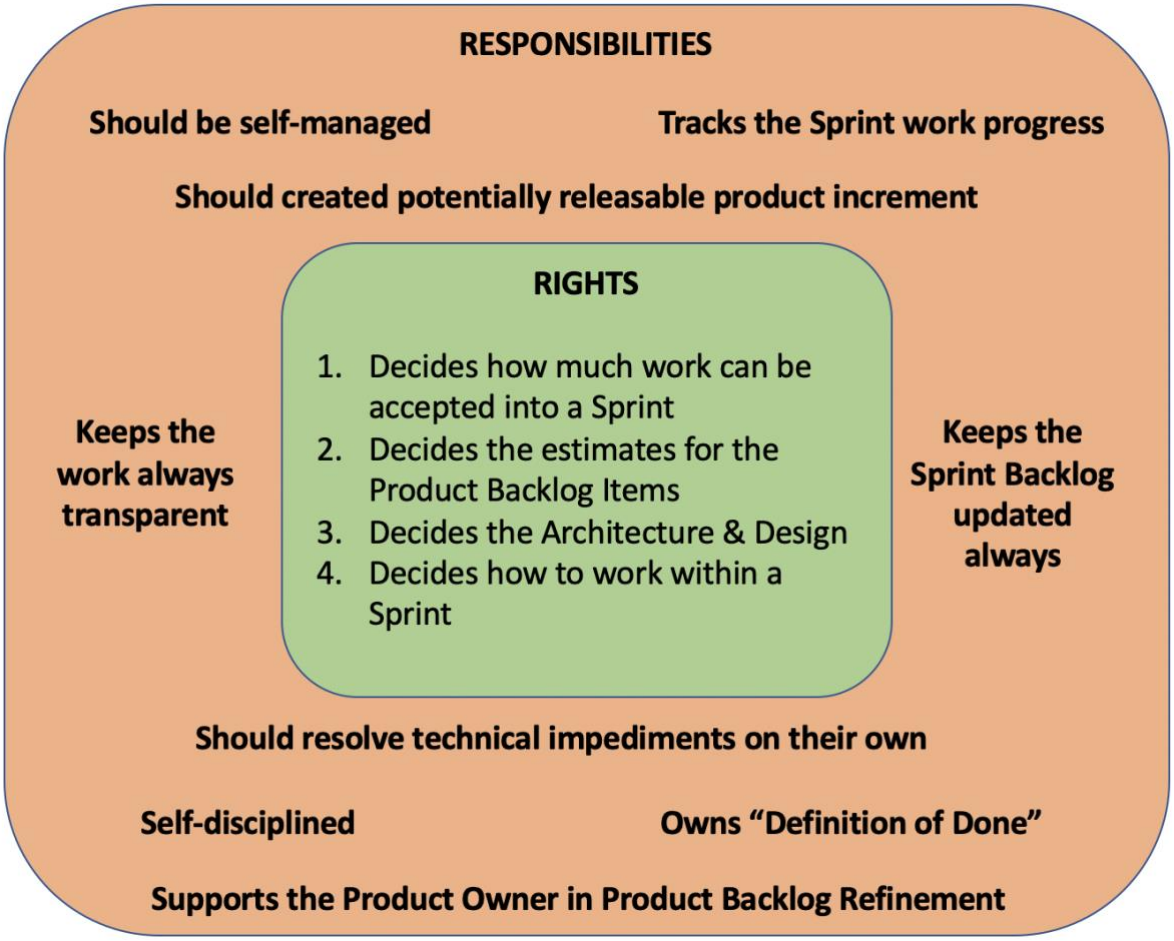

***Instead of having only SPECIALISTS in the team,***

***our goal is to create GENERALISED SPECIALISTS***

## How communication becomes complex with the team size?

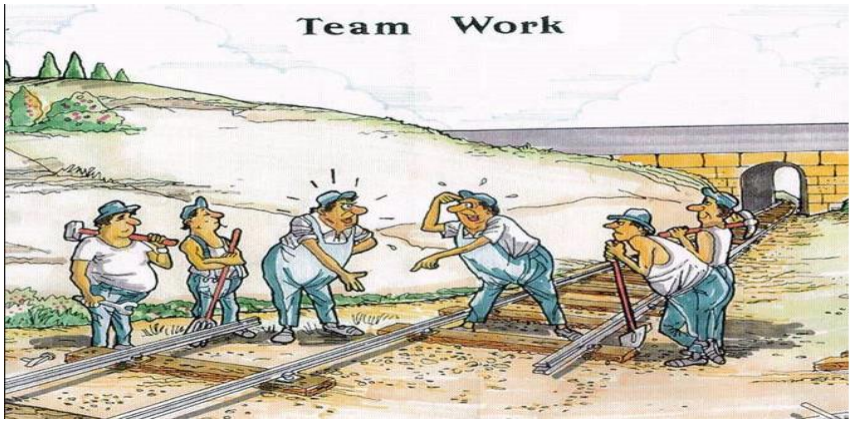


### Development Team Rights & Responsibilities:



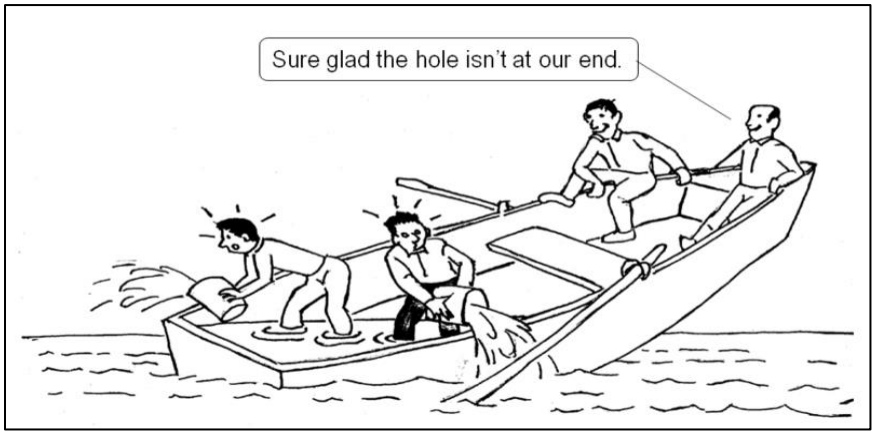
## Behavioral Patterns of Development Team

**Is this Teamwork?**



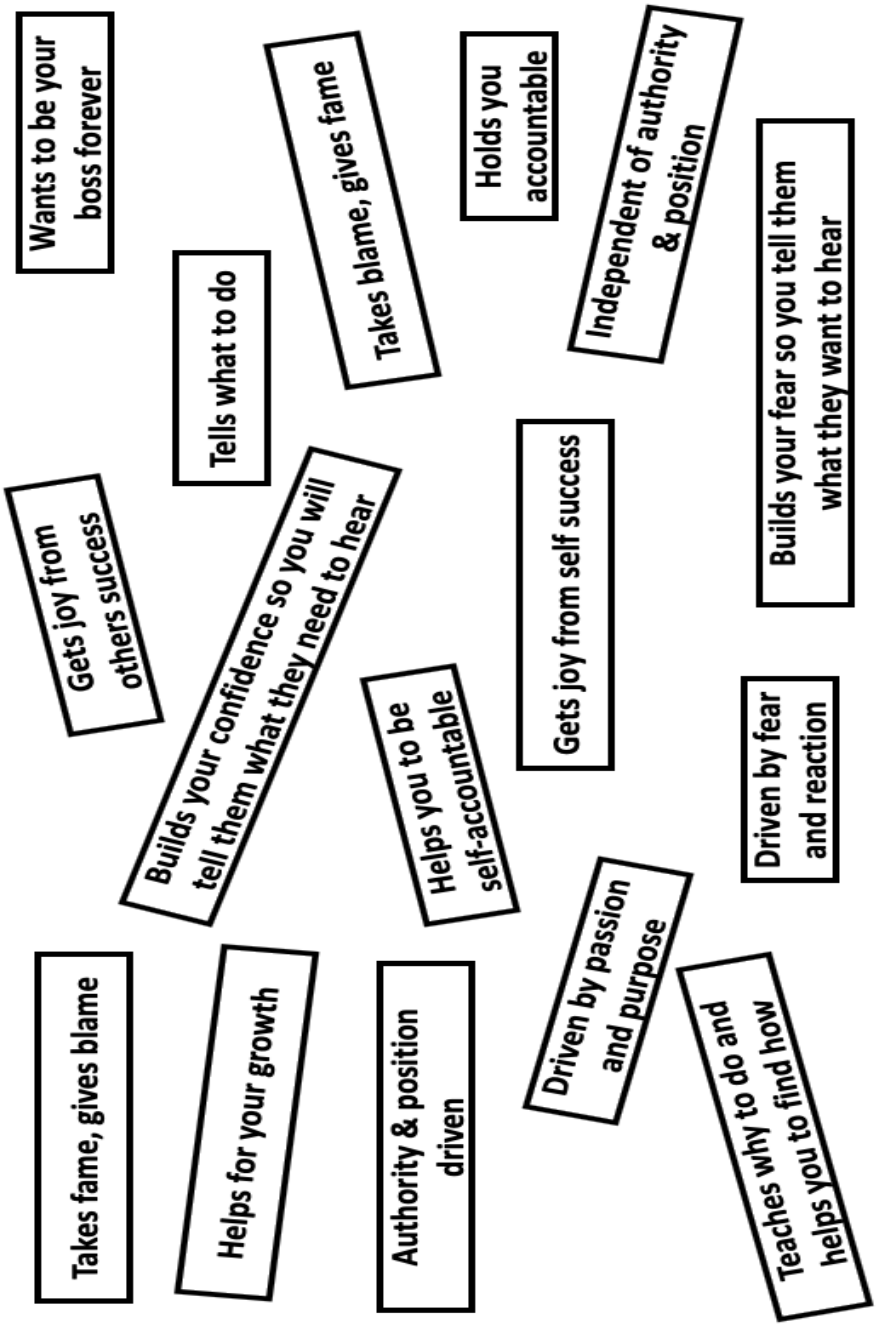

**Whose fault?**

**Collective responsibility?**

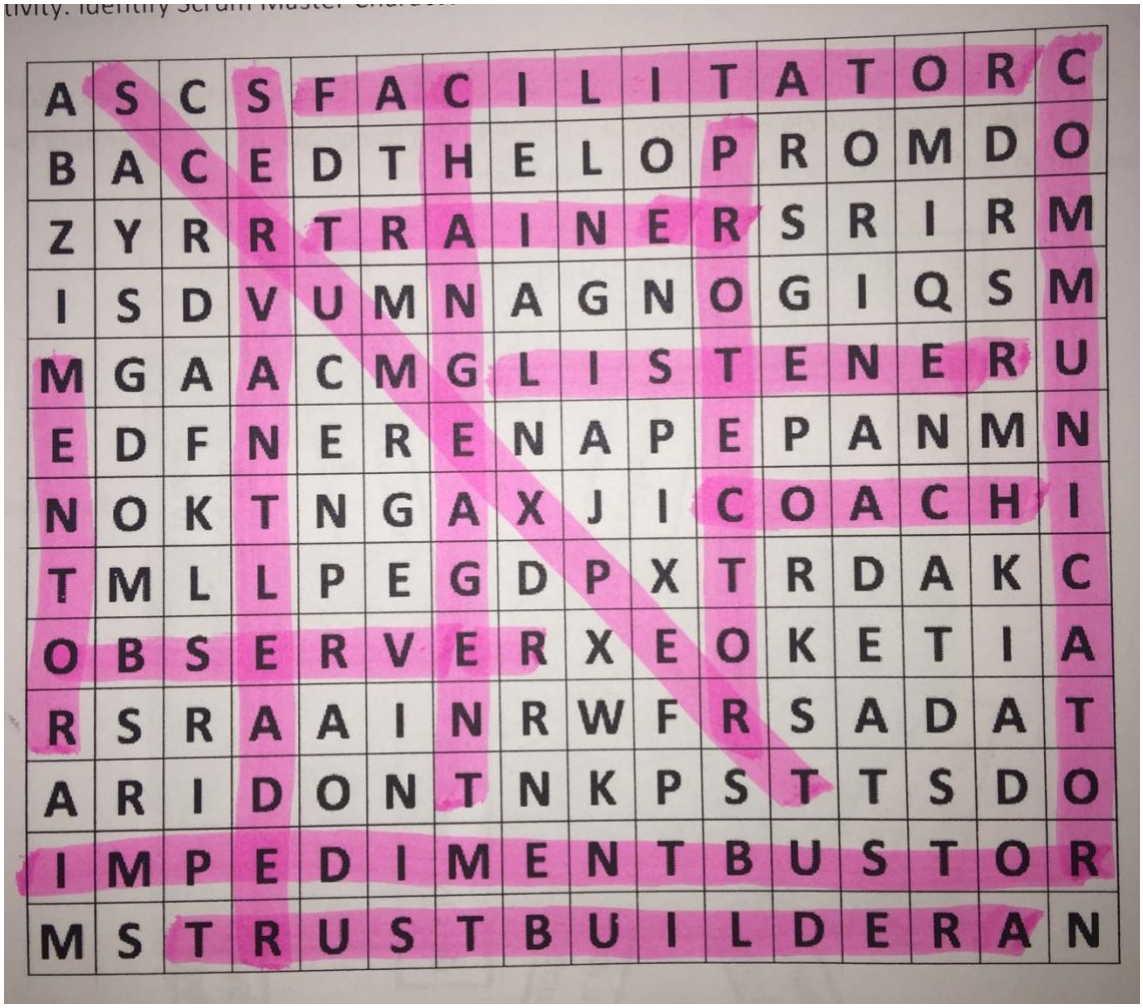


## SCRUM MASTER

**Activity: Pair up with some other participant. One should select the characteristics of a BOSS, the other has to select the characteristics of a LEADER from the below list. Then compare your selection to find if there are any overlapping selection. (Answer: Boxes with tilted angle are Leader characteristics, remaining are Boss characteristics)**



Activity: Identify Scrum Master Characteristics and responsibilities from the below maze.




## Scrum Master – As Servant Leader

Servant Leadership is a philosophy and a set of practices that enrich the lives of individuals, to build better organizations and ultimately creating a caring world. The "Servant Leader" is a Servant first. It begins with a natural feeling that one wants to serve first. Servant leaders have certain characteristics that make them different from a command and control based authoritarian leaders, as follows:

- ✓ Serving others, not yourself
- ✓ Putting others needs first before his/her own needs
- ✓ Not leading by title, but leading by trust
- ✓ Help people develop as high as possible
- ✓ Harnessing on collective power of the team
- ✓ Focus on building trust
- ✓ Truly empowers others
- ✓ Improves transparency
- ✓ Enhance collaboration
- ✓ Great active listening
- ✓ Ethical and caring
- ✓ Humble

**Book Reference:** Read the book "The Servant as Leader" by Robert Greenleaf.

## Scrum Master as a Coach

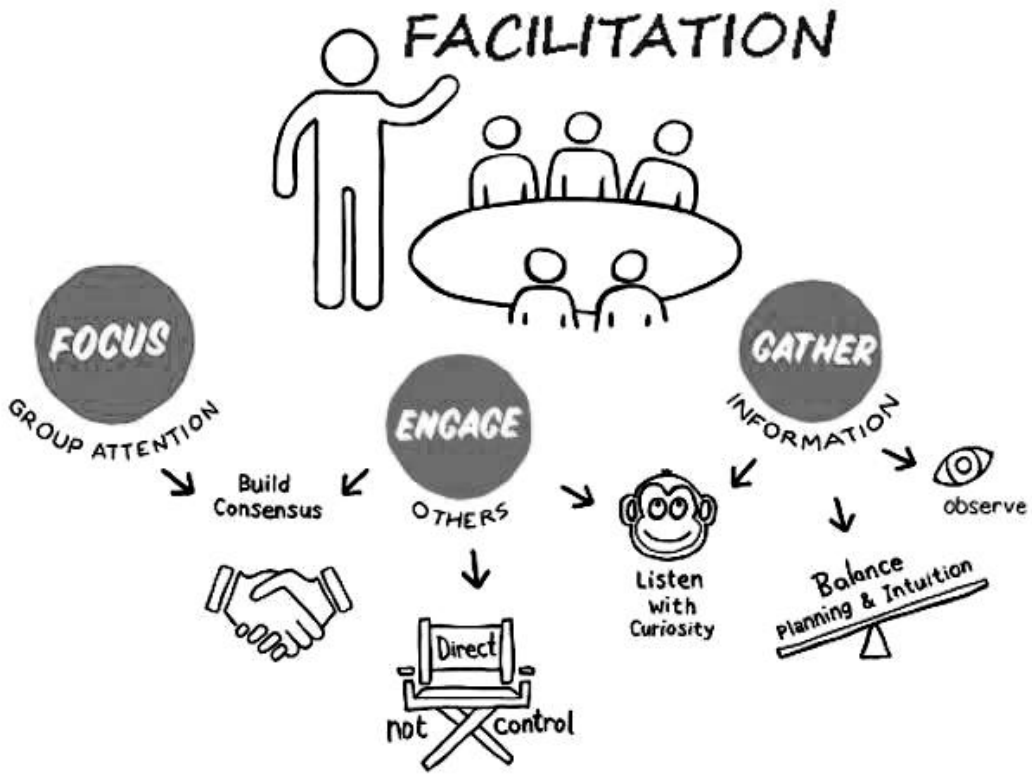
Coaching is unlocking a person's potential to maximize their own performance. It is helping them to learn on their own rather than teaching them. This means, coaching is all about "Asking" instead of "Telling". While asking, powerful questions will be used.

Coaching is a useful way of developing people's skills and abilities, and of boosting performance. It can also help deal with issues and challenges before they become major problems.

A coaching session will typically take place as a conversation between the coach and the coachee (person being coached), and it focuses on helping the coachee discover answers for themselves.

> *Coaching helps people think better.*
> *The person being coached is the expert in their work—*
> *the coach uses a questioning approach to*
> *help the person find their own answers.*
>
> —David Rock
> (www.workplacecoaching.com, June 2007)

## Scrum Master as Facilitator



Facilitation is the process of designing and conducting and concluding successful meeting or event or a decision making that has an objective. Facilitation serves the needs of any group, who are meeting with a common purpose. The person who facilitates the meeting is called a "Facilitator".

**Good facilitation techniques should:**

- Help the participants to be comfortable with each other
- Create fun and interesting learning and collaborative environment
- Boost the energy levels throughout the duration
- Use participatory and interactive techniques

**Facilitator Characteristics:**

- ✓ Neutral
- ✓ Does not stand in the front
- ✓ Active and unbiased
- ✓ Does not own the content and outcome
- ✓ Manage dysfunctions (withdrawn, domination etc.)

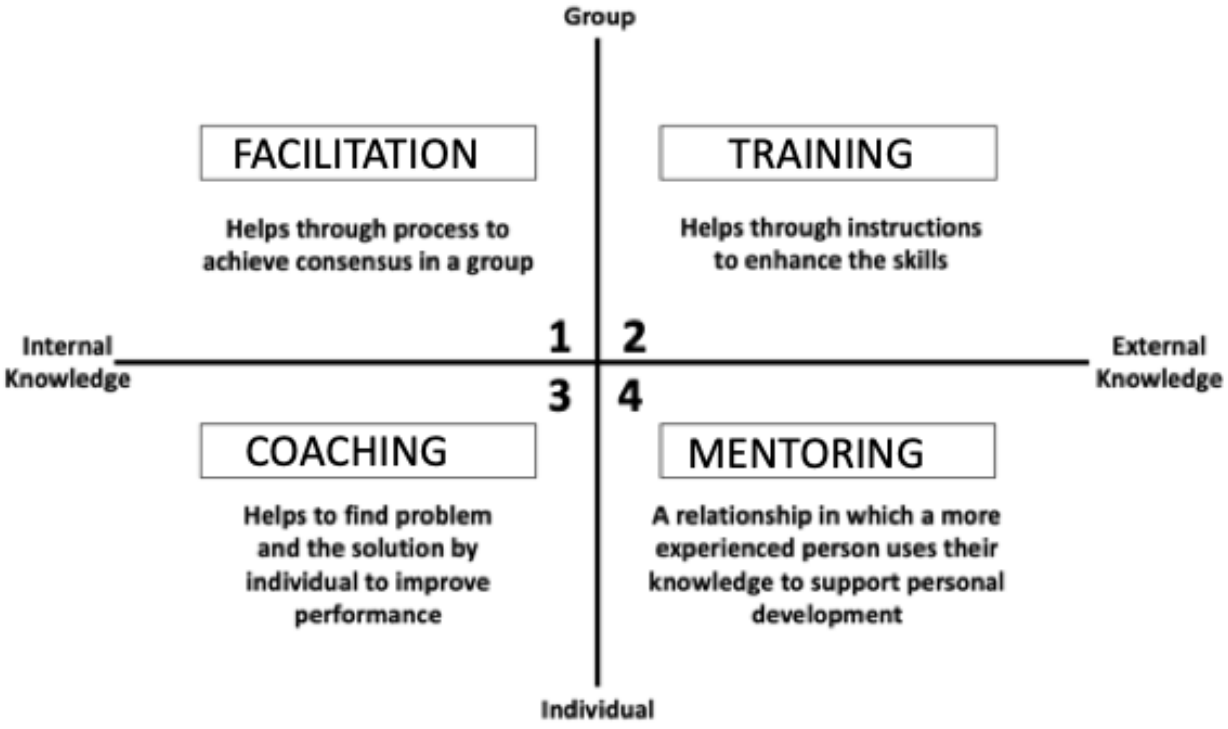
## Activity:

Explore the following Facilitation techniques and give one example where you will use them.

| Facilitation Technique name | Description | Where you can use? |
|---|---|---|
| **DOT VOTING** | Used for quick prioritization or any similar decision making. Everyone will get a fixed number of votes and they can use them against the items which they want to prioritize. After everyone uses their votes, whichever items get more votes that will be considered as important. | ➢ Product Backlog Prioritization<br>➢ To prioritize the improvements identified at retrospective |

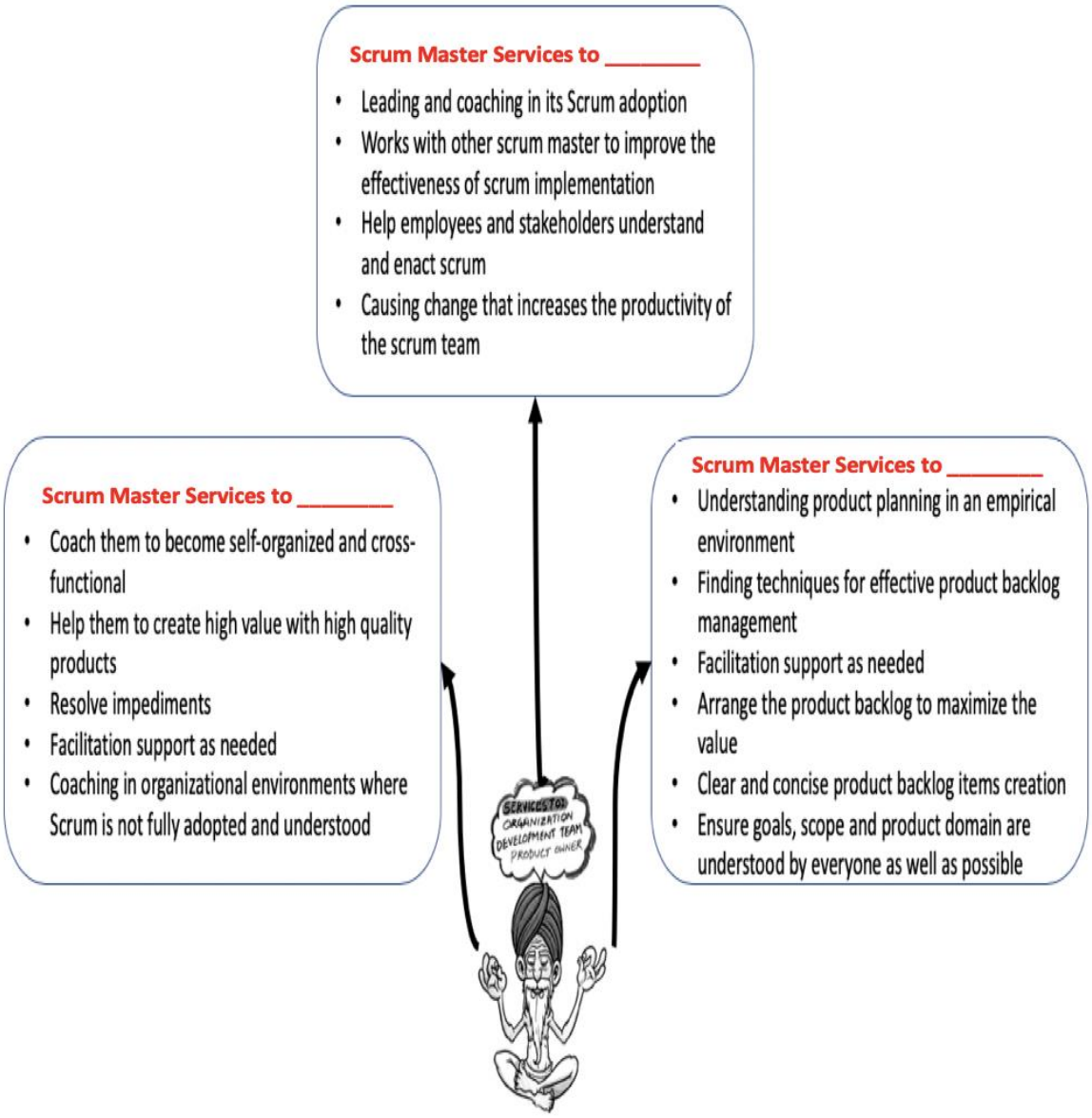
**Activity: Understand the differences**

**Based on the explanation given in each quadrant, map the names given below to each quadrant.**



## Scrum Master Services

Scrum Master serves Organization, Product Owner and Development team to maximize the interactions and bring required mindset change and transform the organization to be more agile.




**ANSWER:**

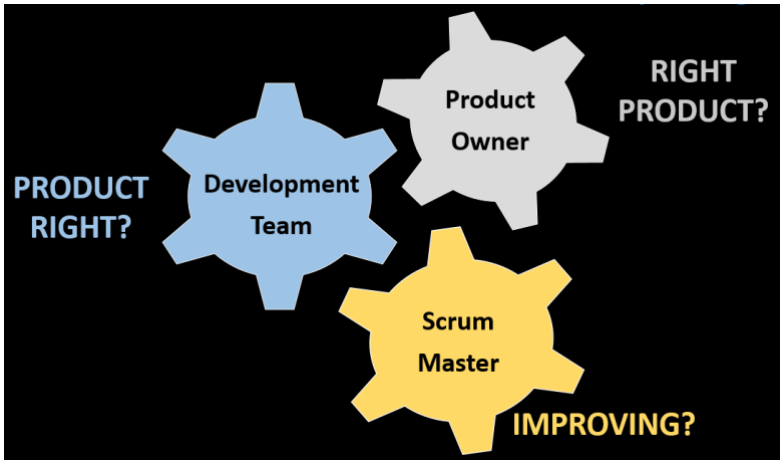
**Left box: Services to Dev Team**

**Top box: Services to Organization**

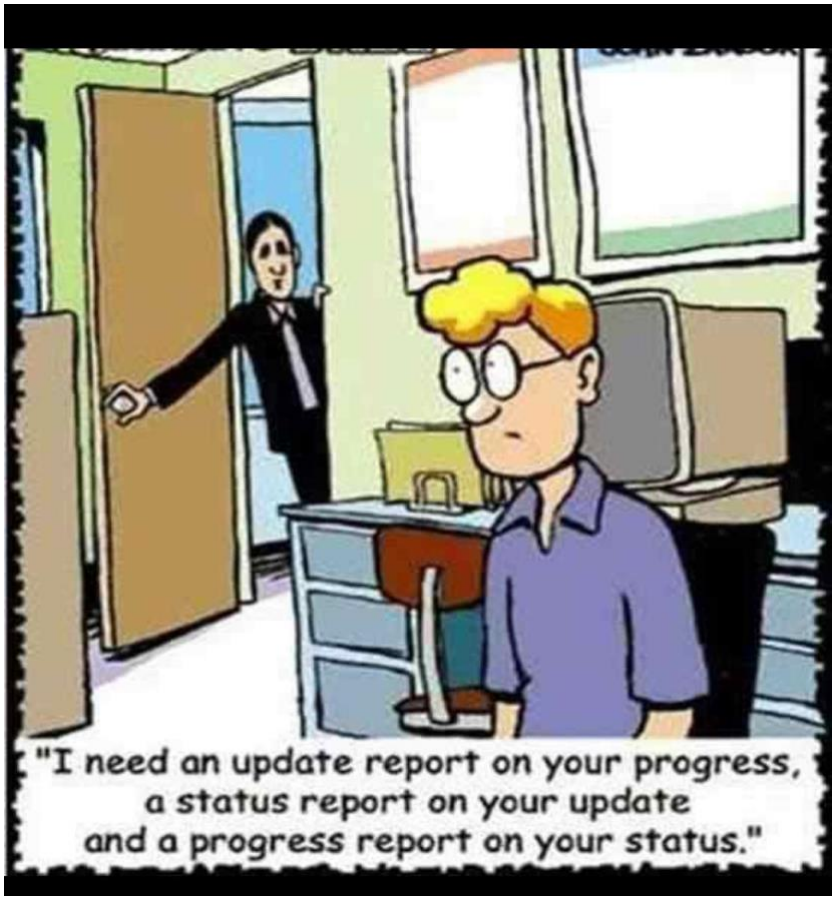
**Right box: Services to Product owner**

## How the 3 roles are related?

There is no reporting relationship among these 3 roles, but they work very collaboratively and work as a team towards meeting the product vision. Every role has certain responsibility towards meeting the product vision as shown below.




**Why there is no Project Manager in Scrum?**

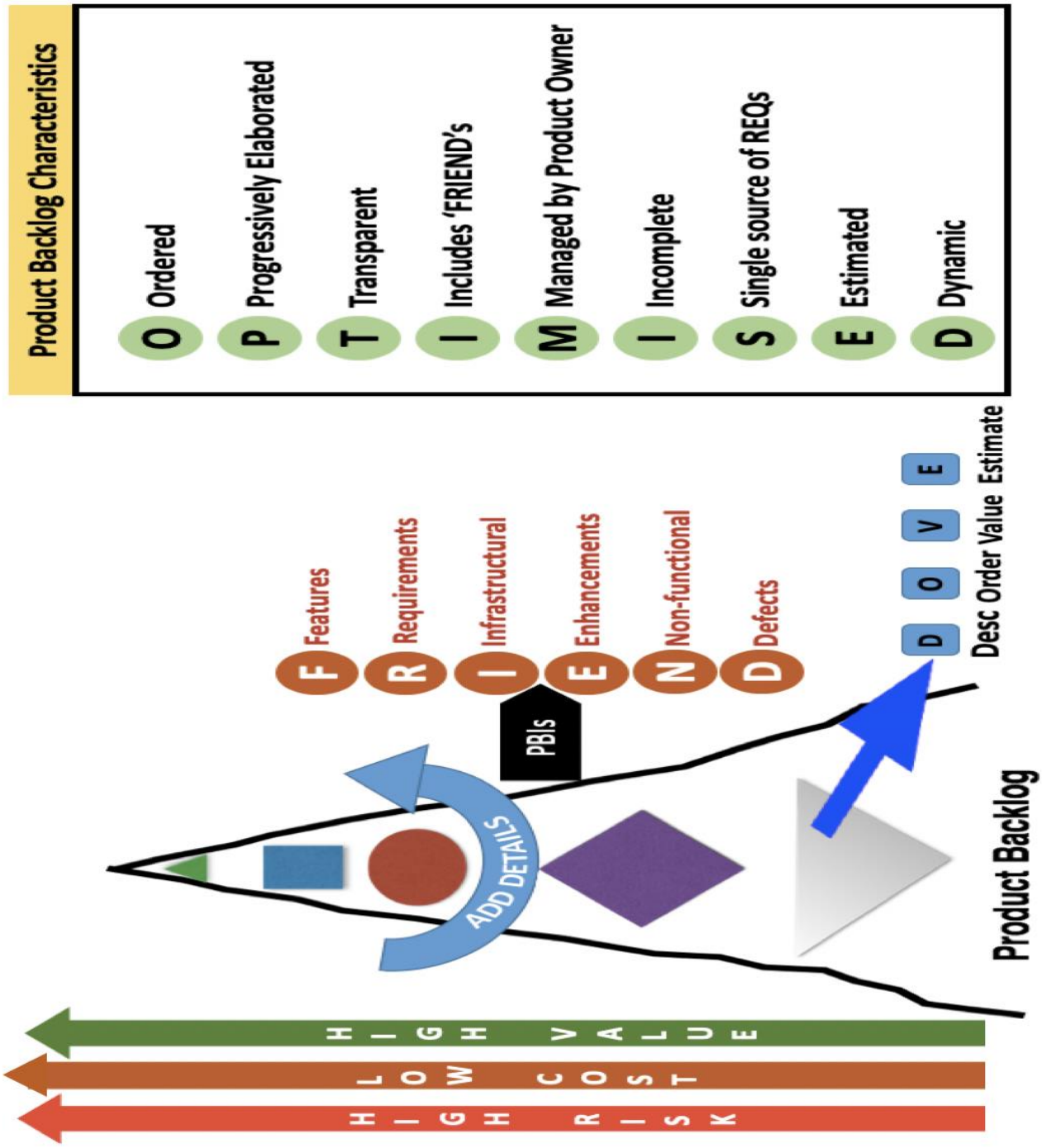


# Scrum Artifacts

## *Product Backlog, Sprint Backlog & Increment*

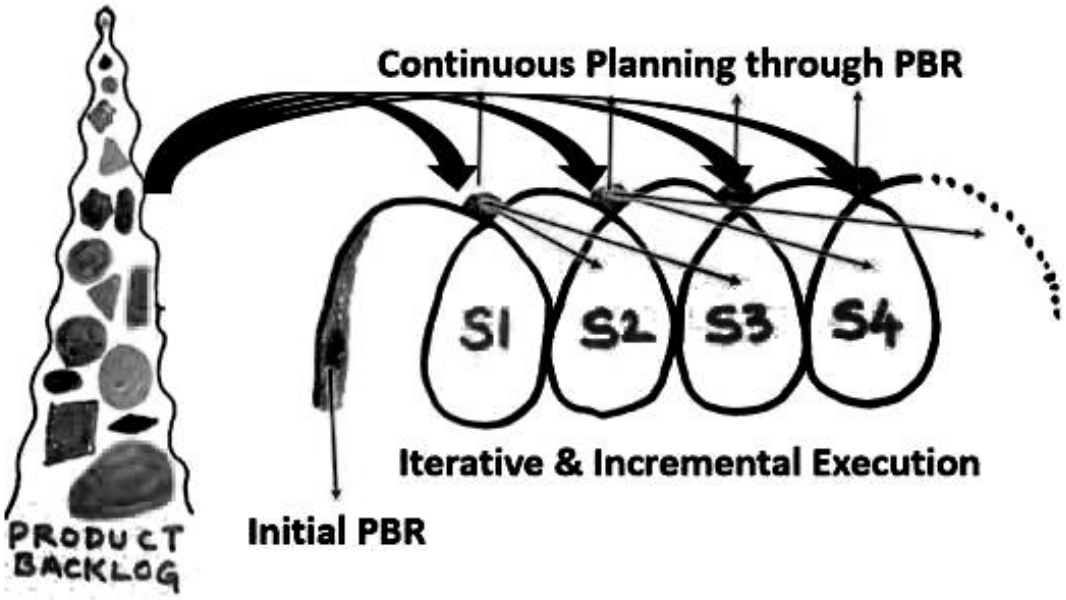
## Scrum Artifacts:

## Product Backlog:




**Answer: All boxes with WHITE COLOR TEXT are True.**

## Product Backlog Refinement:




• Product Backlog Refinement is an act of adding details, estimate & order to the items in the Product Backlog

• Product owner and Development team collaborates for Product Backlog Refinement

• Product Backlog Refinement is NOT an Event it is an ongoing Activity

• How, when, duration will be decided by Scrum team

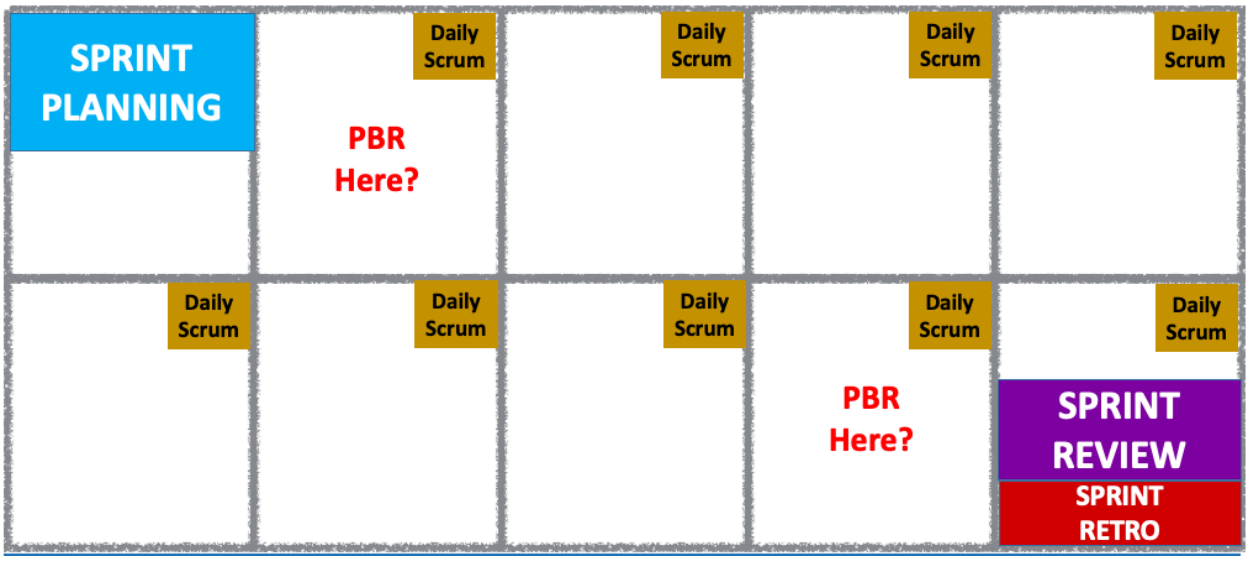
**What happens in the Product Backlog Refinement?**

- Discuss acceptance criteria
- Clarifications on Development Team queries
- Identify dependencies
- User interface
- High level design (if Development team needs)
- Estimation (Done by Development team only)
- Splitting large PBIs into smaller
- Removing non-value add PBIs
- Reordering
- ….

The Scrum team decides ***how, how long*** and ***when*** the PBR happens. However, usually not more than 10% of Development team's capacity of Sprint should be spent for refinement. Product Backlog items can be updated at any time by the Product owner or Product Owner's discretion.

**What is the best day/time to have Product Backlog Refinement in a Sprint?**

Assume that you have a 2 weeks Sprint with no holidays in between, so you will have 10 working days in the Sprint. Each box in the below picture represents a day of the Sprint.

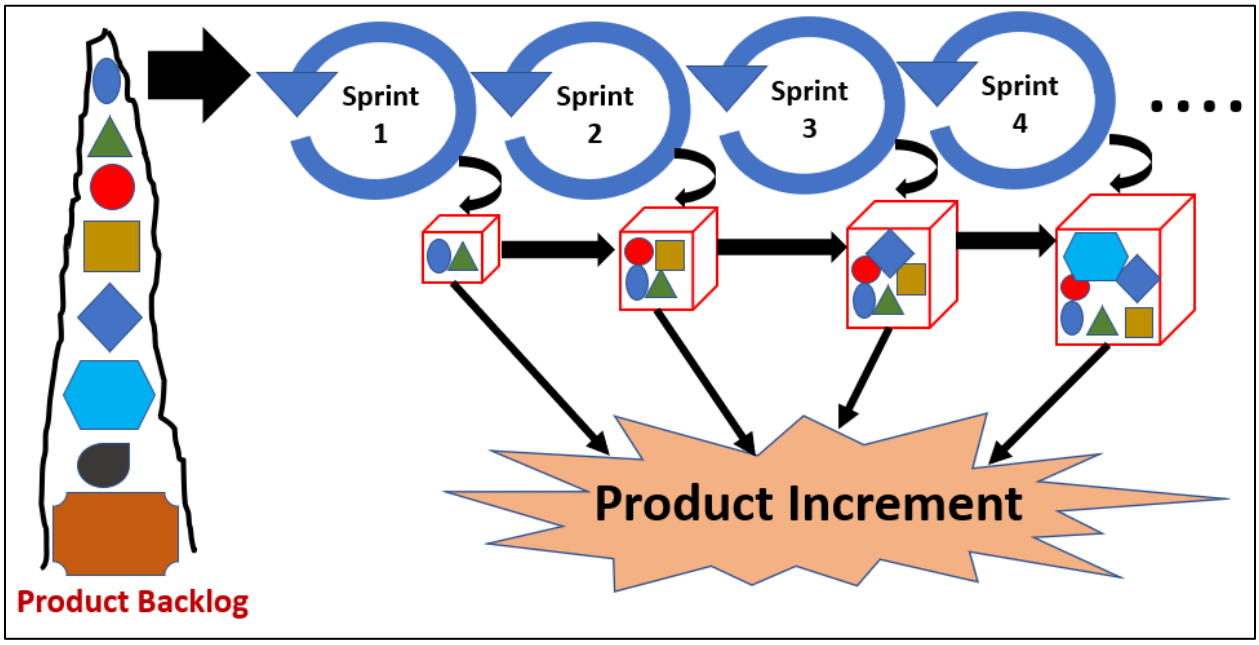



Product Backlog Refinement can be done during the Sprint any day after Sprit Planning and before Sprint Review. Generally it's not recommended towards the end or at the beginning because it will be either too early (if done towards the beginning of the Sprint, after Sprint planning) or too late (if done towards the end of the Sprint closer to the Review).

# Product Increment

*The VALUE created in the Sprint*

## Product Increment




- Created by Development Team
- Increment is the "**Sum of all PBIs completed during the Sprint and the value of the increments of all previous Sprints**"
- The new increment must meet "Definition of Done"
- The increment must be in a usable condition irrespective of Product Owner wants to release or not
- Is a step towards the product vision

**What are uses of incremental delivery?**

- ✓ Elicit feedback and validate assumptions
- ✓ Increases the Business value
- ✓ Reduces the risk by receiving early feedback
- ✓ Will create visibility to the stakeholders

**Factors influence the decision of releasing the increment:**

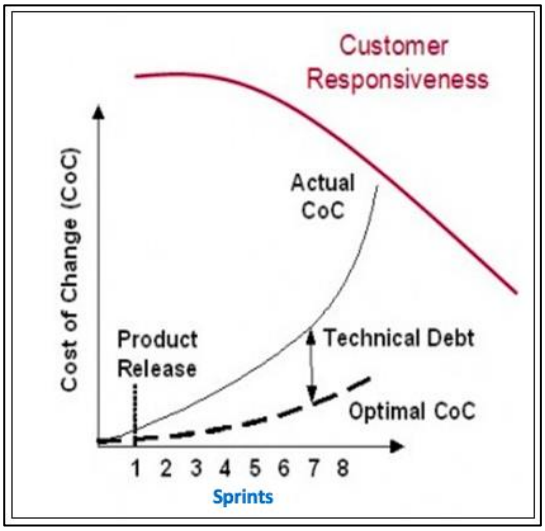
**VALUE and USABLE**

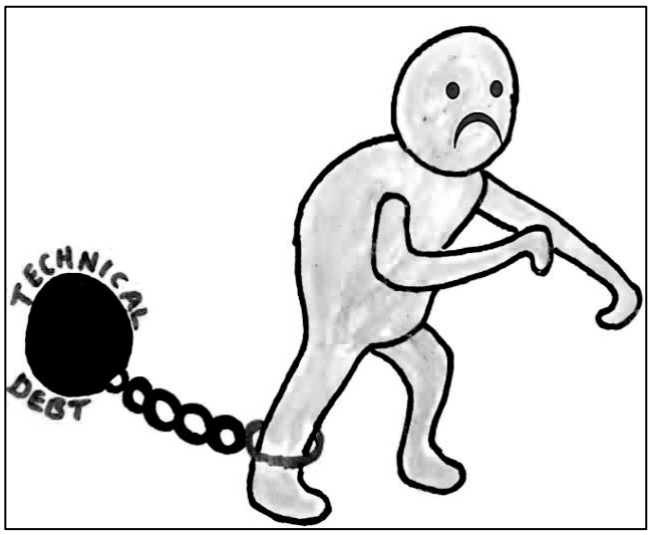
## Technical Debt

Technical debt (also known as design debt or code debt) is a metaphor referring to the eventual consequences of poor or evolving software architecture and software development within a codebase. As a change is started on a codebase, there is often the need to make other coordinated changes at the same time in other parts of the codebase or documentation. The other required, but uncompleted changes, are considered debt that must be paid at some point in the future.

Common causes for Technical Debt:

- ✓ Business pressures
- ✓ Lack of process or understanding
- ✓ Lack of Technical practices
- ✓ Lack of Automation
- ✓ Lack of collaboration
- ✓ Delayed integration
- ✓ Delayed refactoring
- ✓ Poor Definition of Done

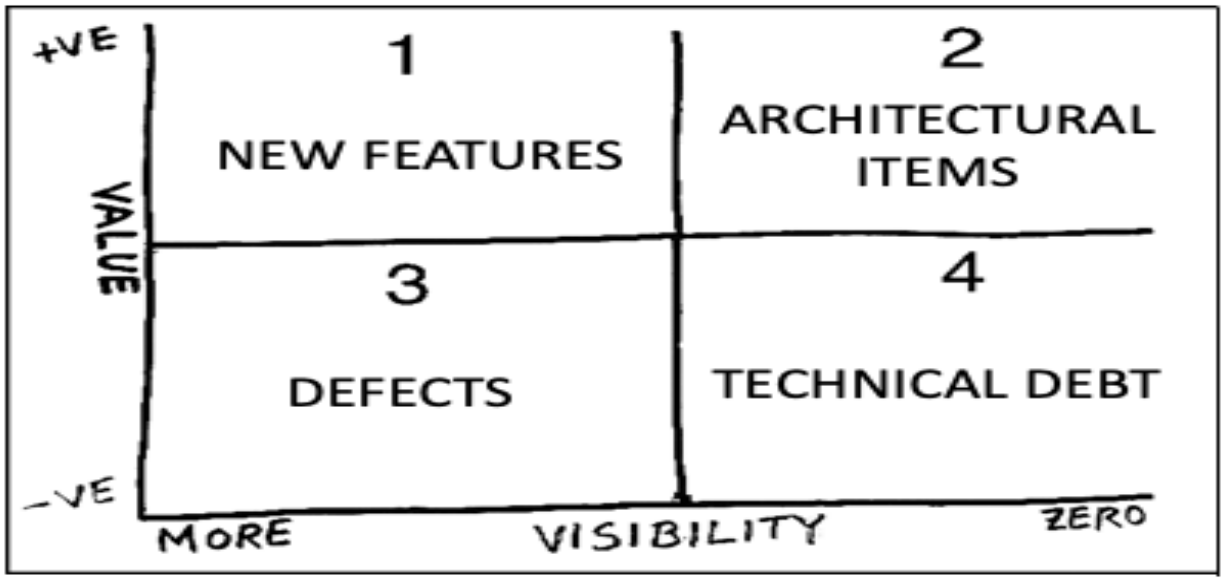



**Technical Debt is the difference between the**

**OPTIMAL cost and the ACTUAL cost of change**

**Activity: Map the 4 items on the right side into respective quadrant in the below picture.**

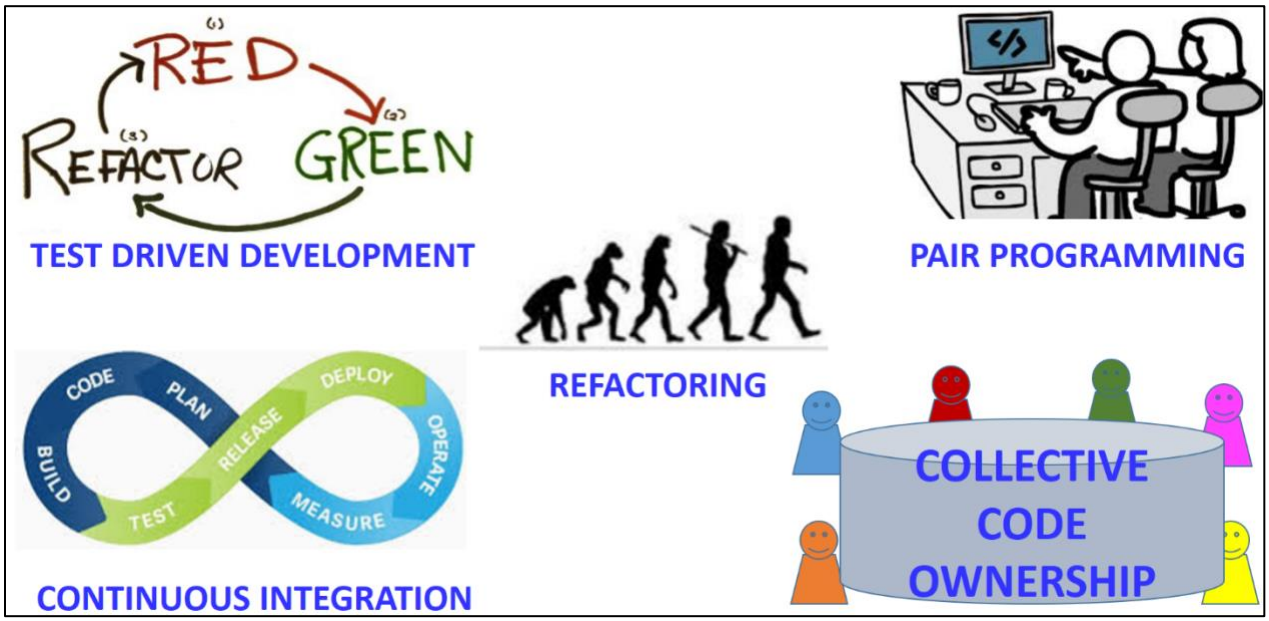



SPACE FOR NOTES

## Engineering Practices

Scrum Master has to help the Development team to follow few important engineering practices (Extreme Programming XP practices) in order for them to be agile with their code base and to effectively manage their development work. Scrum alone works but if Scrum teams follow engineering practices along with Scrum, it works even better and provide great results. Engineering practices help Scrum teams to manage their Technical Debt and to create high quality product increments.

Scrum Master need not to be technical to create the engineering culture at the team level and organization level. However, she/he can take appropriate approaches (example: Let the teams internally learn and implement, talk to the organizational leadership to arrange training).




Refer to my article on the Engineering Practices:

***Add toppings (Engineering Practices) to your ice cream (Scrum) to make it even tastier***

***Watch the below videos at your leisure to understand SPOTIFY engineering culture:***

**Part 1:** https://www.youtube.com/watch?v=Yvfz4HGtoPc

**Part 2:** https://www.youtube.com/watch?v=vOt4BbWLWQw&vl=en

# Agile Estimation

*A different way than traditional …*

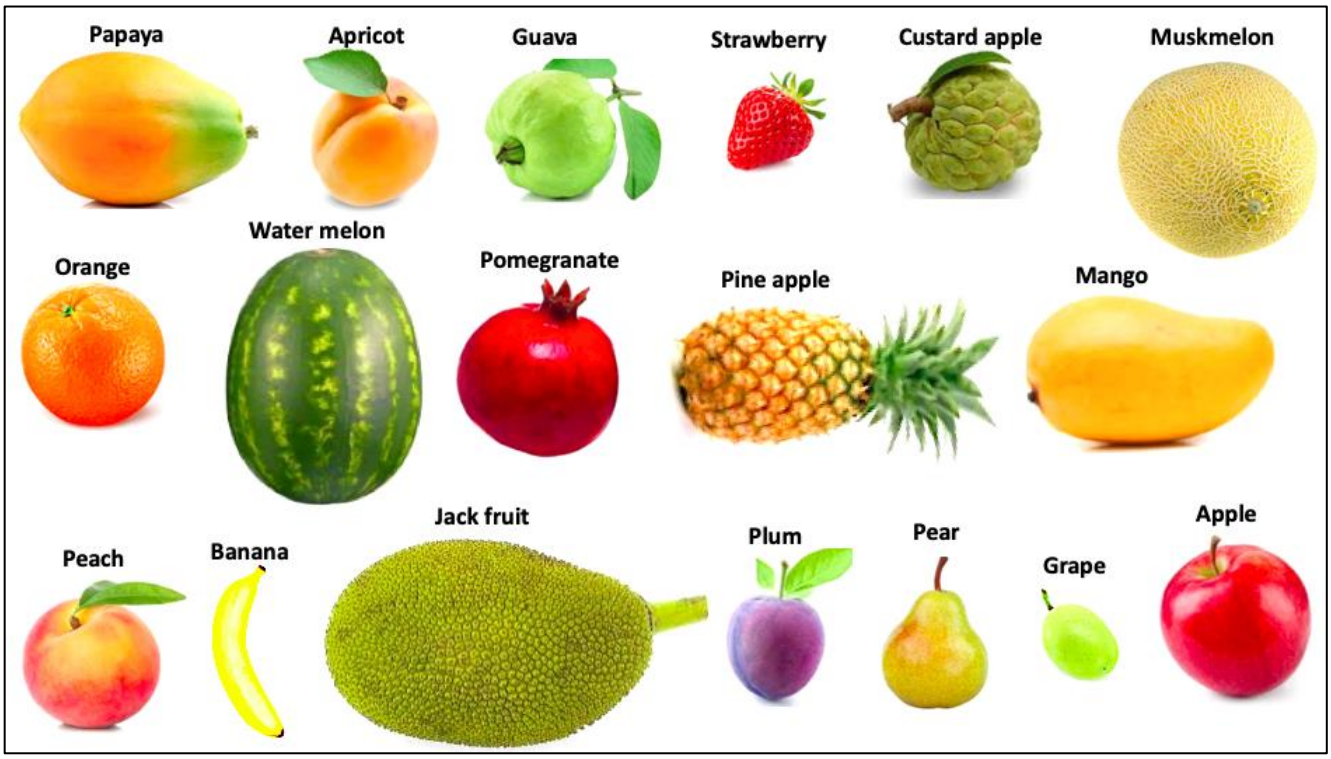
**(Not part of Scrum)**

## What is Estimation & Why Estimation?

Estimation is a ***Prediction***. Agile Estimation helps you to predict the Time (Schedule) through Release Planning.

**Important note:** Scrum does not insist or prescribe one method of estimation. It only recommends that the product backlog items should be estimated in some way and the Scrum team can decide what method they want to use.

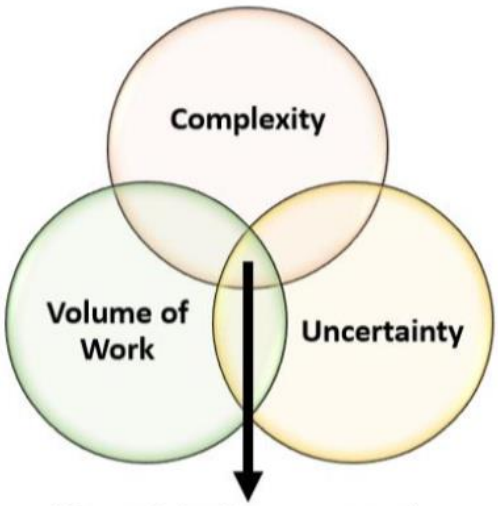
**Relative Estimation:**




Agile way of estimation believes in "***it is okay to be roughly ACCURATE than precisely WRONG***"

SPACE FOR NOTES

## Story point estimation





Story Points work based on the natural pattern Fibonacci Series.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ..... (Previous number + Current number = Next number)

Agile story points estimation pattern:
0, 1/2, 1, 2, 3, 5, 8, 13, 20, 40, 80, 100, ∞

Definite Range | Indefinite Range

### How to Estimate Story Points?

- A "Story Point" is an arbitrary measure used by Scrum teams in estimating the user stories.
- A "Story Point" is the relative measure (in number) to assess the relative size of a User Story
- The teams will come up with the story point estimate by:
  - Discussing with Product Owner to get clarity on the User Story
  - By considering the 3 parameters (Complexity, Uncertainty, Effort)

And also by comparing the story with the other stories that are already estimated (by the same team in the same project)

SPACE FOR NOTES

**Why Story Points?**

- Story Points help drive cross functional behavior
- Story points are a pure measure of size
- Estimating in story points is typically faster
- My ideal days are NOT your ideal days
- Story points help to track the team productivity in terms of velocity
- Story point estimation increases the collaboration among team

**When you are estimating the Story Points, you do not generally consider:**

- ✓ Who will implement the story?
- ✓ How long it takes to implement the story

**Is there any relationship between story points and hours?**

Not really, if you have 2 user stories of size 5 story points each, it is not mandatory that both these stories should take same amount of hours. The effort hours may vary based on the tasks required to complete the story. However, the difference may not be drastically high or low.


SPACE FOR NOTES

## Planning Poker (Wideband Delphi based) Estimation

Planning Poker is a consensus-based approach to agile estimating. It is based upon the "Wideband Delphi" model where subject matter experts work on estimation anonymously, iteratively until they reach consensus.

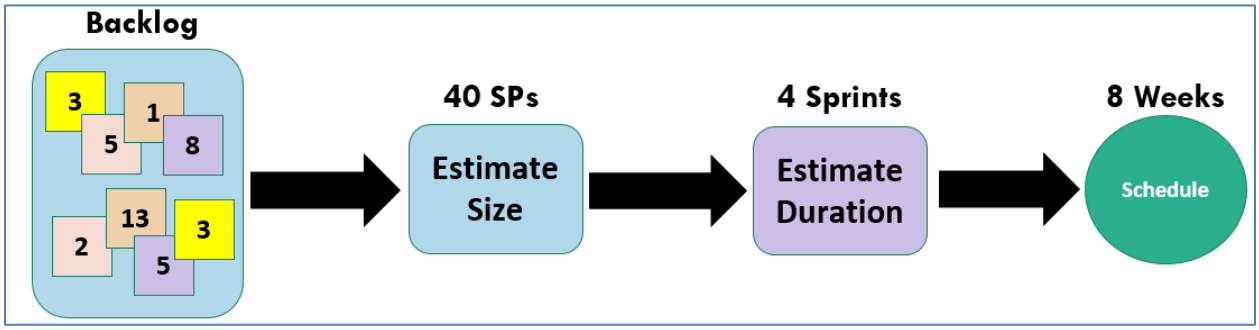
**When used?** *At the beginning of project AND/OR During iterations to estimate*

**Process:**

1. Estimators will be given Poker Cards with estimation numbers printed on them (0,1/2, 1, 2, 3, 5, 8, ?, ∞, coffee cup, etc.)
2. Product owner reads a story and it's discussed briefly among team and PO
3. Each estimator selects a card from his/her deck that's his or her size estimate for that story. Cards will be turned down initially.
4. Cards are turned over, so all can see them once everyone is done with their estimate
5. Discuss differences (especially outliners)
6. Re-estimate until estimates reach consensus

Note: Distributed Agile teams can play the Planning Poker online for the estimation

### How to derive the schedule?




SPACE FOR NOTES

# Definition of Done

*That creates Shared Understanding*

## Definition of Done (DoD)

Activity: Read the Definition done at: https://www.learnovative.com/definition-of-done/ and fill the below paragraph with suitable correct words.

Definition of done is a quality criteria checklist that helps to create potentially releasable product increment. Definition of done is created by Scrum team before Sprint planning. It helps to create shared understanding between Product owner and the Development team on what "Done" means. It also helps the Development team during Sprint planning to decide how much work can be pulled into the Sprint. Definition of done is applicable at the Product backlog level and it is not defined at the individual Product backlog Item level.

If there are multiple teams working on Same product backlog, then all the teams must have a mutually Agreed common definition of Done. Formal opportunity to inspect and adapt (strengthen) the definition of done is the Sprint Retrospective.

A product backlog item that mostly meets the definition of done should NOT be accepted by the Product owner. Accepting work that does not meet the Definition of done creates an illusion of Progress.

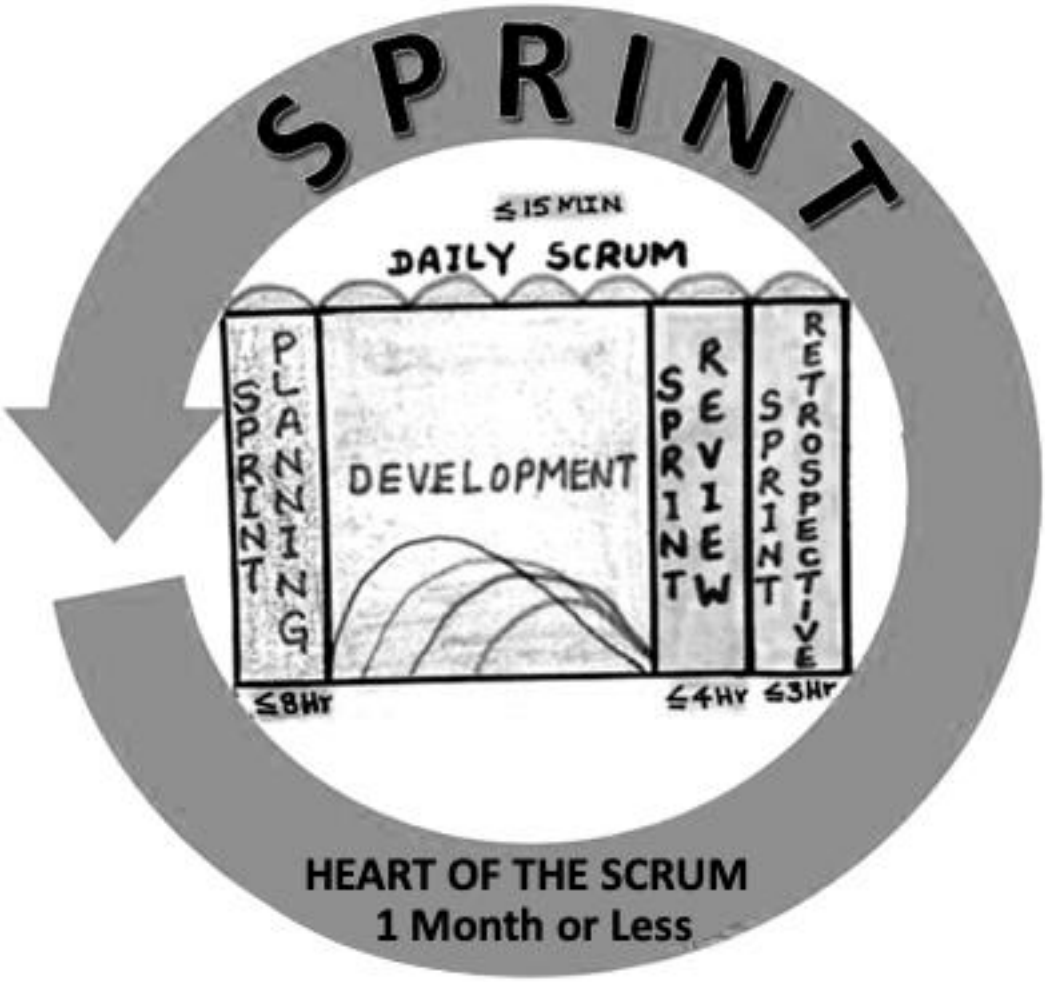
**Definition of Done vs Acceptance Criteria:**

| ACCEPTANCE CRITERIA | DEFINITION OF DONE |
|---|---|
| Related to functionality | Related to quality |
| Owned by Product Owner | Owned by Development Team |
| Defined at any time before accepting into Sprint | Usually defined before Sprint begins |
| Different for every Product Backlog item | Applicable to entire Product Backlog or increment |

# Scrum Events

*Formal opportunities to Inspect & Adapt*

## Scrum Events:

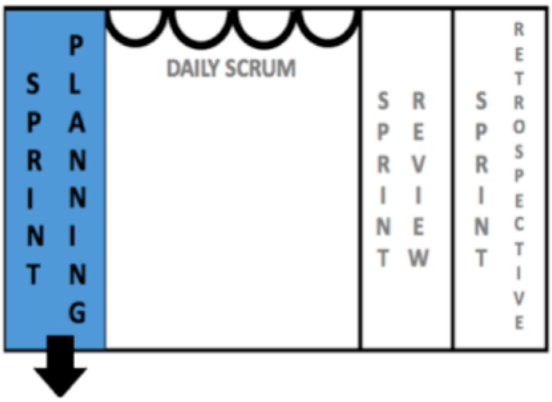



- Events in Scrum are used to create ***REGULARITY*** and minimize the need for ***other meetings***
- Sprint is a ***CONTAINER*** that contains other ***FOUR EVENTS***
- Each event in Scrum is a formal opportunity to ***INSPECT*** and ***ADAPT*** something
- These events are designed to enable ***TRANSPARENCY*** and **inspection**
- Failure to include any of these events results in reduced ***TRANSPARENCY*** and is a ***lost opportunity*** to ***inspect and adapt***

1. **Sprint Planning** to inspect the **Product Backlog** and **Plan the work** for the Sprint
2. **Daily Scrum** to inspect the **Sprint goal** and adapt **the Sprint Backlog**
3. **Sprint Review** to inspect the **Product Increment** and adapt the **Product Backlog**
4. **Sprint Retrospective** to inspect the **Scrum team itself** and **create an improvement plan**

**ALL THE SCRUM EVENTS ARE TIMEBOXED**

## Sprint Planning:




# SPRINT PLANNING

| | | |
|---|---|---|
| **Purpose:**<br>Scrum team collaborates to understand the work of the Sprint in Sprint planning. | **Participants:**<br>Scrum Team and any others like SMEs<br>**Mandatory:** Scrum Team<br>**Optional:** SMEs (Architects, DBAs, etc.,) | **Timebox:**<br>Max 8 hours for 1 month Sprint, shorter for shorter Sprints<br>**When it occurs?** This is the first event in the Sprint, so happens on first day<br>**Frequency:** Every Sprint based on duration |

| | |
|---|---|
| **Topic 1:**<br>What can be achieved in the Sprint. A forecast of the work by Development team | **Topic 2:**<br>How the selected work can be achieved during the Sprint. Decomposing the work and high level design discussion |
| **Inputs:**<br>Product Backlog, Team's past performance, Projected capacity, Latest product increment | **Outputs:**<br>Sprint Goal<br>Sprint Backlog |

**Sprint Planning Meeting – Topic 1: WHAT can be achieved in this Sprint?**

| ROLE | Contribution in WHAT part of Sprint Planning |
|---|---|
|  **Development Team** | - Forecasts the functionality that will be built in the Sprint<br>- Only Development team decides how much work can be pulled into Sprint<br>- No one can force the Development team on the selection of amount of work to be pulled into the Sprint (# of Product Backlog Items) |
|  **Product Owner** | - Discusses the tentative objective for the Sprint (What he/she wants the Development team to achieve in this Sprint) |
|  **Scrum Team** | - The entire Scrum team collaborates to understand the work of the Sprint<br>- Crafts the Sprint Goal (one of the two outputs of the Sprint Planning)<br>- Negotiations can take place between Product Owner and Development team |

***Sprint Planning Meeting – Topic 2: HOW the chosen work will get done?***

| ROLE | Contribution in HOW part of Sprint Planning |
|---|---|
|  **Development Team** | - Based on the Sprint goal and selected product backlog items for the Sprint, the Development team decides how they will build the functionality in to a "Done" product increment during the Sprint.<br>- The selected product backlog items and the plan for delivering them (usually decomposed into technical tasks) is called the Sprint backlog. Sprint backlog is the second output of Sprint planning.<br>- The Development team usually starts the design the system for the increment that they create in the Sprint.<br>- Work may be in different sizes, or estimated effort. However, enough work is planned by the Development team to forecast what they can build in the Sprint.<br>- By the end of the Sprint planning meeting, they decompose the work for first few days. As and when they go into the Sprint, they may decompose the remaining work.<br>- If the Development team feels it has too much or too less work, they may renegotiate the product backlog items with the Product owner. |

| | |
|---|---|
| | - The Development team may also invite subject matter experts/Technical architects/Domain experts to the Sprint planning if they need help.<br>- By the end of the Sprint planning, the Development team should be able to explain to the Product owner and the Scrum Master how they will achieve the Sprint goal as a team together and create the product increment. |
|  <br>**Product Owner** | - Product owner can clarify the selected product backlog items and make trade-offs |
|  <br>**Scrum Master** | - Ensures the event takes place<br>- Makes sure attendees understand the purpose of Sprint planning<br>- Teaches the Scrum team to keep it within the Timebox<br>**(Scrum Master participates in both WHAT and HOW parts of Sprint Planning)** |

## Sprint Goal:

- Collaboratively **created** by the Scrum team during **Sprint Planning meeting**
- It is an **Objective** set for the Sprint
- Provides **Guidance** to Development team on **why** they are building the increment
- Gives **flexibility** to Development team regarding the functionality
- The selected PBIs deliver **one coherent function** which is Sprint Goal
- It brings the entire Development team to **work together**
- During the development, Development team keeps Sprint Goal in mind
- Development team **implements functionality and technology** to satisfy the Sprint Goal

**Examples for GOOD Sprint goals:**

- ***Create basic shopping cart functionality with add, edit items and make payment***
- ***Allow users to make payments online***
- ***Enhance the patient workflow in the hospital management system***
- ***Improve the reports performance by 15%***

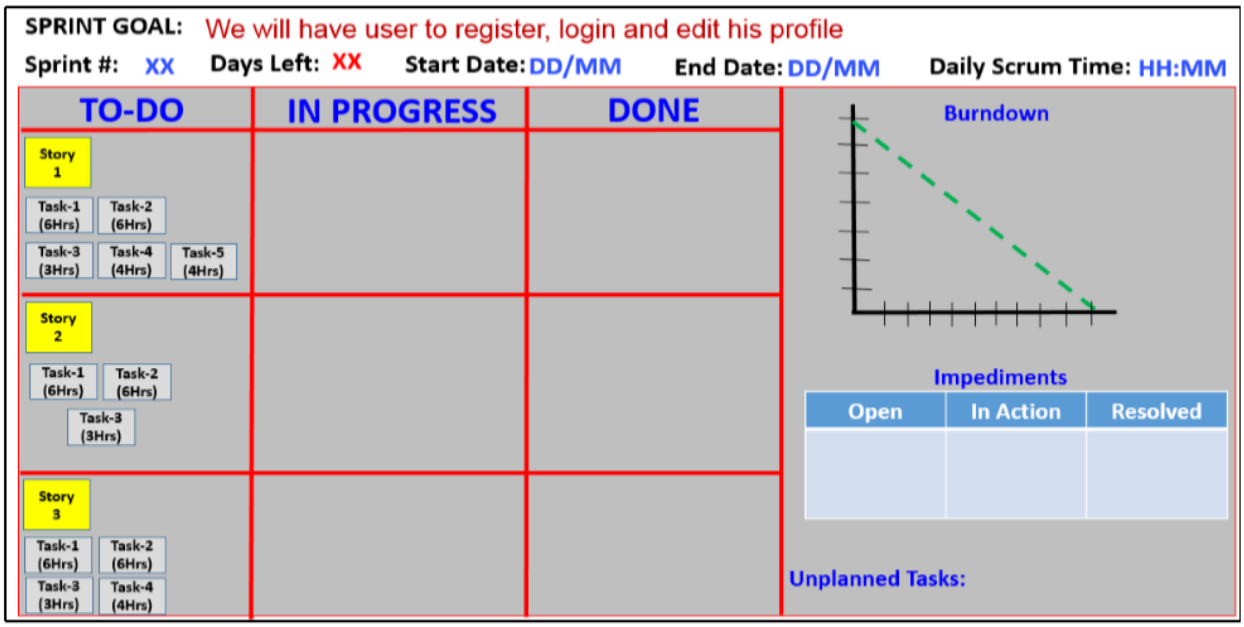
**Examples for NOT SO GOOD Sprint goals:**

- ***Complete 7 user stories and fix 12 bugs***
- ***Work for 350 hours effort in this Sprint***

## Sprint Backlog:

- Contains the Selected PBIs (WHAT) & a Plan (HOW) to deliver the increment
- Output of Sprint Planning meeting
- A Forecast by Development Team on what they can deliver during the Sprint
- Owned by Development Team and only they can change the Sprint backlog
- Helps Development team to Plan and organize their work
- Gets updated during the Sprint every day
- It emerges during the Sprint
- As new work is required, Development team adds it to the Sprint Backlog
- As work is completed, remaining work is updated
- Unnecessary elements are removed from Sprint Backlog
- Highly visible and frequently updated during the sprint
- Keeps the sprint work visible

**Sprint backlog view at the end of Sprint Planning (Sample view only):**



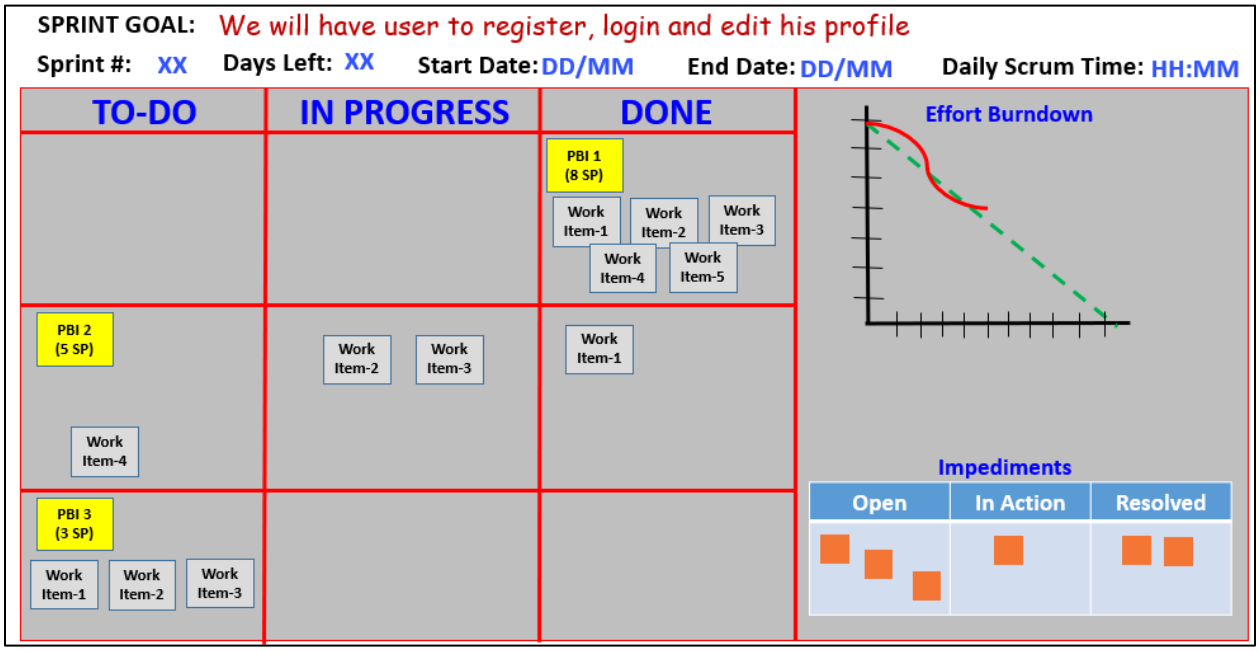

SPACE FOR NOTES

## Sprint Execution (Development):

- After the Sprint planning meeting the Development team is clear on "Why" they are building the current increment and what product backlog items they have to implement the Sprint goal.
- They also will have identified the initial tasks for each backlog item (Tasks are not part of Scrum)
- They keep their sprint backlog visible. No one assigns the work, they pull on their own.
- Development team works during the Sprint using their collective experience and knowledge taking the sprint backlog as their planning tool.
- The design will be done only for the items that are targeted for the sprint (JIT design).
- The sprint backlog gets continuously updated based on the progress of the work.
- Team closely interacts with the Product owner for any clarifications during the sprint.
- During sprint execution, team focuses on quality that meets the Definition of Done.
- Team members work collaboratively during the execution and help each other to complete the Sprint goal collectively (collective accountability to meet the Sprint goal).
- During the sprint the development team does whatever they have to do to meet the sprint goal.
- No one outside the Development team or the Product owner or the Scrum master should direct the Development team during the Sprint. They are self-organized, and they collectively plan and execute the work.
- The Development team tracks their work during the Sprint and they may use some artifacts such as Sprint burn-down/burn-up charts.

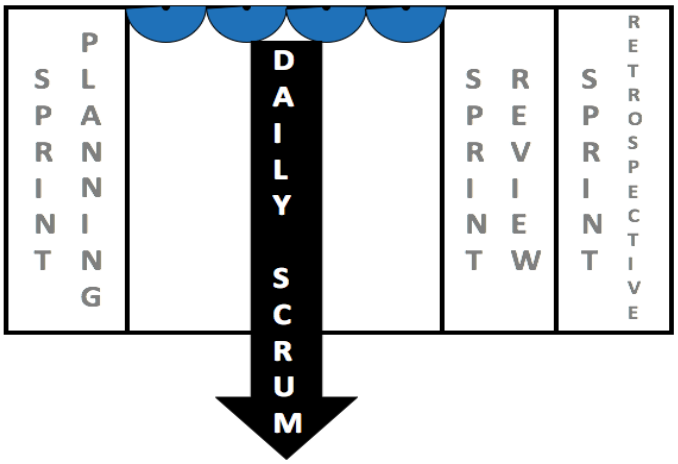

SPACE FOR NOTES

***The Sprint backlog looks as below during the Sprint (Sample only)***




SPACE FOR NOTES

## Daily Scrum




# DAILY SCRUM

| | | |
|---|---|---|
| **Purpose:**<br>Synchronization meeting for Development Team. It is a planning and knowledge sharing meeting. | **Participants:**<br>Development Team<br>**Mandatory:** Development Team<br>**Optional:**<br>Product owner, Scrum master, and any others as silent observers | **Timebox:**<br>Max 15 min for any sprint duration<br>**When it occurs?**<br>Same time, same place<br>**Frequency:**<br>Every day |

| | |
|---|---|
| **What is inspected?**<br>Sprint goal, based on the last 24 hours progress the team has made | **What is adapted?**<br>Sprint backlog, to plan the work for the current day |
| **Inputs:**<br>Work done information on last 24 hours, Sprint Goal, Sprint Backlog | **Outputs:**<br>Updated Sprint backlog with the current day's plan and a list of impediments (if any) |

- Daily Scrum is a Planning & Knowledge Sharing meeting, status is byproduct.
- Structure can be Questions/Discussion based.
- Product Owner and Scrum Master are optional for Daily Scrum.
- Scrum Master ensures it happens and if any others join, Scrum Master makes sure they are silent.
- Daily Scrum is NOT to respolve the impediments, only to highlight them.
- If required, a detailed discussion can happen to discuss impediments or any other things immediately after the Daily Scrum

**Advantages of Daily Scrum Meeting:**

| Advantage description |
| --- |
| **Improves the collaboration and communication within Development team** |
| **Sets the day's direction** |
| **Eliminate need for other meetings** |
| **To identify any impediments that slow down the Development team** |
| **Promotes quick decision making** |
| **Improves Development team's level of knowledge** |
| **Encourages self-organization** |
| **Enhances transparency** |
| **Optimizes the probability of meeting the Sprint goal** |

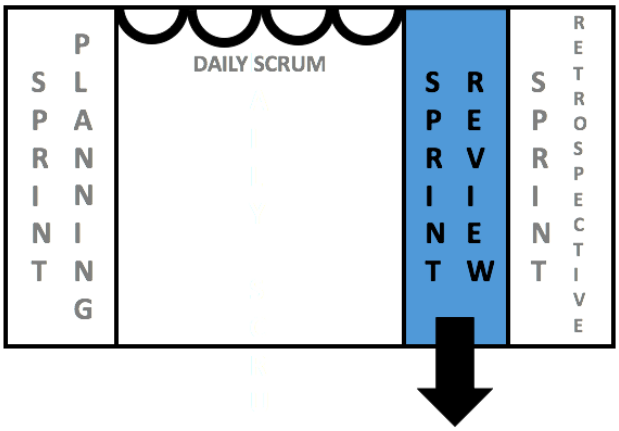
## Common dysfunctions of Daily Scrum

1. Driven by a Manager or Scrum Master
2. Development Team goes to Daily Scrum only when Scrum Master is asking them
3. Team focus only on answering the 3 questions mechanically and handover to the next person
4. Not giving enough importance to highlight the impediments in Daily Scrum
5. Development Team members not paying attention to what other persons are saying in Daily Scrum
6. Development Team members leave the Daily Scrum after their turn of update
7. All Development Team members not attending at the same time for Daily Scrum
8. Development Team members focusing only on their individual work and not considering the Sprint Goal as a common purpose
9. Not maintaining Same time and Same place for Daily Scrum
10. Skipping the Daily Scrum and/or having in some other means like Email/Chat

**Tips to address the above: These are some general recommendations and not prescribed by Scrum Guide.**

- ❑ Clear working agreements for Daily Scrum. Ask the team to come up with what best format will fit for them to give update which is effective, focus on planning and knowledge sharing.
- ❑ It is the Development team's meeting, so Scrum master should ensure the Development team understands the purpose and conduct the meeting even though the Scrum master is not present.
- ❑ Conduct the Daily Scrum at the Sprint backlog to see what is going on. It helps enhancing the transparency and to plan better.
- ❑ Team should identify a suitable time that is more convenient and effective, and everyone can participate.
- ❑ Introduce small fun punishments like if anyone skips the Daily Scrum on the days where they are in office, they should get coffee to the team or they should sing or dance.
- ❑ Inspect and adapt to address slipping attendance and to identify the common suitable time.
- ❑ It is always good the team members touch the task that they are working or completed while giving the update.
- ❑ Team members should be self-organized. They can stand in a circular mode so that everyone can see everyone else.
- ❑ Leaving their mobiles at their work place and come to Daily Scrum without mobiles
- ❑ Keeping a smiley ball and ask team to use that ball to rotate it during Daily Scrum. Whoever holds the ball can only speak and others should listen. A person once completes his update will throw the ball to some other person randomly

## Sprint Review




**SPRINT REVIEW**

| | | |
|---|---|---|
| **Purpose:**<br>To inspect the increment with the Stakeholders and the Scrum Team and stakeholders collaborate about what was done in the Sprint. They also discuss what are the next things that could be done to optimize value | **Participants:**<br>Scrum Team and Stakeholders and any others who are interested<br>**Mandatory:**<br>Scrum Team and Stakeholders<br>**Optional:** Any others who are interested | **Timebox:**<br>Max 4 hours for 1 month Sprint, shorter for shorter Sprints<br>**When it occurs?** Last day of the Sprint, before the Retrospective<br>**Frequency:** Every Sprint once |

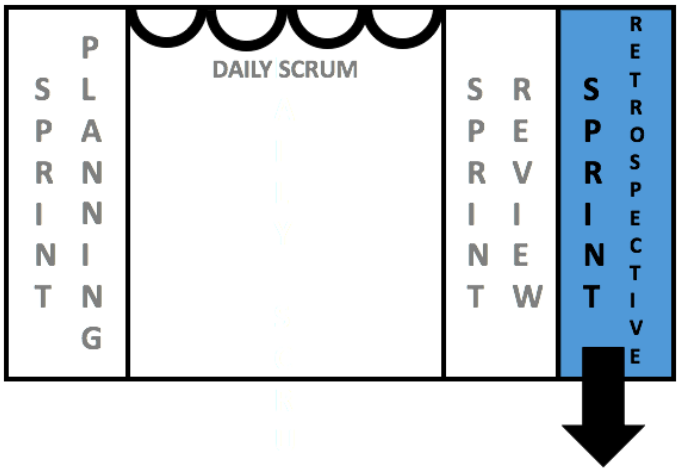
| | |
|---|---|
| **What is inspected?**<br>Product increment | **What is adapted?**<br>Product backlog |
| **Inputs:**<br>Product increment and information of what items are done and what are not done | **Outputs:**<br>Adapted Product backlog (if needed) based on the feedback received from stakeholders |

Main purpose of the Sprint review is to get ***feedback*** from the stakeholders, and not for formal ***approval*** or rejection of the items.

**What kind of changes (adaptation) may be done to the Product Backlog?**

*New items may be added. Unwanted items may be Deleted. Existing items may be modified or reordered.*

## Sprint Retrospective




# SPRINT RETROSPECTIVE

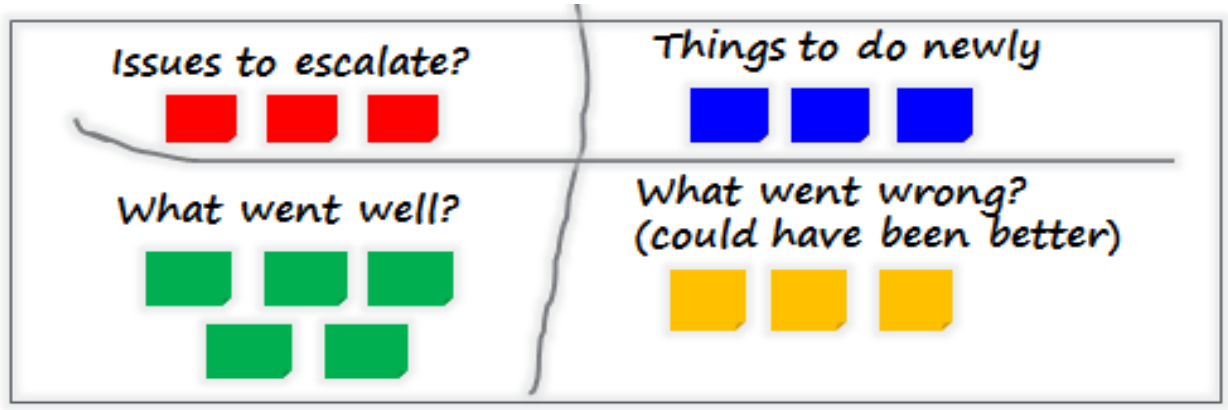
| | | |
|---|---|---|
| **Purpose:**<br>Identify how to become better based on the learnings from the current Sprint | **Participants:**<br>Scrum team only<br>**Mandatory:**<br>Scrum team only<br>**Optional:** None other than Scrum Team | **Timebox:**<br>Not more than 3 hours for 1 month Sprint, shorter for shorter Sprints<br>**When it occurs?**<br>Last day after the Sprint Review<br>**Frequency:**<br>Every Sprint once |

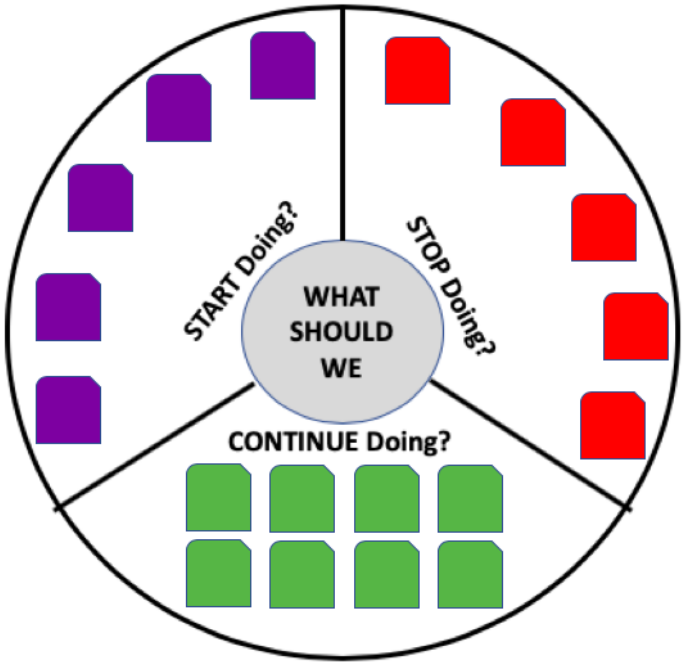
| | |
|---|---|
| **What is inspected?**<br>Scrum team with respect to people, collaboration, communication, tools, techniques, processes, practices, definition of done etc | **What is adapted?**<br>Improvements related to the elements mentioned to be implemented in the next Sprint |
| **Inputs:**<br>Current Sprint information: goal, items completed/not completed, capacity available/utilised, impediments for discussion | **Outputs:**<br>Actionable improvements list. If there are more, Scrum master to help team to prioritise using techniques like Dot Voting |

**Retrospective stages: (Not prescribed Scrum but effective practice)**

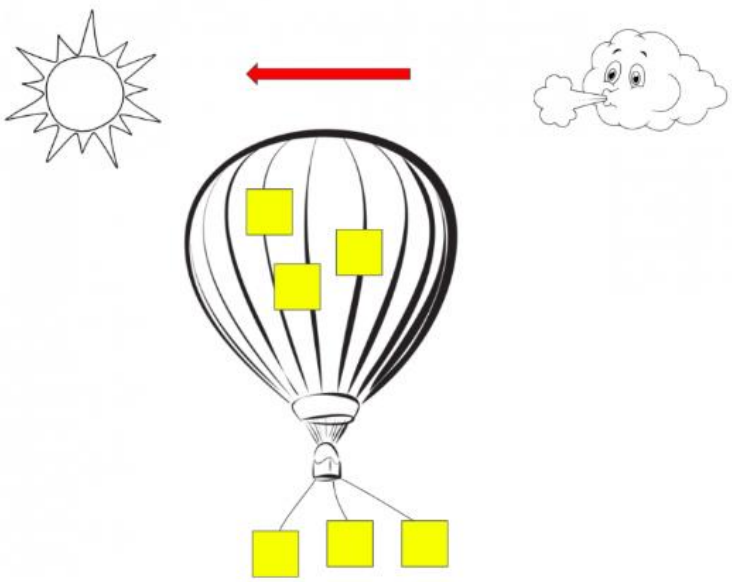
| | |
|---|---|
| Set the stage | • Lay the groundwork for the session by reviewing the goal and agenda.<br>• Create an environment for participation by checking in and establishing working agreements. |
| Gather data | • Review objective and subjective information to create a shared picture.<br>• Bring in each person's perspective. When the group sees the iteration from many points of view, they'll have greater insight. |
| Generate insights | • Step back and look at the picture the team created.<br>• Use activities that help people think together to delve beneath the surface. |
| Decide what to do | • Prioritize the team's insights and choose a few improvements or experiments that will make a difference for the team. |
| Close the retrospective | • Summarize how the team will follow up on plans and commitments. Thank team members for their hard work. Conduct a little retrospective on the retrospective, so you can improve too. |

**Examples to conduct effective retrospectives: (Not prescribed by Scrum)**



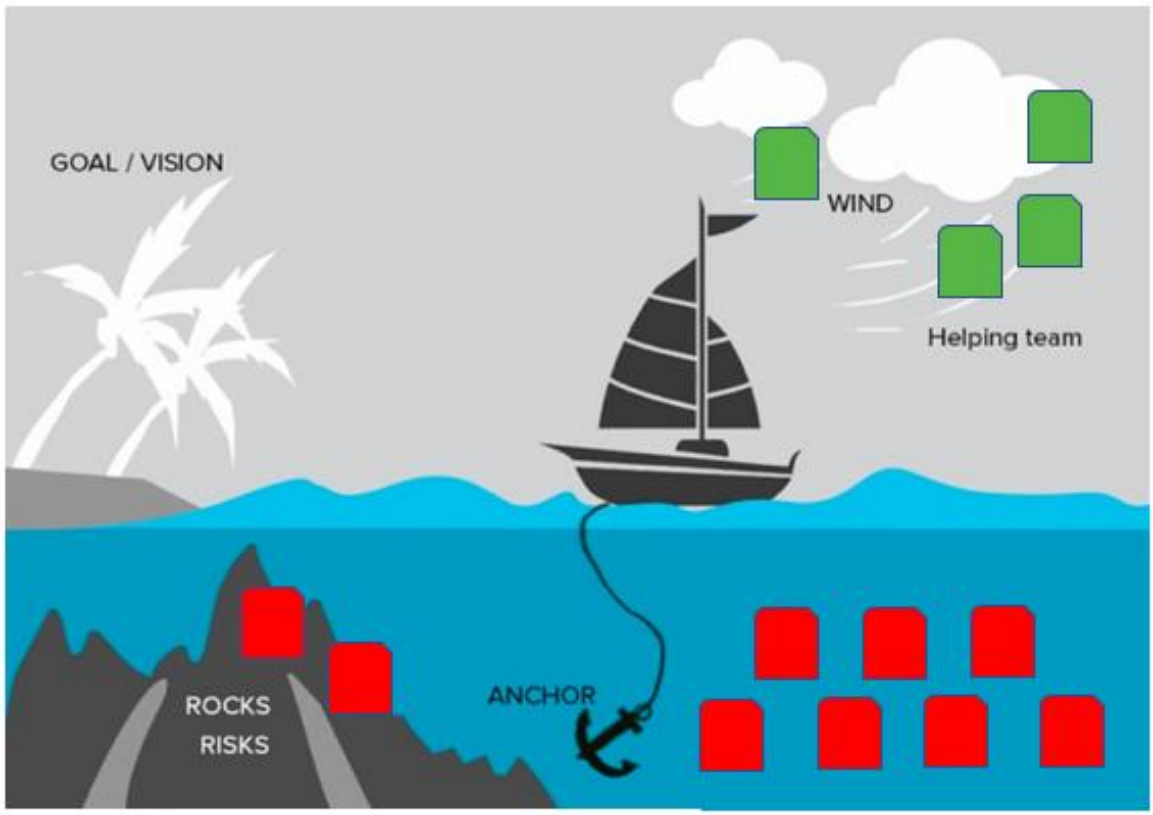



## Hot Air Balloon Technique



## 4L Model (Learned, Liked, Lacked, Long for)

## Speed boat Technique



# APPENDIX

## Space for Notes

## Space for Notes

## Space for Notes

## Space for Notes

## Appendix - 1: About CSM Exam

50 Questions
74% to pass (37 out of 50)

ONE HOUR EXAM DURATION

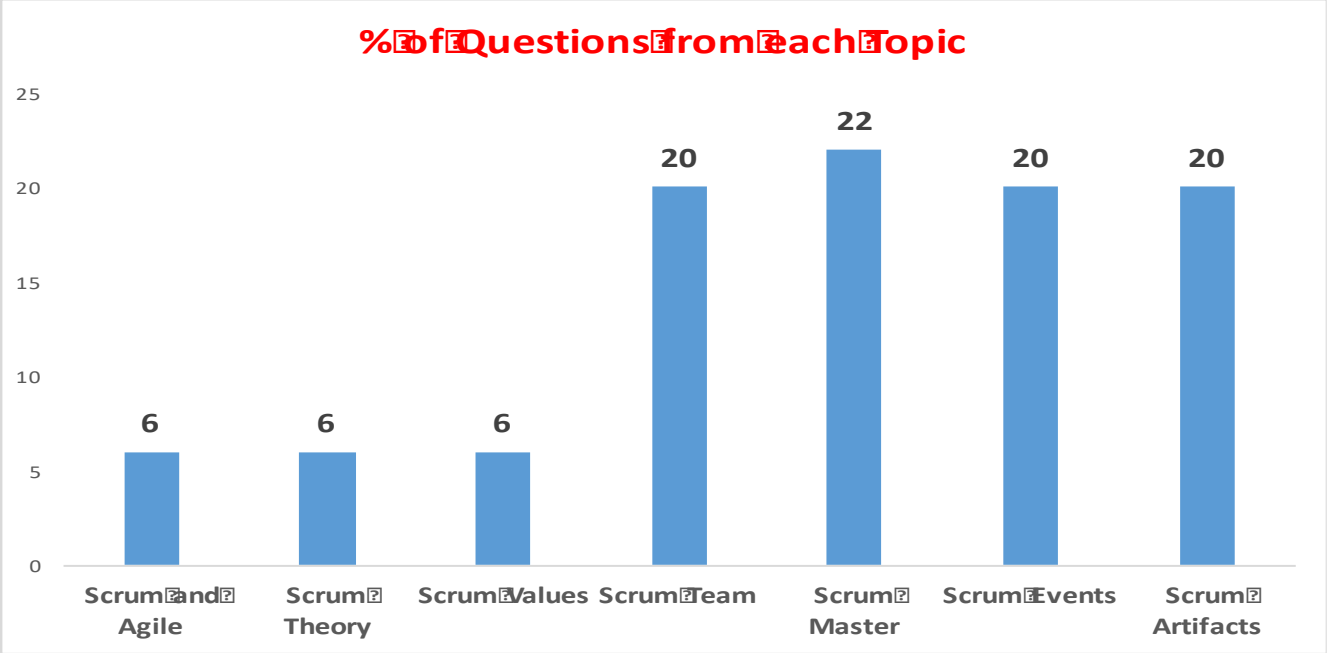
QUESTIONS WILL BE RELATED TO
7 AREAS OF AGILE AND SCRUM

EXAM LINK WILL BE SHARED TO YOUR REGISTERED EMAIL
LINK IS VALID FOR 90 DAYS ONLY
AFTER THAT YOU HAVE TO REQUEST THE FRESH LINK AGAIN
CSM EXAM IS OPEN BOOK TYPE
YOU CAN GIVE THE EXAM AT ANY PLACE AS PER YOUR CONVENIENCE
YOU CAN USE YOUR MOBILE/TAB/LAPTOP TO GIVE THE EXAM
IF YOU PLAN TO GIVE EXAM AT THE VENUE, YOU NEED TO HAVE YOUR NETWORK
YOU NEED TO CLICK THE LINK GIVEN IN THE EMAIL TO START THE EXAM
IT ASKS YOU TO FILL A FEW MANDATORY FIELDS ON YOUR PROFILE
ONCE YOU BEGIN THE EXAM, YOU CANNOT PAUSE THE TIMER
YOU CAN MARK QUESTIONS FOR REVIEW AND REVISIT THEM AGAIN
YOU CAN MOVE BACK AND FORTH DURING THE EXAM
THE RESULT WILL BE SHOWN IMMEDIATELY
AFTER THE EXAM YOU HAVE TO ACCEPT THE AGREEMENT
YOU CAN DOWNLOAD THE CERTIFICATE IMMEDIATELY AFTER THE ABOVE STEP

TWO ATTEMPTS ARE FREE, THIRD ATTEMPT WILL NEED 25 USD FEE
CERTIFICATION VALIDITY IS TWO YEARS
YOU NEED TO PAY 100 USD TO RENEW FOR ANOTHER TWO YEARS
HIGHER CERTIFICATION WILL RENEW ALL LOWER CERTIFICATIONS

NO NEGATIVE MARKING
EACH QUESTION HAS ONE MARK
EACH QUESTION WILL HAVE FOUR OPTIONS
EACH QUESTION WILL HAVE ONE CORRECT ANSWER

## Topic wise Questions Distribution for CSM Exam

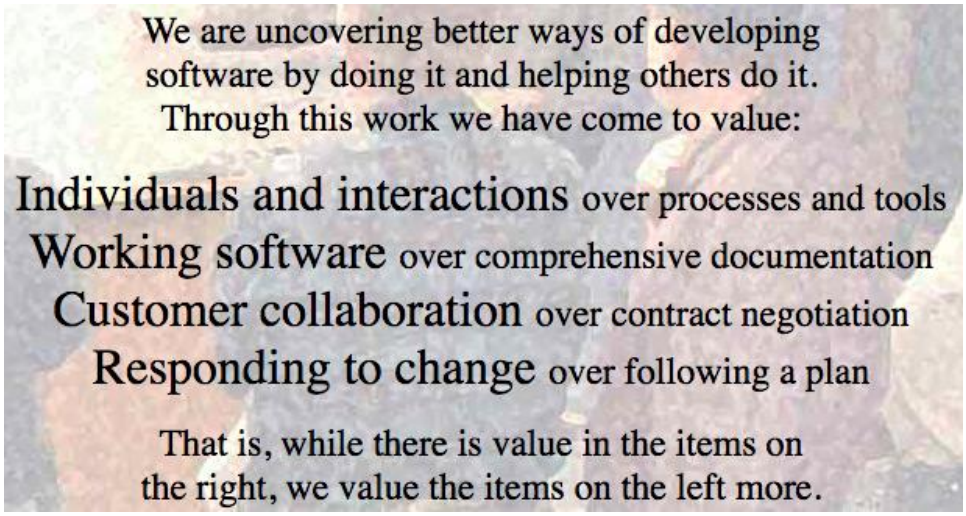


## Category wise topics list

| CSM Domains | CSM should demonstrate knowledge of... | % of CSM test |
|---|---|---|
| A. SCRUM AND AGILE | Four values of the Agile Manifesto | 6% |
| | Twelve principles of the Agile Manifesto | |
| | Definition of Scrum | |
| | Relationship of Scrum to Agile | |
| B. SCRUM THEORY | Empirical process control as it relates to Scrum | 6% |
| | The 3 pillars of empirical process control and their importance | |
| | How and why "***incremental***" is an important characteristic of Scrum | |
| | How and why "***iterative***" is an important characteristic of Scrum | |
| | Applicability of Scrum (addresses complex adaptative problems across multiple industries) | |
| C. SCRUM VALUES | Identify the five Scrum values | 6% |
| | How and why ***commitment*** is an important Scrum value | |
| | How and why ***courage*** is an important Scrum value | |
| | How and why ***focus*** is an important Scrum value | |
| | How and why ***openness*** is an important Scrum value | |
| | How and why ***respect*** is an important Scrum value | |
| D. SCRUM TEAM | Why self-organizing is an important characteristic of Scrum Teams | 20% |
| | Why cross-functional is an important characteristic of Scrum Teams | |
| | Identify the roles on the Scrum Team | |
| | Identify the responsibilities and characteristics of the Scrum Master | |
| | Identify the responsibilities and characteristics of the Scrum Product Owner | |
| | Identify the responsibilities and characteristics of the Scrum Development Team | |
| E. SCRUM MASTER | Understanding responsibilities and characteristics of the Scrum Master -- servant leader for the Scrum Team | 22% |
| | Scrum Master service to the Organization -- coaching, facilitation, removing impediments | |
| | Scrum Master service to the Development Team -- coaching, facilitation, removing impediments | |
| | Scrum Master service to the Product Owner -- coaching, facilitation, removing impediments | |
| F. SCRUM EVENTS | Characteristics, value and/or purpose of the Sprint | 20% |
| | Sprint Planning -- characteristics, value, purpose and/or role of participants | |
| | Daily Scrum -- characteristics, value, purpose and/or role of participants | |
| | Sprint Review -- characteristics, value, purpose and/or role of participants | |
| | Retrospective -- characteristics, value, purpose and/or role of participants | |
| G. SCRUM ARTIFACTS | Understand the purpose and value of Scrum artifacts | 20% |
| | Identify Scrum artifacts | |
| | Product Backlog - characteristics, value and purpose | |
| | Sprint Backlog -- characteristics, value and purpose | |
| | Increment -- characteristics, value and purpose | |
| | Understanding importance of transparency of artifacts to evaluate value and risk | |
| | Identify the downsides of lack of transparency | |
| | Importance of establishing the Definition of Done | |
| | Characteristics of Product Backlog items | |

# APPENDIX – 2: Agile Values & Principles

## 4 Values

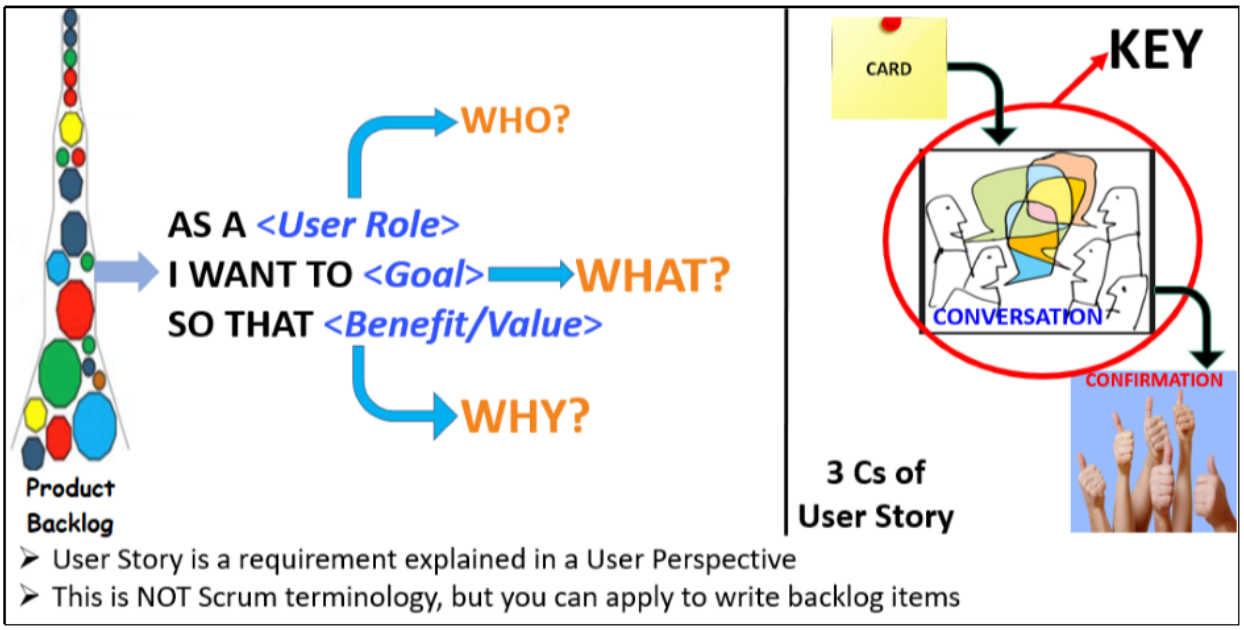


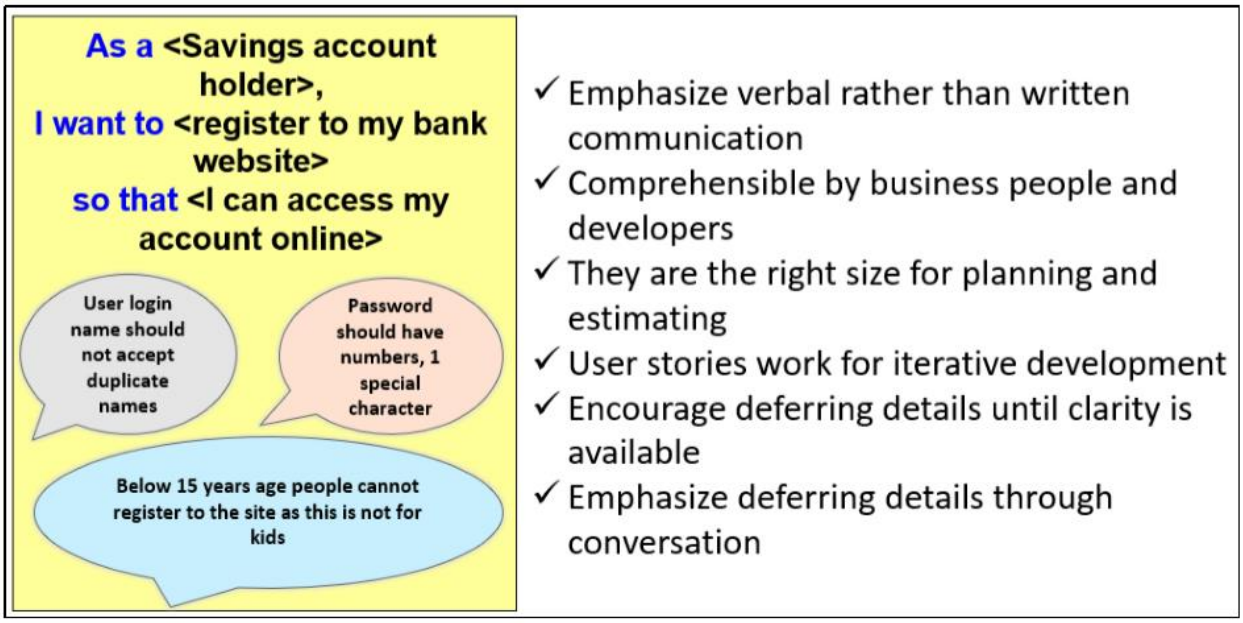

## 12 Principles

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.

## APPENDIX – 3: ADDITIONAL TOPICS (User Stories, Burndown, Release Planning & Scaling)

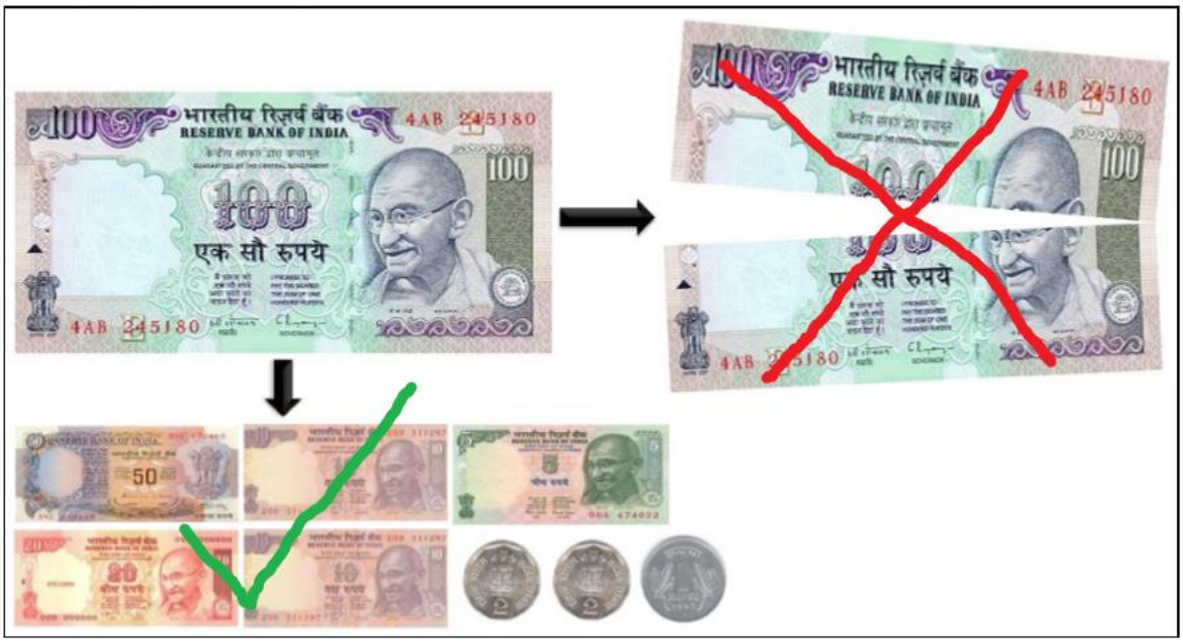
### User Story *(*** Not prescribed by Scrum ***)*

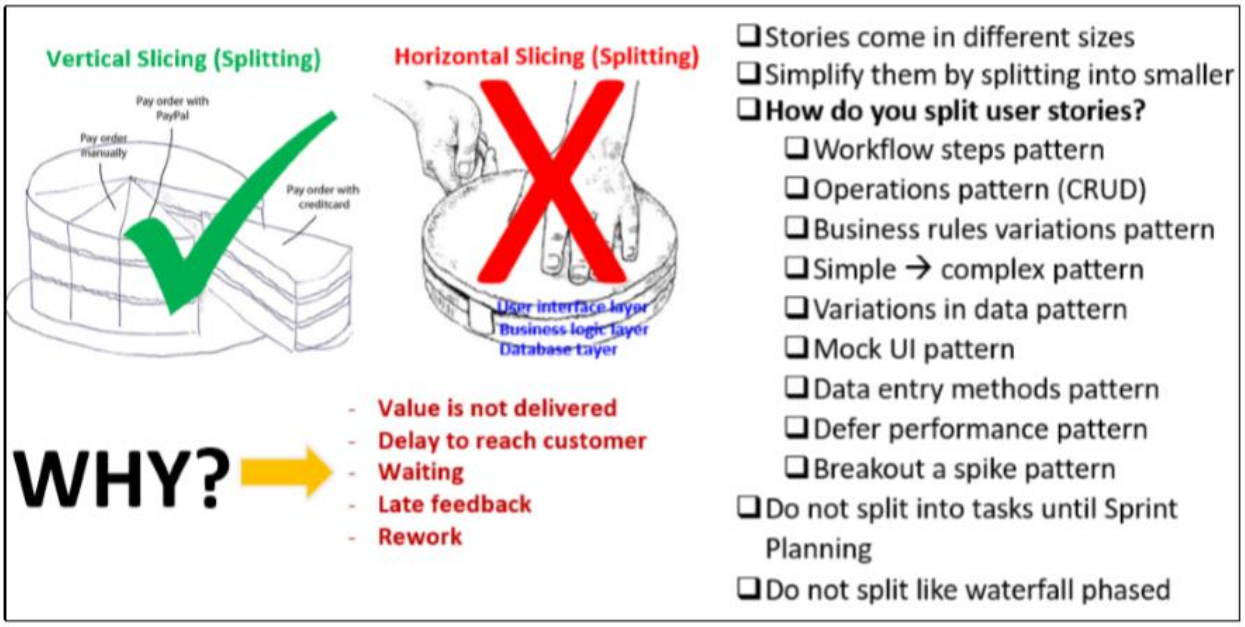


**Example:**


**Split stories if they are oversized:**






**What is a Good User Story?**

| | |
|---|---|
| **INDEPENDENT** | **The user story should be self-contained, in a way that there is no inherent dependency on another user story.** |
| **NEGOTIABLE** | **User stories are not explicit contracts and should leave space for discussion and collaboration.** |
| **VALUABLE** | **A user story must deliver value to the end user.** |
| **ESTIMABLE** | **You must always be able to estimate the size of a user story.** |
| **SMALL/SIZED PROPERLY** | **User stories should not be so big as to become impossible to plan/split/prioritize. They should be small enough so that they can be completed faster.** |
| **TESTABLE** | **The user story or its related description must provide the necessary information to make test development possible.** |

## Progress tracking through Burndown Chart *(Not part of Scrum)*

Burndown chart is the tool to show the work remaining in a Sprint. Team updates this as and when the work gets completed. That means whenever a product backlog item meets the "DoD" corresponding size of the backlog item (story points) will be updated on the chart. This helps Development team to track their progress and plan their work accordingly. Tasks can be re-estimated on daily basis based on the progress made and work to be done.

- Burndown chart helps team to be self-organized
- Burndown chart is **not a** management reporting tool
- It is the team's property, and none has any right to intervene or question on this
- Scrum Master should encourage the team to update the remaining effort on daily basis on their own
- Burndown chart should be placed in a place where it can be viewed by all team members
- Team should have good understanding on how to understand the Burndown chart

As a definition of this chart we can say that the Burndown chart displays the *remaining work for a given period*. Progress of a Sprint can be tracked using Sprint burndown and overall progress of a release can be tracked through Release burndown. Development team owns the Sprint burndown and Product owner owns the Release burndown. Sprint burndown has to be updated at least once in a day and release burndown is updated at the end of every sprint. Release burn down helps Product owner to track progress and communicate to stakeholders effectively.

Burnup chart also very similar but it tracks the "Cumulative completed work". You can use burnup charts also for Sprint level and Release level.

**Burndown Charts – Types (Recommendation is to use Remaining size type to get realistic view)**

**Sprint Burndown Chart X and Y axis:**

X axis to display working days of the Sprint. So if the Sprint duration is 2 weeks then it will have Day1, Day2, …Day10.

Y axis to display remaining Story Points

Ideal effort as a guideline (dotted line)

Actual progress of effort (solid line)

**Velocity (Not prescribed by Scrum):**

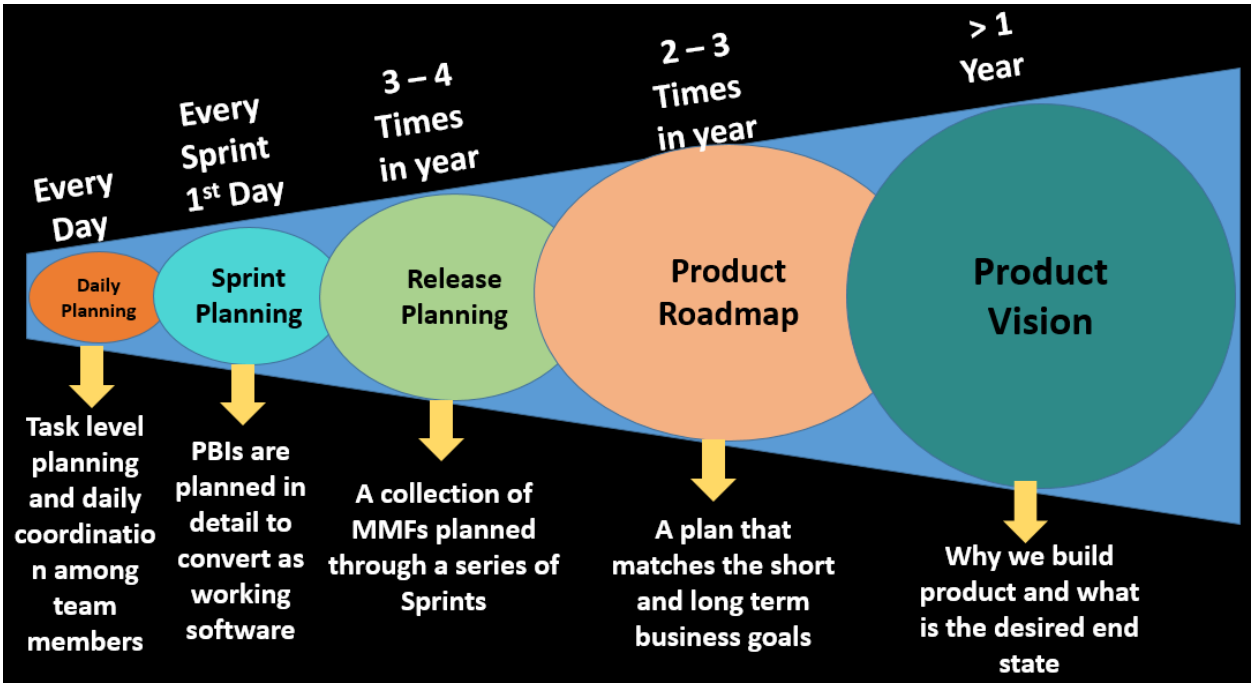
Velocity is a measure of the amount of work a team can tackle in a single sprint. Velocity is calculated at the end of the sprint by totalling the story points for all the fully completed stories in the sprint.

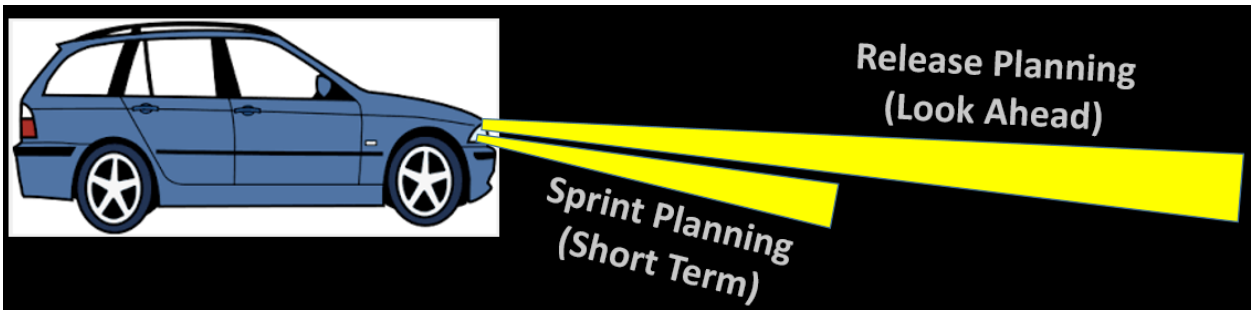
**Key points related to Velocity:**

- Velocity is a forecast, not a commitment
- Velocity cannot be used to compare teams
- Stories that are done 100% will only be considered for Velocity calculation
- Velocity is at the team level, not at individual level

## Release Planning in Scrum (*Not part of Scrum*)

**Planning Levels:**




A very high-level plan for multiple sprints (E.g. 3 – 12 iterations) is done during Scrum release planning

It is a guideline that reflects expectations about which features will be implemented and when they are completed. It also serves as base to monitor progress with the project.

Releases can be intermediate deliverables done during the project or the final delivery at the end.

**Prerequisites to create a Release Plan:**

- A prioritized and estimated Product Backlog

- The estimated Velocity of the scrum team
- Conditions of satisfaction (goals for the schedule, scope, resources, cost etc.)

Depending on the type of project (Feature driven or Date driven) the release plan can be created in different ways:


**If the project is feature-driven:** The sum of all features within a release can be divided by the expected velocity. This will then result in the number of sprints needed to complete the requested functionality.

**If the project is Date-driven**: we can simply multiply the velocity by the number of sprints and we will get the total work that can be completed within the given timeline.

Like the Product Backlog, Release plan is not a static plan. It will change when new knowledge is available, like entries in the product backlog are added, re-estimated.

The release plan should be revisited at regular intervals (ideally at the end of every sprint)

**Feature Driven Release Plan:**

**Release goals:**

1. First release is after user is able to login & logout
2. Second release is after user can manage items
3. Final release is when all other stories are complete

**First Release:**

Stories: 7, 1, 10 → Estimation = 5 points

Estimation/Velocity = 5/3 = 2 Sprints

**Second Release:**

Stories: 9, 2, 4, 3, 5 → Estimation = 15 points

Estimation/Velocity = 15/3 = 5 Sprints

**Final Release:**

All remaining stories → Estimation = 10 points
Estimation/Velocity = 10/3 = 3 Sprints

| ID | Story | Estimate | Priority |
|---|---|---|---|
| 7 | As an authorized user I want to create a new account | 3 | 1 |
| 1 | As an authorized user I want to login | 1 | 2 |
| 10 | As an authorized user I want to logout | 1 | 3 |
| 9 | Create script to purge database | 1 | 4 |
| 2 | As an authorized user I want to see the items list | 2 | 5 |
| 4 | As an authorized user I want to add a new item | 5 | 6 |
| 3 | As an authorized user I want to edit item | 2 | 7 |
| 5 | As an authorized user I want to search for item | 5 | 8 |
| 6 | As an authorized user I want to checkout | 8 | 9 |
| 8 | As an authorized user I want to pay using PayPal | 2 | 10 |
| | TOTAL Stories size in Story Points | 30 | |

**Velocity: 3 points/Sprint** **Release after Sprints 2, 7 and 10**

**Date Driven Release Plan:**

**Release goals:**

1. Release every 6 weeks

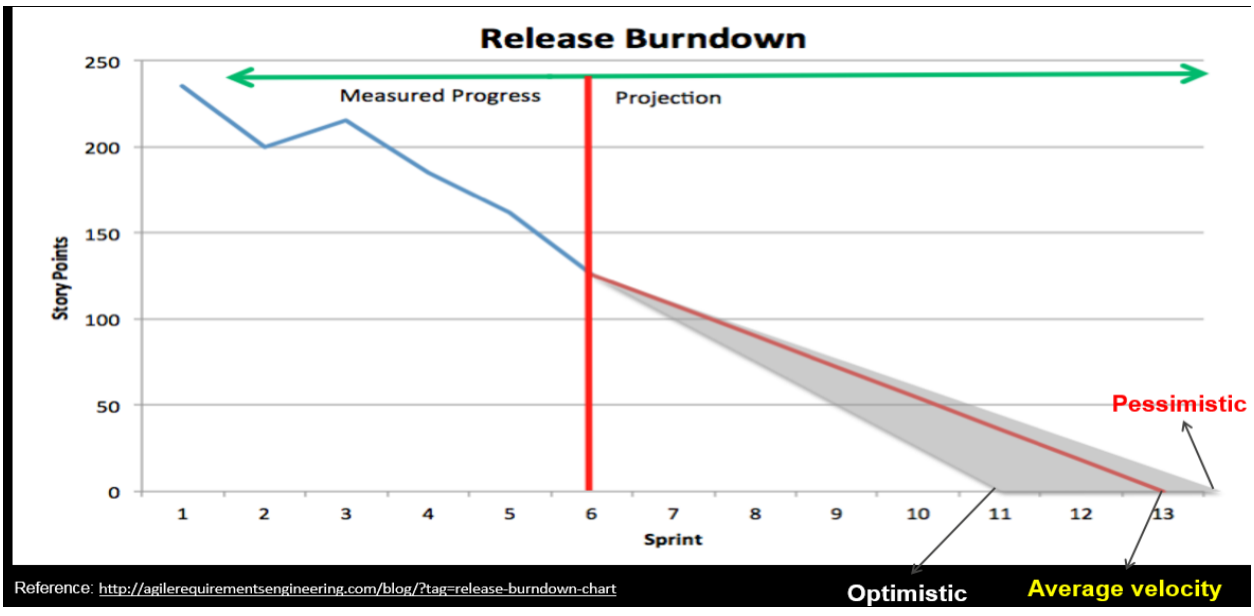
| ID | Story | Estimate | Priority |
|---|---|---|---|
| 7 | As an authorized user I want to create a new account | 3 | 1 |
| 1 | As an authorized user I want to login | 1 | 2 |
| 10 | As an authorized user I want to logout | 1 | 3 |
| 9 | Create script to purge database | 1 | 4 |
| 2 | As an authorized user I want to see the items list | 2 | 5 |
| 4 | As an authorized user I want to add a new item | 5 | 6 |
| 3 | As an authorized user I want to edit item | 2 | 7 |
| 5 | As an authorized user I want to search for item | 5 | 8 |
| 6 | As an authorized user I want to checkout | 8 | 9 |
| 8 | As an authorized user I want to pay using PayPal | 2 | 10 |
| | TOTAL Stories size in Story Points | 30 | |

- The duration of Release Planning meeting will vary depending on sprint length. As a general rule of thumb, the team should allocate 5 to 15 percent of the team's total

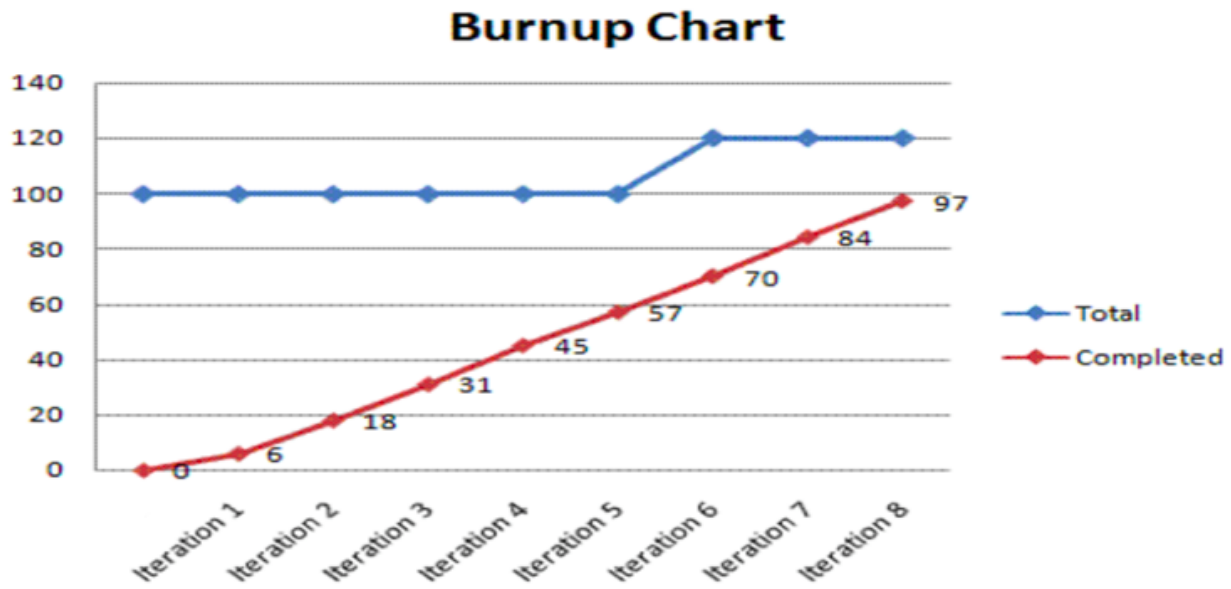
available hours to this meeting early in projects. As the project matures, this time may drop to as low as four hours per sprint.

- The product owner, Development team, Scrum Master and the stakeholders can be part of the Release Planning meeting
- Release Planning meeting can happen during or at the end of last sprint of the previous release
- This meeting is also called as "Grooming" meeting
- Release backlog is the subset of Product Backlog which consists of the stories/features targeted for next current release
- **The "Release Plan" contains:**
    - Product Backlog with items tagged to the Release
    - Assumptions
    - Dependencies
    - Risks
    - Action Items
    - Teams participating in the Release

**Tracking Progress through release Burndown chart:**



**Tracking Progress through release Burnup chart:**



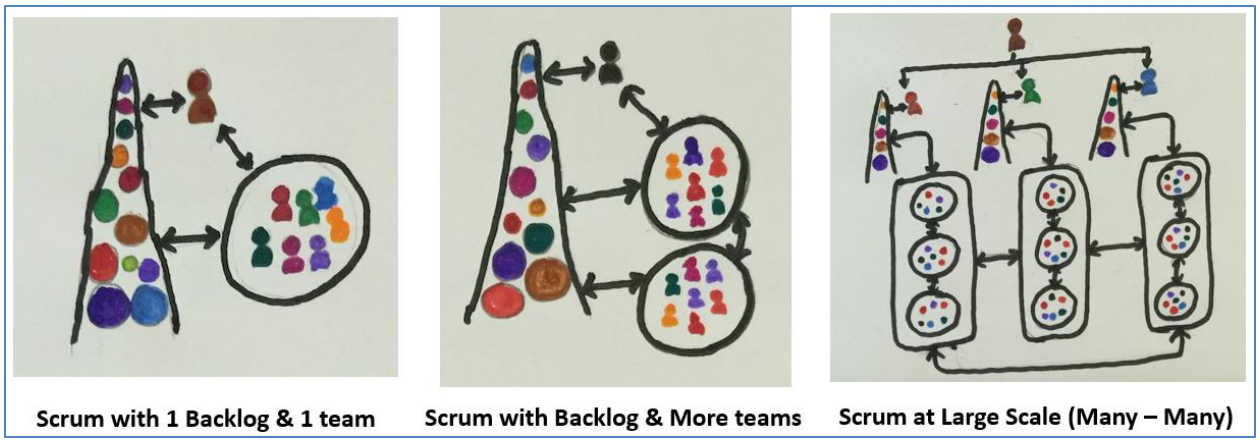

Blue line indicates the overall release backlog size. Till the 5th Sprint, it is same but in Sprint 6 it has increased, that means some new work is added. Red line indicates the cumulative completion of the work in every Sprint. So, at the 8th Sprint, "Remaining work to be complete" can be found by subtracting red line value (work complete) from the blue line value (Total work). So, in the above it is 120 minus 97 which is 23. Now, to understand how many more sprints it is needed to complete the 23 points of work, you have to check the average velocity of all 8 Sprints. In the above it is: 12.125, let's take 12. So now divide the 23 (remaining work) with 12 (Velocity) which gives the number of Sprints required to complete. 23/12 = 2 (rounded). This is how the Product owner will track the overall release progress.

SPACE FOR NOTES

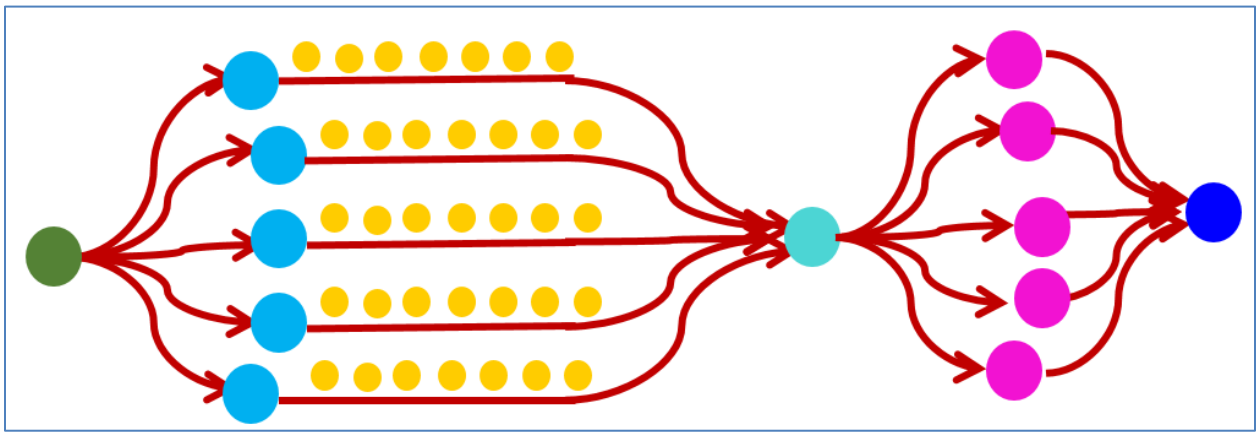
## Scaling Scrum *(Not part of Scrum)*

Scaling comes into picture when more scrum teams work on a product. There are different scaling frameworks available to choose such as LeSS (Large Scale Scrum), SaS (Scrum at Scale), DAD (Disciplined Agile Delivery). The fundamental way of working in these frameworks is Scrum.

Scaling can be understood from the following diagram.




Below is a high-level information of LeSS framework.



- **Sprint Planning Part 1:** In addition to the one Product Owner, it includes people from all teams. Let team members self-manage to decide their division of Product Backlog Items. Team members also discuss opportunities to find shared work and cooperate, especially for related items.
- **Sprint Planning Part 2:** This is held independently (and usually in parallel) by each Team, though sometimes for simple coordination and learning two or more Teams may hold it in the same room (in different areas).
- **Daily Scrum:** This is also held independently by each Team, though a member of Team A may observe Team B's Daily Scrum, to increase information sharing.
- **Coordination:** Just Talk, Communicate in Code, Travelers, Open Space, and Communities.

- **Overall PBR:** There may be an optional and short overall Product Backlog Refinement (PBR) meeting that includes the one Product Owner and people from all teams. The key purpose is to decide which teams are likely to implement which items and therefore select those items for later in-depth single-team PBR. It is also a chance to increase alignment with the Product Owner and all teams.
- **Product Backlog Refinement:** The only requirement in LeSS is single-team PBR, the same as in one-team Scrum. But a common and useful variation is multi-team PBR, where two or more Teams are in the same room together, to increase learning and coordination.
- **Sprint Review:** In addition to the one Product Owner, it includes people from all teams, and relevant customers/users and other stakeholders. For the phase of inspecting the product increment and new items, consider a "bazaar" or "science fair" style: a large room with multiple areas, each staffed by team members, where the items developed by teams are shown and discussed.
- **Team level retrospective:** This is same as the Scrum retrospective. Each team conducts the retrospective for their sprint and identify improvements.
- **Overall Retrospective:** This is a new meeting not found in one-team Scrum, and its purpose is to explore improving the overall system, rather than focusing on one Team. The maximum duration is 45 minutes per week of Sprint. It includes the Product Owner, Scrum Masters, and rotating representatives from each Team.


SPACE FOR NOTES

# APPENDIX – 4: HISTORY OF SCRUM

Jeff Sutherland and Ken Schwaber conceived the Scrum process in the early 90's. They codified Scrum in 1995 to present it at the OOPSLA conference in Austin, Texas (US) and published the paper "SCRUM Software Development Process".

Ken and Jeff inherited the name 'Scrum' from the 1986 groundbreaking paper 'The New New Product Development Game' by Takeuchi and Nonaka, two acknowledged management thinkers. With the term 'Scrum' Nonaka and Takeuchi referred to the game of rugby to stress the importance of teams and some analogies between a team sport like rugby and being successful in the game of new product development. The research described in their paper showed that outstanding performance in the development of new, complex products is achieved when teams, as small and self-organizing units of people, are fed with objectives, not with tasks. The best teams are those that are given direction within which they have room to devise their own tactics on how to best head towards their joint objective. Teams require autonomy to achieve excellence.

The Scrum framework for software development implements the principles described in this paper for developing and sustaining complex software products.

While in the process of developing and using early versions of Scrum, Ken asked Professor Babatunde A. Ogunnaike Tunde, a famous process control research engineer, to look at software development processes. Tunde investigated several commercial software-development methodologies to conclude that the waterfall and predictive process is not a good fit for the work of software development. He confirmed the empirical approach of Scrum to be the preferred process.

Empiricism is used for complex work where more is unknown than is known and predictions have little value given a high rate of change and uncertainty.

Scrum was first tried and refined at Individual, Inc., Fidelity Investments, and IDX (now GE Health).

In February 2001, Jeff and Ken were amongst the 17 software development leaders creating the Manifesto for Agile Software Development.

Following the Agile Manifesto, the Agile Alliance was founded with Ken Schwaber being its first chairman.

In 2001, much inspired by Kent Beck, Ken Schwaber co-authored the first book on Scrum with Mike Beedle, Agile Software Development with Scrum.

In 2002, Ken Schwaber founded the Scrum Alliance with Mike Cohn and Esther Derby, with Ken chairing the organization. In the years to follow the highly successful Certified ScrumMaster programs and its derivatives were created and launched.

In 2006, Jeff Sutherland created his own company, Scrum.inc, while continuing to offer and teach the Certified Scrum courses.

Ken left the Scrum Alliance in the fall of 2009, and founded Scrum.org to further improve the quality and effectiveness of Scrum, mainly through the Professional Scrum series.

With the first publication of the Scrum Guide in 2010, and its incremental updates in 2011, 2013, 2016, and 2017 Jeff and Ken established the globally recognized body of knowledge of Scrum.

Ever since its first publication in 1995 up to now, Scrum has been adopted by a vast amount of software development companies around the world. It is today recognized as the most applied framework for agile software development. More than 1000 books have been published on Scrum.

The method however has also been successfully applied in other domains, e.g. manufacturing, marketing, operations and education.

With their continual work at their respective companies Ken Schwaber and Jeff Sutherland relentlessly keep setting the vision for success with Scrum.

# APPENDIX – 5: SCRUM MASTER CHECKLIST

## An Example Checklist for ScrumMasters

Michael James
(mj4scrum@gmail.com)
14 September 2007
(Revised 2 Feb 2016)

### A Full Time Facilitator?

An adequate ScrumMaster can handle two or three teams at a time. If you're content to limit your role to organizing meetings, enforcing timeboxes, and responding to the impediments people explicitly report, you can get by with part time attention to this role. The team will probably still exceed the baseline, pre-Scrum expectation at your organization, and probably nothing catastrophic will happen.

But if you can envision a team that has a great time accomplishing things no one previously thought possible, within a transformed organization, consider being a *great* ScrumMaster.

A great ScrumMaster can handle *one* team at a time.

We recommend one dedicated ScrumMaster per team of about seven when starting out.

If you haven't discovered all the work there is to do, tune in to your Product Owner, your team, your team's engineering practices, and the organization outside your team. While there's no single prescription for everyone, I've outlined typical things I've seen ScrumMasters overlook. Please mark each box with √, Δ, ?, or N/A, as described on the last page.

### Part I -- How Is My Product Owner Doing?

ScrumMasters improve Product Owner effectiveness by helping them find ways to maintain the Product Backlog and release plan. (Note that the Product Owner is the one responsible for the prioritized backlog.)

☐ Is the Product Backlog prioritized according to his/her latest thinking?

☐ Are requirements and desirements from all stakeholders captured in the Product Backlog? Remember: the backlog is *emergent*.

☐ Is the Product Backlog a manageable size? To maintain a manageable number of items, keep things more granular towards the top, with general epics at the bottom. It's counterproductive to overanalyze too far past the top of the Product Backlog. Your requirements will change in an ongoing conversation between the developing product and the stakeholders/customers.

☐ Could any requirements (especially those near the top of the Product Backlog) be better expressed as independent, negotiable, valuable, estimable, small, and testable user stories[1]?

☐ Have you educated your Product Owner about *technical debt* and how to avoid it? One piece of the puzzle may be to write automated test and refactoring into the definition of "done" for each backlog item.

☐ Is the backlog an *information radiator,* immediately visible to all stakeholders?

☐ If you're using an automated tool for backlog management, does everyone know how to use it easily? Automated management tools introduce the danger of becoming *information refrigerators* without active radiation from the ScrumMaster.

☐ Can you help radiate information by showing everyone printouts?

☐ Can you help radiate information by creating big visible charts?

☐ Have you helped your Product Owner organize backlog items into appropriate releases or priority groups?

☐ Does everyone know whether the release plan still matches reality? You might try showing everyone Product/ Release Burndown Charts[2] after the items have been acknowledged as “done” during every Sprint Review Meeting. Charts showing both the rate of PBIs actually completed and new ones added allow early discovery of scope/schedule drift.

☐ Did your Product Owner adjust the release plan after the last Sprint Review Meeting? The minority of Product Owners who ship adequately tested products on time *re-plan* the release every Sprint. This probably requires deferring some work for future releases as more important work is discovered.

**Part II -- How Is My Team Doing?**

[1] http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/
[2] Mike Cohn, *Agile Estimation and Planning*. (2005).

While you are encouraged to lead by the example of collaborating with team members on their work, there is a risk you will get lost in technical tasks. Consider your primary responsibilities to the team:

☐ Is your team in the state of *flow*? Some characteristics of this state[3]:

- Clear goals (expectations and rules are discernible and goals are attainable, aligning appropriately with one's skill set and abilities).
- Concentration and focus, a high degree of concentration on a limited field of attention.
- A loss of the feeling of self-consciousness, the merging of action and awareness.
- Direct and immediate feedback (successes and failures in the course of the activity are apparent, so that behavior can be adjusted as needed).
- Balance between ability level and challenge (the activity is neither too easy nor too difficult).
- A sense of personal control over the situation or activity.
- The activity is intrinsically rewarding, so there is an effortlessness of action.

☐ Do team members seem to like each other, goof off together, and celebrate each other's success?

☐ Do team members hold each other accountable to high standards, and challenge each other to grow?

☐ Are there issues/opportunities the team isn't discussing because they're too uncomfortable?[4]

☐ Have you tried a variety of formats and locations for Sprint Retrospective Meetings?[5]

☐ Has the team kept focus on Sprint goals? Perhaps you should conduct a mid-Sprint checkup to re-review the acceptance criteria of the Product Backlog Items committed for this Sprint.

☐ Does the Sprint taskboard reflect what the team is actually doing? Beware the "dark matter" of undisclosed tasks and tasks bigger than one day's work. Tasks not related to Sprint commitments are impediments to those commitments.

[3] Mihaly Csikszentmihalyi, *Flow: The Psychology of Optimal Experience* (1990).

[4] Marshall Rosenberg, *Nonviolent Communication: A Language of Life: Life-Changing Tools for Healthy Relationships* (2003). Also consider enlisting a professional facilitator who can make uncomfortable conversations more comfortable.

[5] Derby/Larson *Agile Retrospectives: Making Good Teams Great* (2006).

☐ Does your team have 3-9 people with a sufficient mix of skills to build a potentially shippable product increment?

☐ Is your team's taskboard up to date?

☐ Are the team self-management artifacts visible to the team, convenient for the team to use?

☐ Are these artifacts adequately protected from meddlers? Excess scrutiny of daily activity by people outside the team may impede team internal transparency and self-management.

☐ Do team members volunteer for tasks?

☐ Has the need for technical debt repayment been made explicit in the definition of *done*, gradually making the code a more pleasant place to work?

☐ Are team members leaving their job titles at the door of the team room, collectively responsible for all aspects of agreed work (testing, user documentation, etc.)?

### Part III -- How Are Our Engineering Practices Doing?

☐ Does your system in development have a "push to test" button allowing anyone (same team or different team) to conveniently detect when they've caused a regression failure (broken previously-working functionality)? Typically this is achieved through the xUnit framework (JUnit, NUnit, etc.).

☐ Do you have an appropriate balance of automated end-to-end system tests (a.k.a. "functional tests") and automated unit tests?

☐ Is the team writing both system tests and unit tests in the same language as the system they're developing? Collaboration is not enhanced by proprietary scripting languages or capture playback tools that only a subset of the team knows how to maintain.

☐ Has your team discovered the useful gray area between system tests and unit tests[6]?

☐ Does a continuous integration[7] server automatically sound an alarm when someone causes a regression failure? Can this feedback loop be reduced to hours or minutes? ("Daily builds are for wimps." -- Kent Beck)

☐ Do *all* tests roll up into the continuous integration server result?

☐ Have team members discovered the joy of continuous design and constant refactoring[8], as an alternative to Big Up Front Design? Refactoring has a strict definition: changing internal structure without changing external behavior. Refactoring should occur several times per hour, whenever there is duplicate code, complex conditional logic (visible by excess indenting or long methods), poorly named identifiers, excessive coupling between objects, etc. Refactoring with confidence is only possible with automated test coverage. Neglecting refactoring makes it hard to change the product in the future, especially since it's hard to find good developers willing to work on bad code.

☐ Does your definition of "done" for each Product Backlog Item include full automated test coverage and refactoring? Learning Test Driven Development (TDD) increases the probability of achieving this.

☐ Are team members pair programming most of the time? Pair programming may dramatically increase code maintainability and reduce bug rates. It challenges people's boundaries and sometimes seems to take longer (if we measure by lines of code rather than shippable functionality). Lead by example by initiating paired workdays with team members. Some of them will start to prefer working this way.

### Part IV -- How Is The Organization Doing?

☐ Is the appropriate amount of inter-team communication happening? "Scrum of Scrums" is only one way to achieve this, and rarely the best.[9]

[6] http://blogs.collab.net/agile/junit-is-not-just-for-unit-testing-anymore

[7] http://www.martinfowler.com/articles/continuousIntegration.html

[8] Martin Fowler, *Refactoring: Improving the Design of Existing Code* (1999).
[9] See http://less.works/less/framework/coordination-and-integration.html for alternatives.

☐ Are teams independently able to produce working features, even spanning architectural boundaries?[10]

☐ Are your ScrumMasters meeting with each other, working the organizational impediments list?

☐ When appropriate, are the organizational impediments pasted to the wall of the development director's office? Can the cost be quantified in dollars, lost time to market, lost quality, or lost customer opportunities? (But learn from Ken Schwaber's mistakes: "A dead ScrumMaster is a useless ScrumMaster."[11])

☐ Is your organization one of the few with career paths compatible with the collective goals of your teams? Answer "no" if there's a career incentive[12] to do programming or architecture work at the expense of testing, test automation, or user documentation.

☐ Has your organization been recognized by the trade press or other independent sources as one of the best places to work, or a leader in your industry?

☐ Are you creating a *learning organization*?

## Conclusion

If you can check off most of these items and still have time left during the day, I'd like to hear from you.

There's no canned formula for creating human ingenuity. This paper lists points which may, or may not, help in your situation.

Once you start to realize what you could do to make a difference, you may find yourself afraid to do it. This is a sign you're on the right track

[10] http://FeatureTeamPrimer.org/

[11] Ken Schwaber, *Agile Project Management with Scrum* (2004)

[12] Alfie Kohn, *Punished By Rewards: The Trouble with Gold Stars, Incentive Plans, A's, Praise, and Other Bribes* (1999)

# APPENDIX – 6: HOW TO BEGIN SCRUM IMPLEMENTATION

## 1. Pick a Product Owner

This person is the one with the Vision of what you are going to do, make or accomplish. Takes into account Risks, rewards, what is possible, what can be done, and what she/he is passionate about.

## 2. Pick a Development Team

These are the people who are actually doing the work of the Product that you build as per the vision of the Product Owner. This team needs to have all the skills needed to take a Product Owner's vision and make it reality. They should have these skills as a team.

## 3. Pick a Scrum Master

This is the person who will coach the rest of the team through the Scrum framework and help the team eliminate anything that is slowing them down.

## 4. Create a Prioritize a Product Backlog

This is a list at a high level of everything that needs to be built or done to make the vision a reality. This backlog evolves over time and is a living artifact. It is the product roadmap. Only a single Product backlog exists. Product owns the Product Backlog and manage it throughout the Product development.

## 5. Refine and estimate the Product Backlog

It is critical that the people who are actually going to complete the item in Product backlog estimates how much effort they will take. Estimation method is up to them, they can use any technique. The team should look at each backlog item and see if it is doable. IS there complete information to complete the item (like acceptance criteria, user interface requirements and so on). Is there a definition of Done? Is there enough architecture and infrastructure to build the backlog items?

## 6. Sprint Planning

This is the first Scrum meeting. The Product owner, the Scrum Master and the Team sit down to plan the work for the Sprint. Sprints are always fixed duration that should not exceed a month. They can decide what should be the suitable duration below one month. The team looks at the Product backlog and forecasts how much they can complete in this Sprint. For this forecast they can consider their past performance (some times called as Velocity) and also their available capacity of this Sprint. The Scrum team should agree on a Sprint Goal in this meeting. No changes that endanger the Sprint Goal are allowed during the Sprint.

## 7. Make work visible

You can make the work visible by creating a visual board with three columns: To-do, Doing, Done and keep all the items the team has selected (forecasted) for the current Sprint. As and when items are started the team has to move them into "Doing" and as and when they are done, they need to move it to "Done". This has to be done by the Team and keep it up-to-date. The team can also use Burndown chart to visually represent the "work remaining" against the Sprint timeline.

## 8. Daily Scrum

This is the heartbeat of the Scrum. Each working day, same time, same place for no more than 15 minutes, the Team and Scrum Master meet and answer three questions.

1. **What did I do yesterday to help team to meet the Sprint goal?**
2. **What I am going to do today to help the team to meet the Sprint Goal?**
3. **Is there any obstacle for me or the team?**

If it is taking more than 15 minutes you are doing it wrong. Do not try to resolve the impediments and issues in that meeting. If required, you can have a quick separate meeting immediately to resolve the impediments. The Scrum Master is responsible to address the impediments that the team cannot resolve on their own.

## 9. Backlog Refinement

Make sure that Product Backlog Refinement happens by collaborating with Product Owner and Development team during the Sprint.

## 10. Sprint Review

In this meeting, the Scrum team and the Stakeholders invited by the Product owner participate. This is to inspect the increment built by the team in this Sprint with Stakeholders and adapt the Product backlog with their feedback. Only items meeting the definition of done are demonstrated in this meeting.

## 11. Sprint Retrospective

This is the last meeting in the Sprint. This is Scrum team's private meeting. All Scrum team members (Product Owner, Scrum Master and the Development team) sit together and inspect and adapt on the current Sprint's work with respect to the People, collaboration, communication, tools, definition of done, practice and so on. The Scrum team should aim for identifying at least one actionable improvements from this meeting and they must implement them in the upcoming Sprint to become better. This is not a blame game, this is fact finding activity. This improvement should be added to the next Sprint backlog so that the team can easily remember this and work on it.

## 12. Repeat the Sprint cycle

## Scrum Checklist (Helps to implement Scrum effectively)

**THE BOTTOM LINE**

***If you achieve the following three items in every Sprint, you can ignore the rest of the checklist.***

- ❑ Deliver working software that is potentially releasable in every 4 weeks or less
- ❑ Delivery focus is on business value first
- ❑ Process, People, Practices, Techniques, Communication, Collaboration etc are continuously improving

**CORE SCRUM**

***These are essential and critical to Scrum, without practicing these elements, if you are doing Scrum, you should not call it Scrum!!***

- ❑ HAVE DEFINITION OF DONE
  - ❑ Development Team must ensure that every Sprint they achieve the Definition of Done
  - ❑ Development Team respects their Definition of Done
  - ❑ Definition of Done is often inspected and adapted during Retrospective
  - ❑ In a multi-team environment a mutually agreed Definition of Done is created
  - ❑ Definition of Done is visible to the whole team and everyone in the Scrum team has same understanding on the Definition of Done criteria

- ❑ PRODUCT BACKLOG
  - ❑ Product Owner has authority on Product Backlog
  - ❑ Product Owner always keep top items ordered based on Business Value, Cost and Risk
  - ❑ Top few items are estimated (at least 2 Sprints worth of items)
  - ❑ Estimates are given by the Development Team
  - ❑ Top items in the Product Backlog are small and more detailed
  - ❑ Product Owner clearly defines the top few backlog items with acceptance criteria, any user interface requirements, validations and so on
  - ❑ Product Backlog refinement is happening at least once in every Sprint and whole team attends that
  - ❑ Product Backlog is highly visible

- ❑ TIMEBOXING
  - ❑ Make sure your Sprints are within 4 weeks or less
  - ❑ Fixed duration for every Sprint and no changes or gaps in between Sprints
  - ❑ No extensions to the Sprint duration
  - ❑ Development team is not disturbed during the Sprint

- ❑ Keep all events timeboxed
- ❑ Development Team delivers what they have forecasted by the end of every Sprint
- ❑ If Sprint goal has no value, it should be terminated early without waiting for approvals (Product Owner should have this authority)

- ❑ CLEARLY DEFINED PRODUCT OWNER ROLE
  - ❑ Product Owner is empowered to make decisions
  - ❑ Product Owner has in depth domain knowledge
  - ❑ Product Owner works closely and directly with the Development Team
  - ❑ Product Owner has direct contact with the Stakeholders
  - ❑ Product Owner manages the Product Backlog effectively

- ❑ DEVELOPMENT TEAM
  - ❑ Development Team size is between 3 and 9
  - ❑ All Team members interact closely every day
  - ❑ Development Team is self-organized
  - ❑ Development Team is cross-functional

- ❑ SCRUM MASTER
  - ❑ A Scrum Master is always available to the Team
  - ❑ As much as possible Scrum Master sits along with the Development Team
  - ❑ Scrum Master focuses on resolving impediments and help team to focus on the increment creation
  - ❑ Scrum Master creates a collaborative, transparent, and healthy work environment

- ❑ SPRINT PLANNING
  - ❑ Whole Scrum Team participates (including Product Owner and entire Dev team)
  - ❑ Product Owner comes to Sprint Planning with a prioritized Product Backlog
  - ❑ Product Owner and Development team must agree on “What” can be done and “How”

    A Sprint Backlog is created

- ❑ DEVELOPMENT TEAM HAS SPRINT BACKLOG
  - ❑ Owned by Development Team
  - ❑ Highly Visible
  - ❑ Gets updated on Daily basis at least (if not for every item when done)
  - ❑ No one tries to push items into Sprint Backlog without Development Team’s consent

- DAILY SCRUM HAPPENS DAILY
  - Whole Development Team attends on daily basis
  - Happens even the Scrum Master is absent
  - Team focuses on Planning and Knowledge sharing
  - Impediments and Problems are identified

- DEMO OF WORKING SOFTWARE EVERY SPRINT
  - Team demonstrates a working increment that meets the Definition of Done every Sprint
  - Feedback is received from Stakeholders and Product Owner updates that in Product Backlog
  - If you are working in a multi-team environment, the increment must meet the mutually agreed Definition of Done

- EFFECTIVE RETROSPECTIVES
  - Every Sprint should have a Retrospective at the end
  - Identify at least 2 concrete actionable improvements identified with respect to people, collaboration, communication, tools, practices, processes and so on
  - Previously identified action items are reviewed and implemented
  - Whole Scrum Team (including Product Owner) must present in Retrospective
  -

**ADDITIONAL (RECOMMENDED) PRACTICES**

- Team members are not locked into Specific roles
- Product Owner should have a clear and concise Product Vision and it is shared to stakeholders and also to the Development Team
- Everyone on the Development Team participates in the estimation, not just a few
- Product Owner is available when the Development Team is estimating
- The Estimation technique used should be mutually agreed by Scrum team (preferably size based estimation technique such as story points)
- Keep an impediments radiator in the team space
- Whole team should know what are the TOP 3 impediments
- Scrum Master should have specific strategies to resolve TOP impediments
- Development Team should resolve team level impediments without waiting for Scrum Master
- Scrum Master should be courageous to escalate global/systemic impediments to get management attention and support
  - Introduce Org level impediments board technique for this
- Sprint Backlog items are broken down into small tasks of less than a day for at least few items of Sprint Backlog
- Sprint Backlog items are re-estimated on daily basis

- When calculating Velocity consider only the items that are “Done”
- Use velocity only for release forecast but not to compare the teams
- Scrum Master focus on say-do ratio and encourage the team to discuss it in the Sprint Retrospective if the gap between say-do ratio is huge
- Development Team has a Sprint tracking mechanism (preferably Sprint Burndown) and it gets updated whenever an item meets Definition of Done
- Sprint Burndown should be highly visible
- Daily Scrum should be done same time same place
- Product Owner must be available at Daily Scrum if Development Team wants him/her at there
- Each Development Team member must know what other members in the team are working on
- Use Scrum of Scrums to effectively manage the dependencies across teams if you are using a scaled framework
- Ensure your team has fun in the work and they are highly energetic, more transparent and collaborative
- Overtime work is rare and it is done voluntarily but not on force by Product Owner or someone else like engineering manager
- Constructive disagreements are encouraged
- Development Team should experiment new things
- Core engineering Practices mentioned below are practiced
    - Test Driven Development
    - Pair Programming
    - Collective Code Ownership
    - Refactoring
    - Continuous integration
    - Automated refactoring, integration and performance tests

# APPENDIX – 7: SCRUM CASE STUDIES

**Below are few Case Studies of Scrum Implementation**

| TITLE | URL (Case sensitive) | QR Code |
|---|---|---|
| Intel: **How did they reduce their cycle time by 66% and addressed the schedule slippage?** | https://bit.ly/2FVKgLJ | |
| BBC: **How Scrum helped BBC to boost their effectiveness? (Video)** | https://bit.ly/2sRZbhy | |
| Dutch Railways: **How they implemented Distributed Scrum?** | https://bit.ly/2RRNIO5 | |
| Richard Banks: **How Richard Banks failed in their first attempt and how did they come back within 9 months?** | https://bit.ly/2RnTfqi | |
| Yahoo: **How did this giant company rolled out Agile in their organization?** | https://bit.ly/2SaAprs | |
| Tester in Scrum: **Detailed experiential report of a Tester worked in Scrum team** | https://bit.ly/2RU4zQ5 | |
| A Tale of Two Scrums: **Jeff Sutherland's personal interview one one example of Good Scrum implementation and another one that has failed.** | https://bit.ly/2HFDr2K | |

# APPENDIX – 8: AGILE & SCRUM RESOURCES

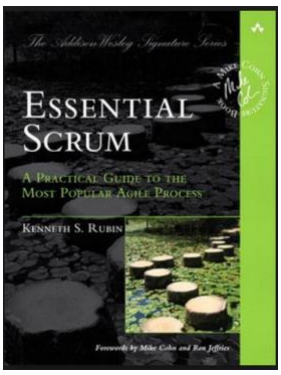
Below are some resources that help you to enhance your Agile and Scrum knowledge. Gaining knowledge through these resources will certainly help building your agile and scrum career further.

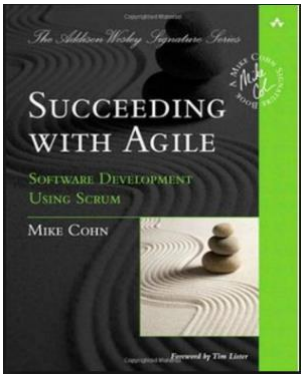
## Online Resources:

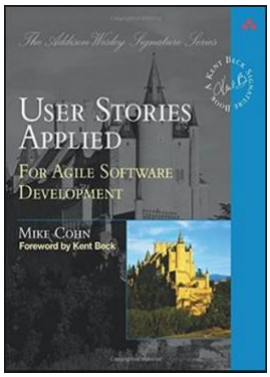
1. **CSM certification renewal process - https://www.scrumalliance.org/get-certified/renewing-certifications**
2. **Scrum Guide - http://www.scrumguides.org/**
3. **Scrum Alliance – http://www.scrumalliance.org/**
4. **My articles blog – https://www.learnovative.com/blog/**
5. **Linked in user Group – https://www.linkedin.com/groups/8305379**
6. **The New New Product Development Game (Available on Google Drive, link given below)**
7. **For different retrospective models:**
   - **www.funretrospectives.com**
   - **www.tastycupcakes.com**

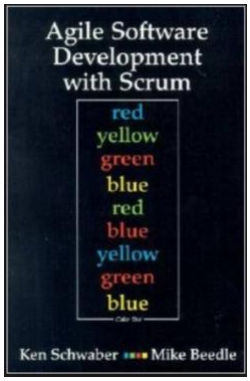
**Google Drive link for additional content: https://bit.ly/2QRnyKM**

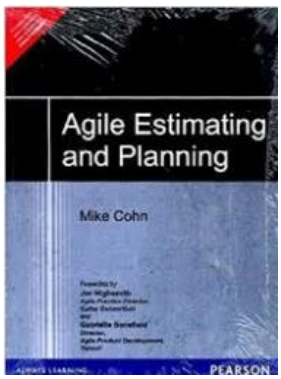
## Books: (www.pdfdrive.net is a free PDF repository, you can find loads of books there)

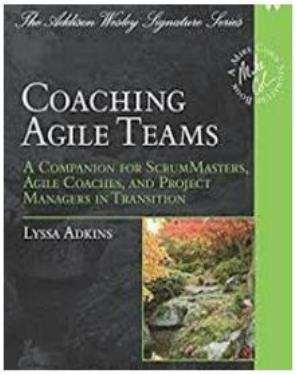


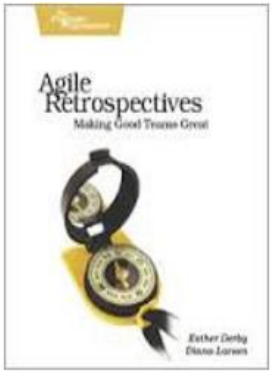



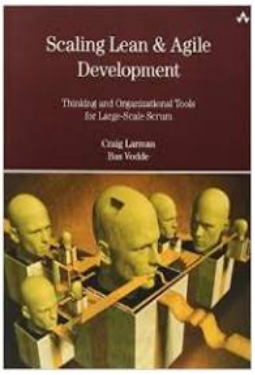



# APPENDIX – 9: GLOSSARY OF TERMS

| Term | Meaning |
|---|---|
| **Acceptance Criteria** | Tests which must pass for the Product Owner or customer to consider the Story accepted. The Team should verify these before submitting a story for final approval. Acceptance tests help ensure External Quality. Most Product Backlog Items can be mapped to one or more Acceptance Criteria. |
| **Agile** | A movement for finding better ways of developing software. More of a mindset/philosophy. Scrum and Extreme Programming are two leading examples. Others, such as Kanban or Lean Startup do not define themselves in the Agile tradition, but are based on compatible values and principles. |
| **Artifact** | Something that archaeologists find when digging. Often used to describe the documents produced by a project management methodology. Scrum artifacts are all living documents to guide and monitor work. In Personal Agility they are called "tools." |
| **Ceremony** | A different word for a meeting or routine process. In Scrum framework they are called "Events" to signify that something important happens and you want and need to be there. There are 5 events in Scrum: Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective. |
| **Chickens** | Deprecated term for people interested in the results of project, but not 100% committed to its success (e.g. due to conflicting priorities). Chickens can be very disruptive to the Team. Never call someone a chicken. Spectators is a better metaphor. The professional game is played for the spectators, but the spectators are not allowed to interfere with the game. |
| **Commitment** | One of the 5 values of Scrum which should not be interpreted to mean that the Team is expected to burn itself out trying to achieve unrealistic goals Sprint after Sprint after Sprint. |
| **Daily Scrum** | A daily opportunity for the team to inspect and adapt on their progress throughout the sprint. 3 defined questions to recognize that they need to talk to each other (preferably right after the Daily Scrum). |
| **Definition of Done** | An agreement (shared understanding) on what 'this backlog item is done' actually means. Helps assure Internal and External Quality for each Story. Often expressed as a checklist to be completed before submitting the Story to the P-O. The Definition of Done applies to individual Stories, not to Tasks or the overall release. |
| **Development Team** | An interdisciplinary team with all the skills necessary to get a problem from wish to done. |
| **Done (for a feature/user story)** | A binary state. Either a Story is completed according to the Definition of Done, or not. |
| **Done (for a product increment)** | A judgement call by the Product Owner. At the end of a Sprint, if the P-O believes that it's worthwhile to release, the product should be releasable. If it's not, there is undone work which should be addressed at the level of the Definition of Done in future sprints. |

| Term | Meaning |
|---|---|
| **Estimate** | A Team's best guess (prediction) at the size, complexity or effort involved to convert a PBI into a piece of finished functionality. An estimate is not a commitment. |
| **Extreme Programming** | An Agile approach to Software Development, often applied in conjunction with Scrum. XP **(XP)** defines the engineering practices needed to produce quality software in an iterative environment. |
| **Forecast** | A Team's best guess at how much finished functionality it can deliver by the end of a Sprint. The team is normally expected to respect all the terms of the Sprint Contract, i.e. Quality, Time and Cost, which are more important than Scope |
| **Impediment** | Anything which slows the team down or prevents someone from working. Although the Scrum Master is charged with removing impediments and all Scrum meetings provide regular opportunities to recognize them, impediments can be identified and eliminated at any time by anyone. |
| **Increment** | An additional slice of customer visible value, delivered by the end of the sprint. The latest increment must integrate with the previous delivered increments to form a working whole. |
| **Multitasking** | Pretending you can do more than one thing concurrently. If there is unused capacity available, multitasking can improve performance. However multitasking has a cost, and if there is no free capacity it lowers performance by introducing wait times and creating dependencies between otherwise independent processes. |
| **Order** | A unique ordering. First, Second, Third... The product backlog is sequenced. |
| **PBI** | Product Backlog Item. All the items of the Product Backlog are called as PBIs. They can be of type 'FRIEND's (Features, Requirements, Infrastructural, Enhancements, Non-functional, Defects) |
| **Pigs** | Deprecated term for those people 100% committed to the project at hand. Always refers to Scrum Master and Development Team. If it does not refer to the P-O, this is a sign of dysfunction. While it might be OK to call yourself a pig, "players on the field in a professional match" is a better metaphor. Yes the game is played for the spectators, and the spectators can have a surprising influence on the result, but the players must be able to play without undue interference. |
| **Priority** | Sequence—which item comes first, second, third, etc. The term priority is deprecated because a) it contains emotional overtones and b) two items could have the same priority, but must a have a unique place in line. |
| **Product Backlog** | The single source of requirements for the product under development. It consists of functional and non-functional **requirements**. It is not used to plan work or define intermediate artifacts, like a specification, which have no value for the customer or user. |
| **Product Backlog Item** | An entry in the Product Backlog, consisting of a description (often a user story), a sequence position, and an estimate. Often enriched with Acceptance Criteria and other useful **(PBI)** information. A PBI is not a specification, but rather a reminder to hold a conversation shortly before implementation. |

| Term | Meaning |
|---|---|
| **Product Owner** | Who guides the Development Team to produce customer visible value. Sometimes called the Voice of the Customer (or User), the role represents all interests outside the Development Team to the Team. |
| **Release Burn-down/up** | A tool for visualizing the progress of the team toward a medium term release goal. The y-axis is the sum of the estimates in the Product Backlog. When a PBI is Done, its estimate can be **Chart** deducted from the Burn Down chart. It is the primary tool for ensuring that wishes and probable reality stay reasonably aligned. |
| **Release Planning Meeting** | Team and Product Owner come together to refine the Product Backlog. Although time-boxed, there is no decision to be taken at the end of the meeting, so it is often a useful preparation for SP1. This is not core event of Scrum. |
| **Retrospective** | The Team (and anybody they invite) reflects on how they worked to identify improvements for the next Sprint. |
| **Ritual** | A different name for an Event in Scrum. |
| **Scrum** | A simple, team-based approach to solving complex problems. A mindset based on a culture of transparency and regular cycles of inspection and adaption. A popular approach for developing software. |
| **Scrum Master** | A servant leader who helps Product Owner and Development Team perform better. Coaches & Facilitates. Removes impediments. Sometimes called the voice of common sense. Helps organization to become better by helping to improve creativity, flexibility and productivity. |
| **Scrum Team** | All three roles together make up the Scrum Team. Sometimes called the Whole Team |
| **Spillover** | Work that has been started but not completed by the end of the sprint. Contrary to popular belief, spillover does **not** automatically carry over into next sprint. Excessive Spillover is typically a symptom of over-commitment in sprint planning and/or multitasking in the team. Technical Debt is a subtle form of spillover. |
| **Sprint** | A time-boxed period for completing work. A Sprint consists of planning, doing and review, both of the results and of how the Team worked. Maximum time-box is 30 days. 2 weeks is common. All forecast work should be Done by the end of the Sprint. |
| **Sprint Contract** | The agreement between Product Owner and Team at the beginning of a sprint: Time (Sprint Duration), Cost (Team Composition), Quality (Definition of Done) and Scope (Selected Product Backlog). If the team should fail to deliver on any aspect, it should fail on Scope. |
| **Sprint Planning** | Sprint planning addresses 2 questions: What and How. The meeting is divided in two halves, SP 1 and SP 2 for addressing these questions. While the Scrum Guide considers this to be one activity, many practitioners consider each half to be a separate meeting with its own time-box. |

| Term | Meaning |
|---|---|
| **SprintPlanning1 (SP1)** | The Product Owner and Development Team agree on what will be developed during this sprint. The PO defines priorities, the Team estimates how much is doable. So both parties influence the final agreement: the Forecast and the Sprint Goal. |
| **SprintPlanning2 (SP2)** | The Development Team decides how to solve the problem accepted in SP1. The result is a technical concept and a task planning, often in the form of a task board. |
| **Sprint Review** | The Team and Product Owner come together to inspect and adapt the product, based on Done functionality. They will review what has and has not been completed, and reflect on how to change the Product Backlog before the next sprint planning |
| **Story** | Term often used to refer to a Product Backlog item, even if not formulated as a User Story. Can also refer to a medium sized backlog item (on the scale of Epic >> Story >> Grain of Sand) |
| **Story Point (SP)** | A unit to gauge the size of a PBI relative to other PBIs, estimate the size of a project and monitor progress. |
| **Task** | The Team uses Tasks to plan the work in the Sprint. When all Tasks associated with a Story are completed, the Story should be Done. Typically a Task represents a goal for the day, or something smaller. Most coaches no longer recommend estimating tasks in hours. |
| **Task Board** | A visual representation of the work to be completed in the Sprint. Typically 4 columns, organized in swim lanes, per story: Story, Tasks Waiting, Tasks in Progress, Tasks Done. Often supplemented with Burn-down Charts, Impediments and other useful information. |
| **TDD (Test Driven Development)** | Also known as Red-Green-Refactor. 1) Write a failing unit test (red) 2) Code a first draft to turn the test green, keeping all other tests green). 3) "Refactor" to create an improved and final draft. TDD improves productivity by reducing misunderstood requirements, rework, and escaped errors. |
| **Team** | An older term for the Development Team. Because effective collaboration between P-O and Development Team is associated with high performance. In the latest Scrum Guide, the Product Owner, the Scrum Master and the Development Team together are referred to as the "Scrum Team." |
| **Technical Debt** | A consequence of poor engineering practices which make a program difficult to modify. Like financial debt, technical debt must be paid off or technical bankruptcy follows: Throw the program away and write a new one. |
| **Time-box** | A constraint to prevent a complex situation from degenerating into chaos. All rituals in Scrum are time-boxed. |
| **Undone Work** | Can you release the product at the end of the Sprint? If not, there is undone work. Typical examples include: regression testing, usability testing, customer acceptance tests. The less undone work you have, the more predictable your release dates. See Spillover. |
| **Unit Tests** | Automated tests written by the Development Team to assure Internal Quality. Unit tests enable Refactoring and provide an essential safety net, so that changes and fixes do not introduce new errors. |

| Term | Meaning |
|---|---|
| **User Story** | A people-centered approach to defining requirements with a standardized form: As <some role or persona> I want <some value> so that I can achieve <some goal or purpose>. The word 'user' should never appear in a User Story. |
| **Value** | A capability delivered to customer using which he/she can attain a tangible or intangible benefit. |
| **Velocity** | A unit to gauge the speed of development and estimate the completion date of large projects. Usually expressed as Story Points per Sprint. |
| **WAP** | Widely Adopted Practice, often used together with Scrum, but not part of Scrum—you may do it or not if you feel it applies to you. Examples include Story Points, User Stories, Definition of Ready. |
| **Waste** | Opposite of Value. Something customer does not want to pay for. Example: Features that do not meet definition of done is a waste. |
| **Work in Progress** | Work that has started but has not yet been completed. Lots of WIP is associated with poor **"WIP"** performance and inability to get things done. See "Spillover." |
| **Working Agreement** | An agreement among interested parties to enable more effective work. Working agreements are the basis for improvement in Scrum. |