Build Great Teams That Always Deliver Awesome Products

Certified Scrum Master (CSM®) Workshop

Version: 1.1 (Revised on 9th November 2018)

NAME:

Trainer

VIJAY BANDARU

Trainer Profile

Hello, welcome to my Certified Scrum Master ® workshop. My name is ***Vijay Bandaru,***I have 20 years of IT experience. I started my career as a Programmer in 1998 and played various roles in Technology and Leadership. I am the ***world's 1st*** Certified Team Coach [**CTC**] and also a Certified Scrum Trainer [**CST**] from Scrum Alliance.I have been into Project Management and Agile training since 2009. I believe in gaining knowledge from training is the primary goal and the certification is the byproduct. So, I will give you enough confidence to you to start your Agile and Scrum journey in this workshop. Of course, you will also be prepared to achieve the CSM ® certification.

My Vision is to: Make Learning "SPECIAL"

Simple,Practical, Effective, Collaborative, Innovative, Adaptive, Long lasting

Few more details about myself:

- ✓ Founder of Learnovative (www.learnovative.com)
- ✓ Certified "Training from the Back of the Room" (TBR)
- ✓ Certified Agile coach
- ✓ Has hands on experience in 3 Scrum roles: Product Owner, Scrum Master & Developer
- ✓ Has organizational transformation experience
- ✓ Speaker in several regional and international Agile conferences
- ✓ Received "Best Trainer" award multiple times
- ✓ Achieved 100% Net Promoter Score so far
- ✓ Coached more than 60 teams and close to 75 Scrum Masters
- ✓ Published several articles on Agile and Scrum
- ✓ Created WONDER Coaching model
- ✓ Active knowledge sharing through Linked in Group "Universal Agile & Scrum Community of Practice"

You can find **my previous Trainings Videos, Photos and Feedback** at: **www.facebook.com/learnovative**

You can also view more details at my Personal website:**www.vijaybandaru.com**

In case you would like to contact me directly, you are most welcome, my details are below:

Email: vkbandaru@yahoo.com **Phone: +91-9848032144**

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

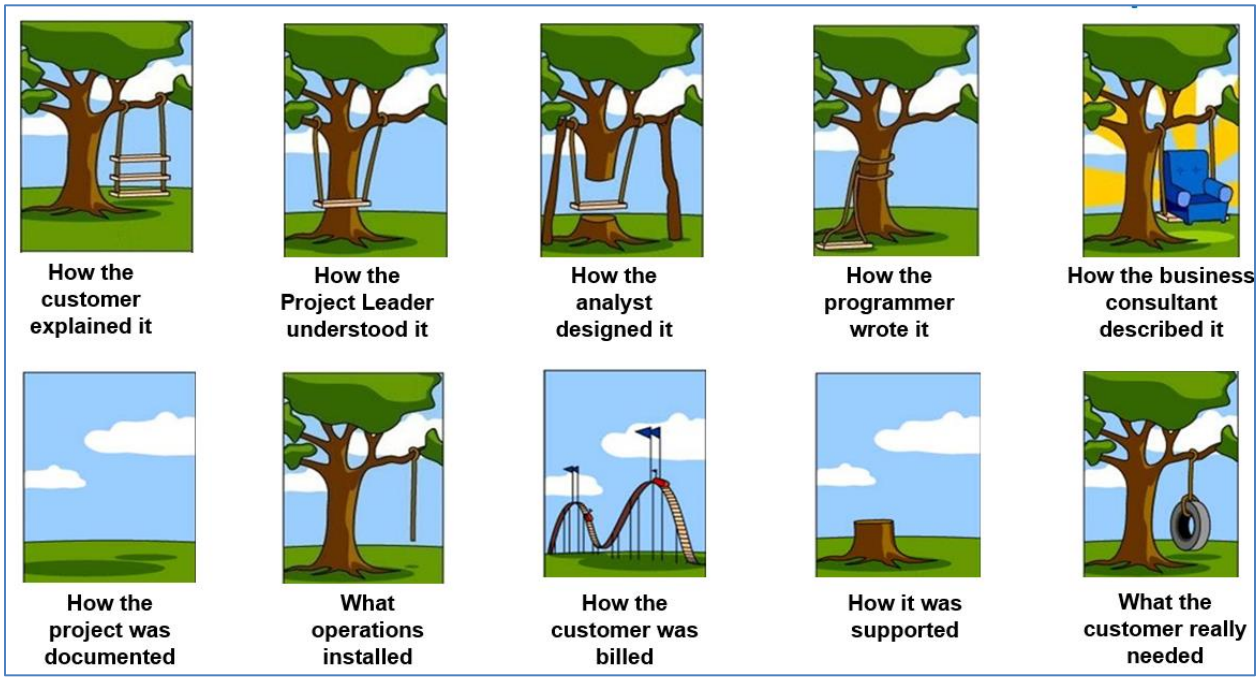
Hope you would enjoy the training and enrich your knowledge. All the best and Be Agile!

TABLE OF CONTENTS

AGILE

INTRODUCTION

The Classic Problem



Based on the above scenario, answer the following questions:

1. **When the value was delivered to the customer?**

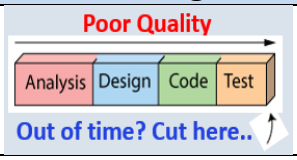
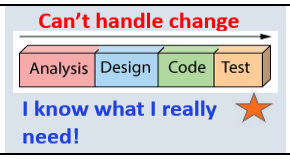
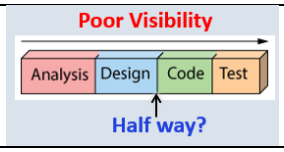
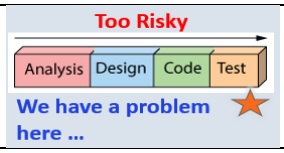
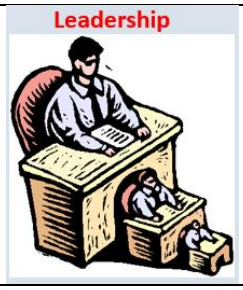
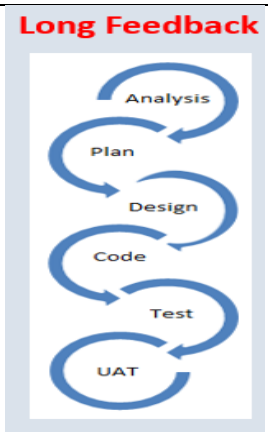
2. **Who is at fault in the above project? Customer or Delivery Team? Whey they are at fault?**

3. **How was the execution of the project?**

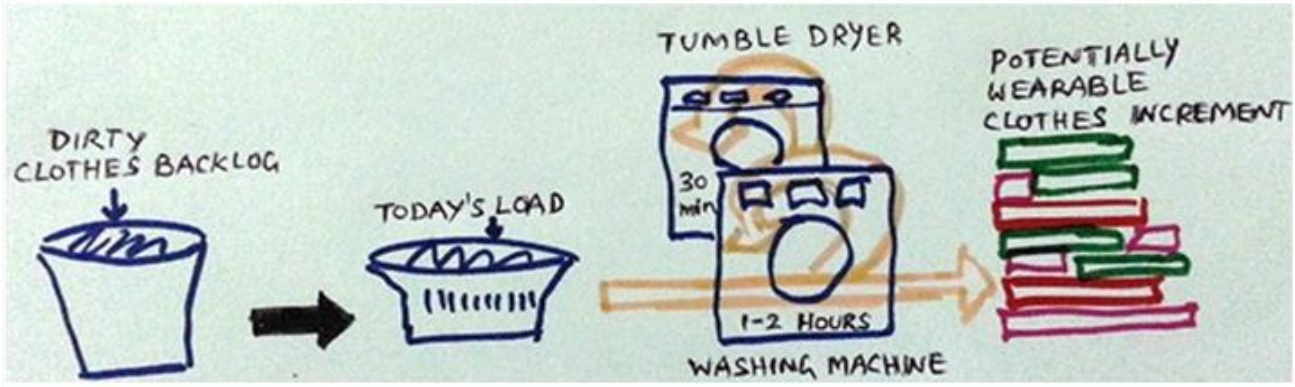
4. **How was the communication in the project?**

5. **If you want to do the same project again and address the above issues, how do you do it differently?**

Waterfall Challenges:

Challenge	Description
	-
	-
	-
	-
	-
	-
	-

Agile Introduction

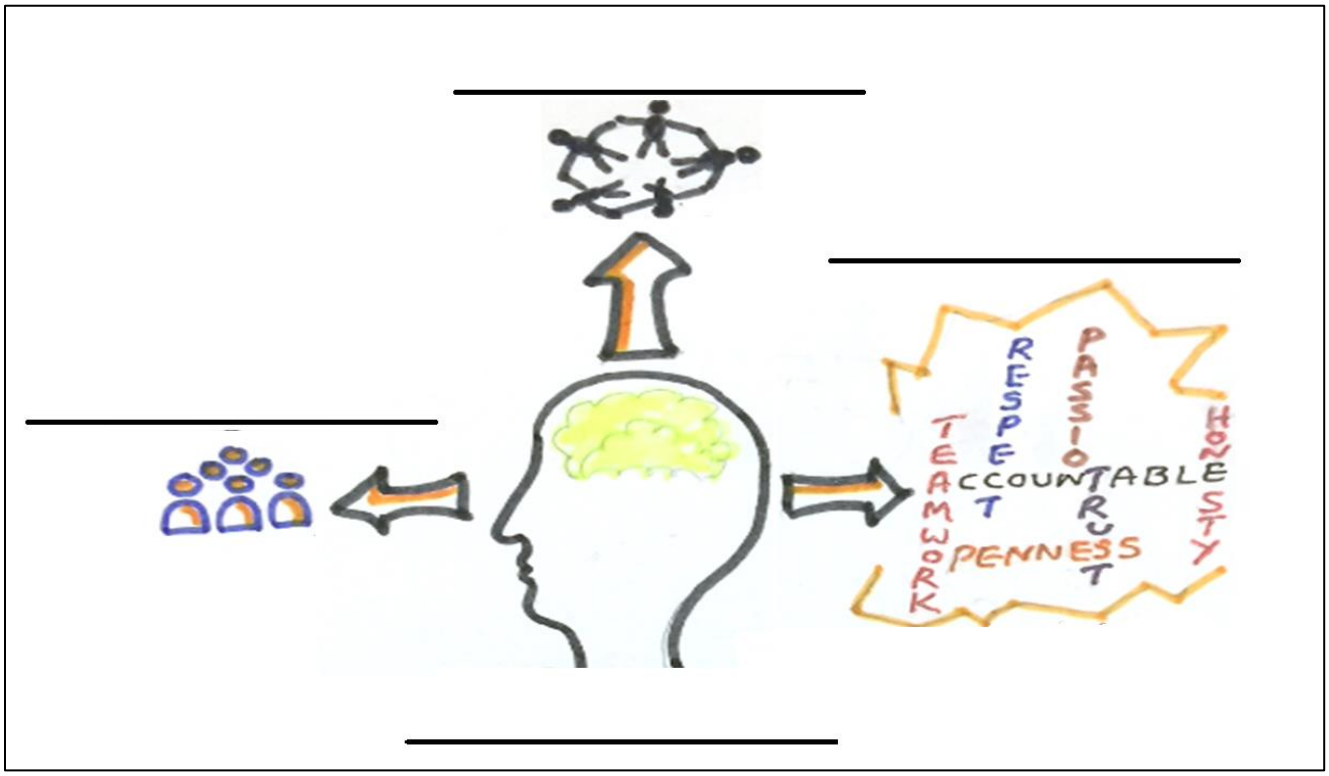


Which of the above options is good? And Why?

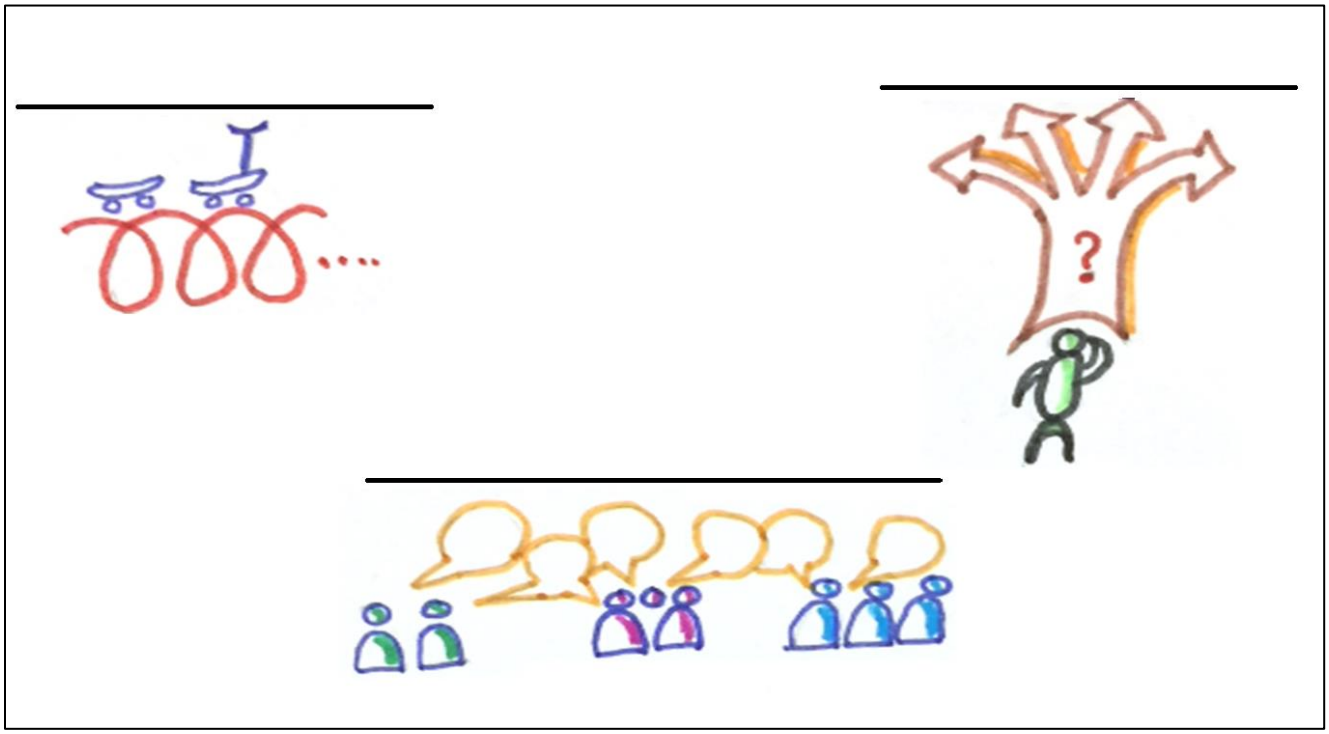
-
-
-
-
-
-
-

If you agree that the bottom option is good, then can you draw a similar approach for a Software Development project/Product?

What is Agile?



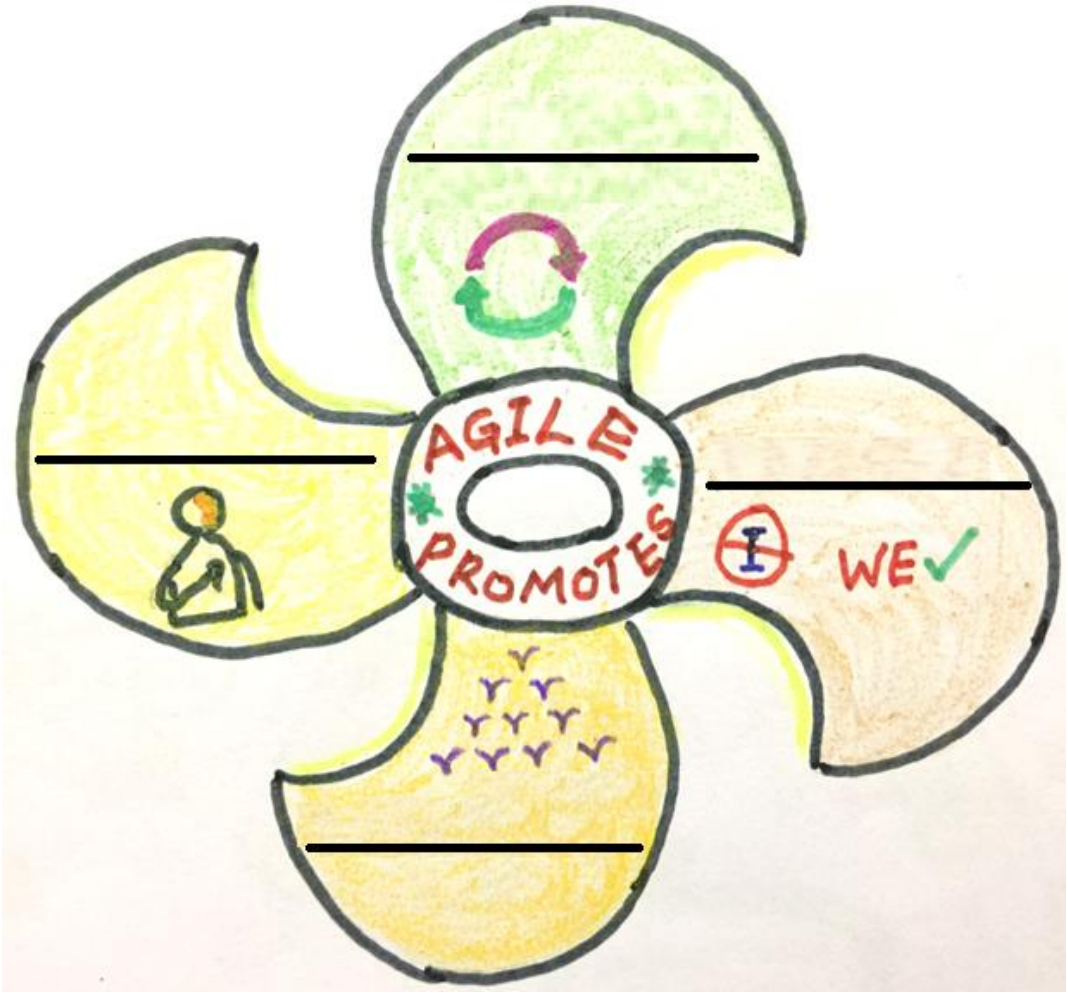
Agile Primarily focuses on:



- **Agile is a philosophy/mind-set** that uses organizational models based on people, collaboration and shared values
- The phrase "Agile" was coined in February 2001 at a summit held in Snowbird, Utah.
- *Agile uses:*
  - ***Iterative and incremental delivery:*** Process is used in a repetitive way and produce add-on deliverables (requirements in a working software model)
  - **Rapid and flexible response to change:** It emphasizes the importance of accepting changes and take them into consideration even late in the software development
  - ***Open communication between teams, stakeholders and customers:*** Agile encourages more direct communication among all stakeholders rather than written (documented or email). Agile promotes "Talk more…write less (only what is required and important)

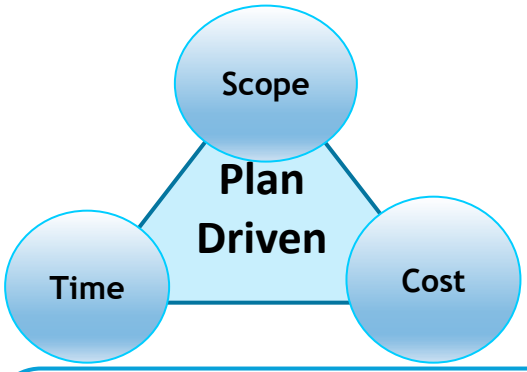
SPACE FOR NOTES:

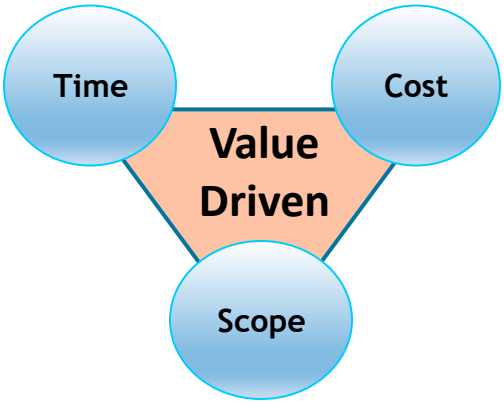
Agile generally promotes:



SPACE FOR NOTES:

Plan Driven& Value Driven:





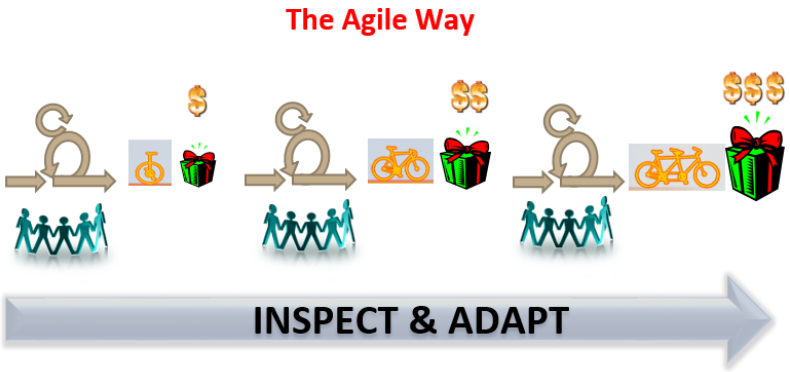
Traditional

Features/Functions are fixed in the Requirements Spec at an early point, Time/Cost estimated/quoted, and are then delivered within this Requirements Spec constraint

Agile

Time/Cost are fixed at an early point and the highest priority Features/Functions are then delivered within this Time/Cost constraint

Value realization:



SPACE FOR NOTES:

What is Agile and what is NOT Agile? (Tick for what is Agile and cross for Not Agile)

Description	Tick/Cross
A disciplined and structural approach to iterative development	
Getting the same amount of work in less time and cost	
A set of practices that accommodate changing requirements	
A way for the business to change it's direction on the fly	
A set of practices that will help improving the quality	
A project management methodology	
An umbrella of different iterative and incremental software development frameworks	
Agile is a Process	

SPACE FOR NOTES:

Agile Manifesto:

On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch Mountains of Utah, 17 people met to talk, ski, relax, and try to find common ground. The Agile "Software Development" Manifesto was emerged from that meeting. Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened.

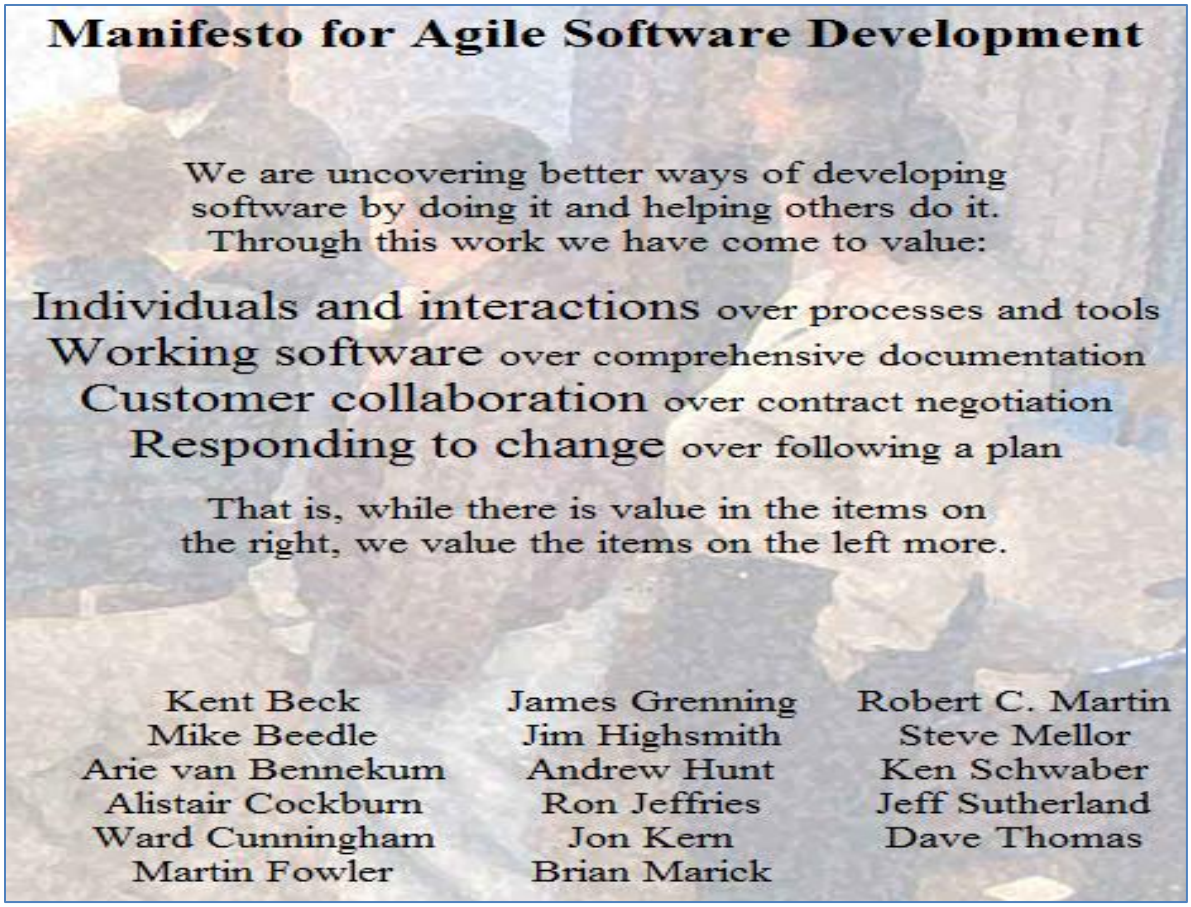
“A Manifesto for Agile Software Development" was emerged and signed by all participants. The term agile came from Martin Fowler. The group was named as "The Agile Alliance"

Exercise –Agile Manifesto

You see 8 different words listed below. Try to map them to the respective blanks on the boxes given below them.

Plan	***Tools***	***Interactions***	***Collaboration***
Software	***Documentation***	***Change***	***Negotiation***

Individuals and _____	**O**	**Processes and _____**
Working _____	**V**	**Comprehensive _____**
Customer_____	**E**	**Contract _____**
Responding to _____	**R**	**Following a _____**



SPACE FOR NOTES:

Agile Manifesto Value Statement	Description
Individuals & Interactions Over Processes & Tools	- Suppose you have technically very strong team but they do not work as a team. They have egos and reservations. But you have great processes and tools. Will it work? - In reverse you have technically a mix of excellent and good but with a good team culture and help each other and you have limited tools available. Will it work? - Assume that you have an excellent documentation process in place and you received a requirements specification document. Will it guarantee that the requirement is understood properly, and you can convert it into a working functionality? - Individuals are the team members and the product or customer and anyone who is involved in the product - Interaction is in the sense of collaboration, support, team spirit, knowledge sharing etc. and with a continuous improvement goal - They are self-organized. That means they will not wait for someone to tell them what to do, when to do and how to do. They make their own decisions to reach their goal - Success comes through the project team but not by the processes or tools - Individuals are the ultimate source of value generation to the customers! - People design and build software products
Working software over Comprehensive documentation	- Assume that you have a complete set of requirements defined upfront and in very detailed. Will that add any value to the customer? - It can't guarantee that requirements will not change - I do not know how many of those add value and how many doesn't - Assume that you have a great and in depth technical high level and low-level design documentation that covers the entire

	product design and architecture and the database design in place. Does it help the customer to get some revenue? - In contrast, you have discussed 2 requirements with customer that add highest value and created only required (JIT) documentation and built those features and delivered to customer as working software. Does it any better than the previous scenarios? - It helps to get your early feedback - Keep the cost of change low - Show progress to the customer and make him happy - "I know when I see it" this is true in most of the cases. So, customers will know better what they want by seeing a working software. But not a well-defined documentation. - Focus of the team is to deliver valuable software to customers - Quality driven through Validated learning - Customer satisfaction is given highest importance
Customer collaboration over Contract negotiation	- You have a clearly defined scope and a well-defined contract (agreement) between the customer and you. Does it help the customer to have flexibility of changing the scope in between? - You will not allow - Even you allow you charge him a lot - Who is paying for the product? Customer - Does it make sense to stop him changing? - Suppose you have customer working with you as part of the team and provide you the requirements face to face and prioritize them to be delivered as working software. Does it have any advantages over the previous scenario? - So, it says that the development team and the customer collaborate in such a way that together they deliver a successful valuable product. - Both should work hand in hand without

	creating walls in the format of contract. Of course, contract is required but not for in this direction. It is more on the payment terms and other points while highlighting the flexibility and collaboration. - Collaborate, and deliver the product as per the customer vision - Maintain healthy and cooperative relationships
Responding to change over Following a plan	- Does this value statement say there is no plan required? Was planning not there earlier? - Planning is everything and plans are nothing - Assume that you have a well-defined start to end plan in Microsoft project. Does it guarantee that there will be no change? No! - Suppose you have a change received and how quickly you check and what is the impact on your overall plan? It takes lot of time to analyse - Instead, if you plan for a short duration with what you know and keep flexibility of changing it based on what customer wants, will it add value? - This is called rolling wave plan. Means plan the work that you know very well for a short duration and keep the others at a very high-level plan that might change - When we say change, this means any change that adds value over what you are currently working on. - World is very Dynamic People change their priorities very often based on their tastes and marketing fluctuations - Change is a fact and inevitable. Honor changes as and they come - Inspect, Adapt, and Improve; continuously rather than sticking to a concrete plan

SPACE FOR NOTES:

Agile Principles

Exercise –12 Agile Principles

1. Our highest _____ through early and continuous delivery of valuable software
2. Welcome _____in development. Agile processes harness change for the customer's competitive advantage
3. Deliver _____ to a couple of months, with a preference to the shorter timescale
4. Business people and developers _____ throughout the project
5. Build projects around motivated individuals. _____, and trust them to get the job done
6. The most efficient and effective _____is face-to-face conversation
7. Working _____ measure of progress
8. Agile processes promote _____. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to _____ enhances agility
10. Simplicity--the _____ not done--is essential
11. The best architectures, requirements, and designs _____ teams
12. At regular intervals, the team reflects on _____ and adjusts its behavior accordingly

Phrases:

A. working software frequently, from a couple of weeks

B. Give them the environment and support they need

C. must work together daily

D. priority is to satisfy the customer

E. software is the primary

F. sustainable development

G. method of conveying information to and within a development team

H. changing requirements, even late

I. art of maximizing the amount of work

J. technical excellence and good design

K. how to become more effective, then tunes

L. emerge from self-organizing

Agile Manifesto 12 Principles:

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

   Agile development differs from traditional development in that rather than a "big bang" delivery after months or even years of work, subsets of functionality are delivered within weeks of the start of the project. Software is delivery is incremental rather than in a single lump. Early, continuous delivery of useful software promotes customer confidence and satisfaction.

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

   The whole point of delivery should be meeting the customer's needs, thereby giving them a competitive edge. Rather than blocking necessary changes to the product as requirements change, agile software development is designed to cope with evolving requirements.

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

   Again, agile development emphasizes fast, iterative deliveries rather than the "big bang" delivery of software after a lengthy development process.

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

   Rather than working FOR a customer, agile development encourages development teams to work WITH customers, daily. In this way a better working relationship can be established, problems can be spotted and corrected more quickly, and customers can see progress daily.

   Instead of focusing on a waterfall Gantt chart and controlling impossible things, we focus on communicating with the customer and setting realistic expectations which will enhance transparency and create trust.

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

   Agile software development recognizes the importance of the team, the working environment, and stresses the need for support and trust in the team as critical for the projects - and the software's - success. Traditional development methodologies stress processes, tools, plans and rules.

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

   While documentation, status reports and memos are necessary evils in most businesses, discussing a project in person can eliminate many misunderstandings and omissions. Agile encourages on “Talk more .. write less”. So, discuss the requirements and other important points face-to-face and then only create required documentation so that it will make things easy and fast.

7. **Working software is the primary measure of progress.**

   Traditional development and project methodologies often put too much emphasis on milestones which have little, if anything, to do with the actual progress of the project. In agile software development, working software is THE measure of progress and success.

   Also, if your traditional project has 4 phases, completion of two phases does not guarantee that 50% of your project is complete. Instead, if you have 100 requirements and out of them 50 requirements are delivered to customer in “working software” model, this confirms 50% of the project completion.

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

   Agile development proceeds at a more consistent pace, rather than suffering from lulls while waiting for decisions or approvals, followed by a flurry of activity to meet arbitrary deadlines. Sustainable pace helps the product owner to have better prediction on the future work so that he can plan releases effectively.

   It is just like Sprinter Vs Marathon runner. Sprinter can run only a certain distance with high speed, but marathon runner can run for a very long distance by adjusting this speed keeping the goal and the remaining time in mind. So agile teams should be like marathon runners.

9. **Continuous attention to technical excellence and good design enhances agility.**

   While agile methods improve customer satisfaction and the quality of the product, striving for excellence technically also promotes development "agility". This means a focus on the just in time design, engineering practices, loosely coupled components etc. will help improving the agility in terms of design and architecture.

10. **Simplicity--the art of maximizing the amount of work not done--is essential.**

    "Simplicity is the ultimate sophistication “. Simplicity is focusing on the product’s essence, building only what is really needed, and being able to adjust and extend the product easily.

80% of the value comes from only 20% of the features. So agile promotes delivering the top 20% high value requirements early in the cycle.

Too many traditionally designed software products are complex, not only because of cumbersome requirements, but because of cumbersome design, development and project methodologies. Stressing simplicity in all aspects of development will help to ensure a simple, workable, more easily maintained solution.

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

    Artificially organized teams - when a team's structure is fixed by upper management with little knowledge of the requirements, customer or the personalities of the team itself - can create situations where personnel are under-utilized. In the worst case, projects fail simply because of poorly organized teams. In an agile development environment, the team is empowered to organize themselves in a way which allows them to be most effective and efficient.

    Instead of treating people like resources (did someone say, 'Cattle'), we treat them as trusted team mates.

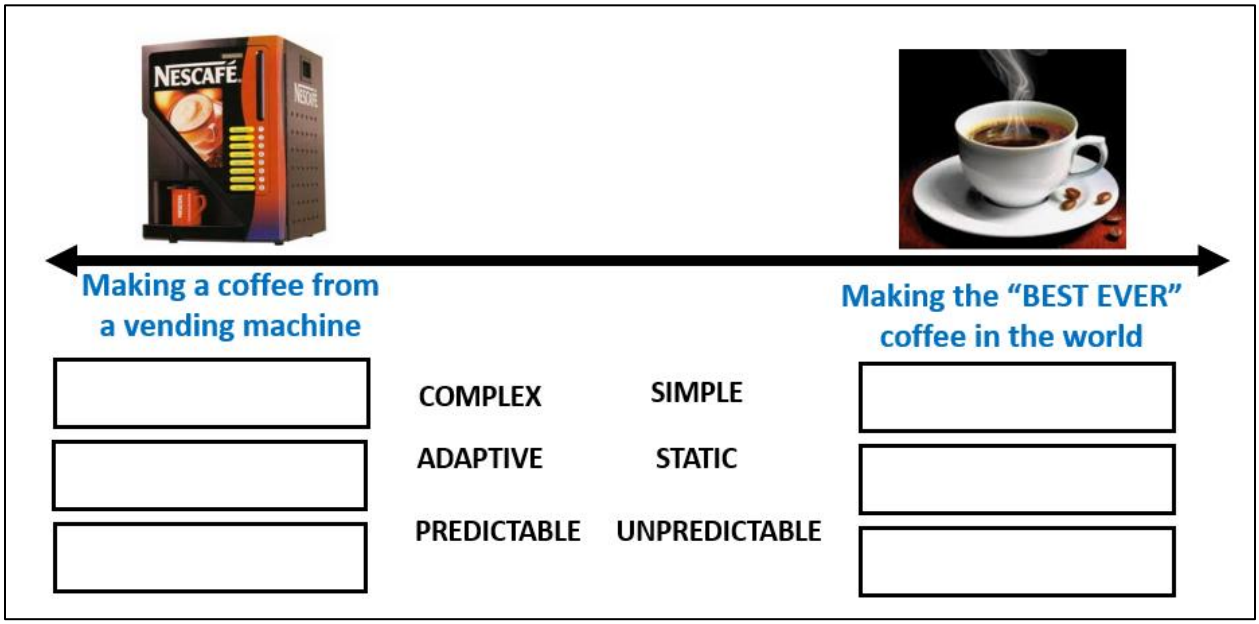
12. **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.**

    Rather than waiting for the end of the project to conduct a post-project review, the agile team is constantly reviewing their progress and processes to ensure they are working efficiently and effectively.

    Agile methodologies may not solve all the problems associated with software development, but the principles laid out in the manifesto offer a common-sense approach to resolving some of the reoccurring issues experienced by many teams, customers and projects. By adopting just a few of these guidelines, software development can be simplified and more successful in the primary goal - delivering useful, cost effective software.

SPACE FOR NOTES:

Defined Vs. Empirical Process:



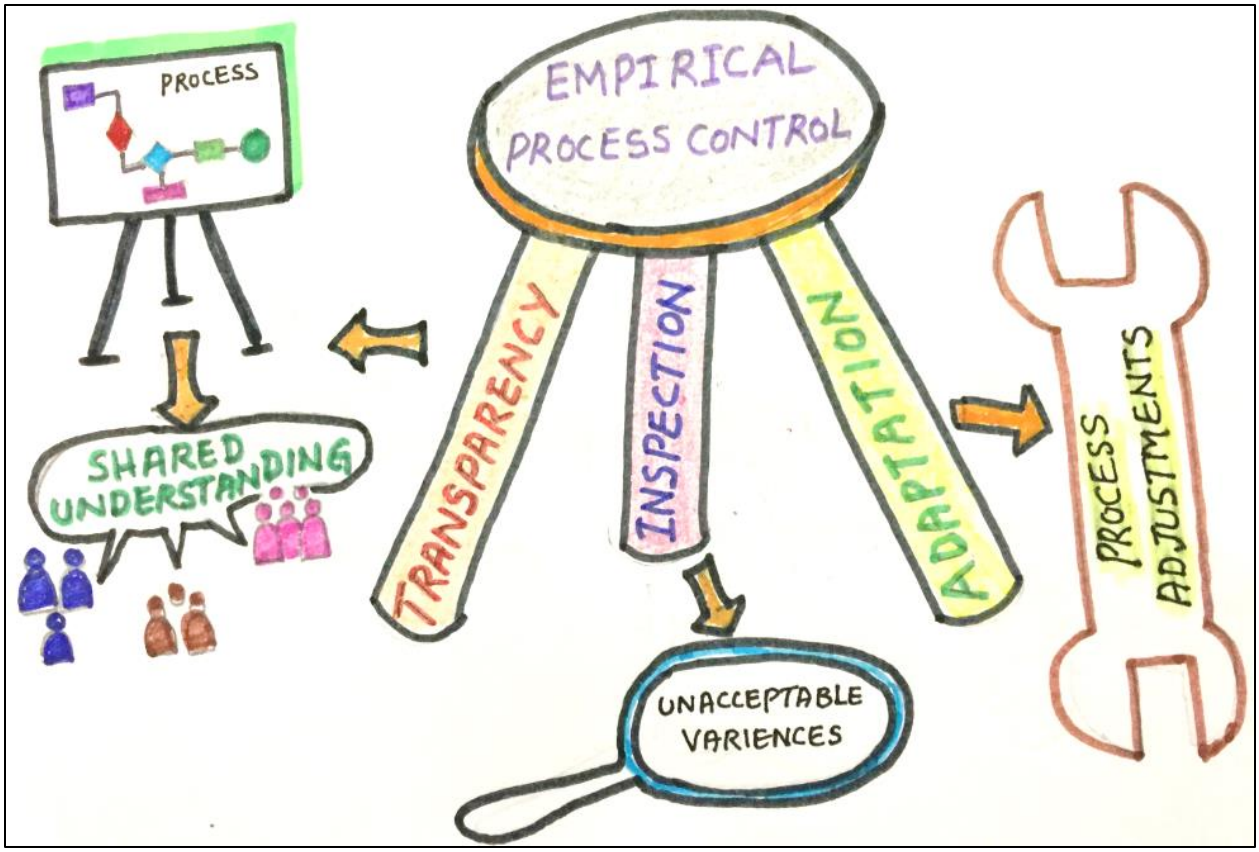
Considering the two examples mentioned in the above diagram about making coffee, map the 6 characteristics of those problems 3 on left and 3 on right.

"Simple" problem is something that is clear, inputs and process steps are clear and output is predictable and same always. So a "Defined" process can be used to address "Simple" problems. You can have a plan using which you can try to solve the Simple problem.

"Complex" problem is something that has unknown is more than known. So "Defined" approach cannot be applied to solve "Complex" problems. You need a different approach to solve them, that is "Empirical Process".

Agile software development is "Complex" environment as there is more unknown than known. So it cannot be effectively done using "Defined" process, you need an empirical (experimental) process. Empirical process is experimental way of learning through experience (i.e knowledge comes through experience), making decisions based on what is known with frequent inspection and adaption. ***Transparency, Inspection and Adaption*** are 3 legs of empirical process as shown below.

SPACE FOR NOTES:



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).

Being agile and adaptive means that the entire project team - managers, development teams, project leaders, etc. -must have some idea of where they are going, and a mindset that adjustments to that plan (adaptation) are normal.

SPACE FOR NOTES:

Exercise: Identify what type of processes the below are

PROCESS	EMPIRICAL PROCESS	DEFINED PROCESS
1. Bringing up children	☐	☐
2. Product development	☐	☐
3. Baking	☐	☐
4. Mobile App creation	☐	☐
5. Making a movie	☐	☐
6. Attempt a competitive exam	☐	☐
7. Cooking	☐	☐
8. Car driving	☐	☐
9. Trekking trip	☐	☐

SPACE FOR NOTES:

SCRUM

INTRODUCTION

Scrum - Introduction

Jeff Sutherland and Ken Schwaber developed Scrum and they wrote the Scrum Guide. It was developed based on "The New New Product Development Game" document submitted at Harvard business review in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka.

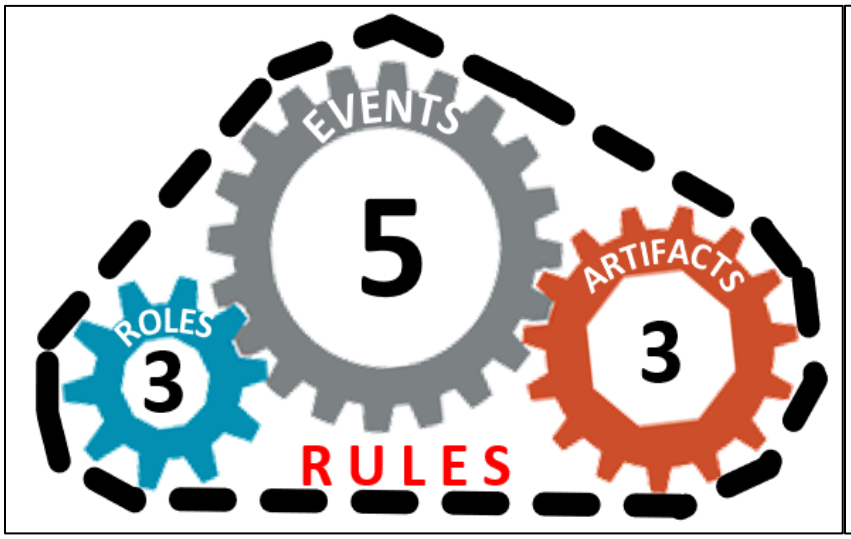
Scrumis a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. Scrum is a framework for developing, delivering and sustaining complex products.

- ✓ Scrum is ***not*** a process or technique
- ✓ Scrum is ***not*** a methodology

People can use various processes, practices and techniques that can fit in the Scrum framework to develop the products.

Scrum consists of: Roles, Events, Artifacts and Rules that bind them together. Each component within the framework has a specific purpose and is essential to Scrum's success and usage.

Scrum framework:



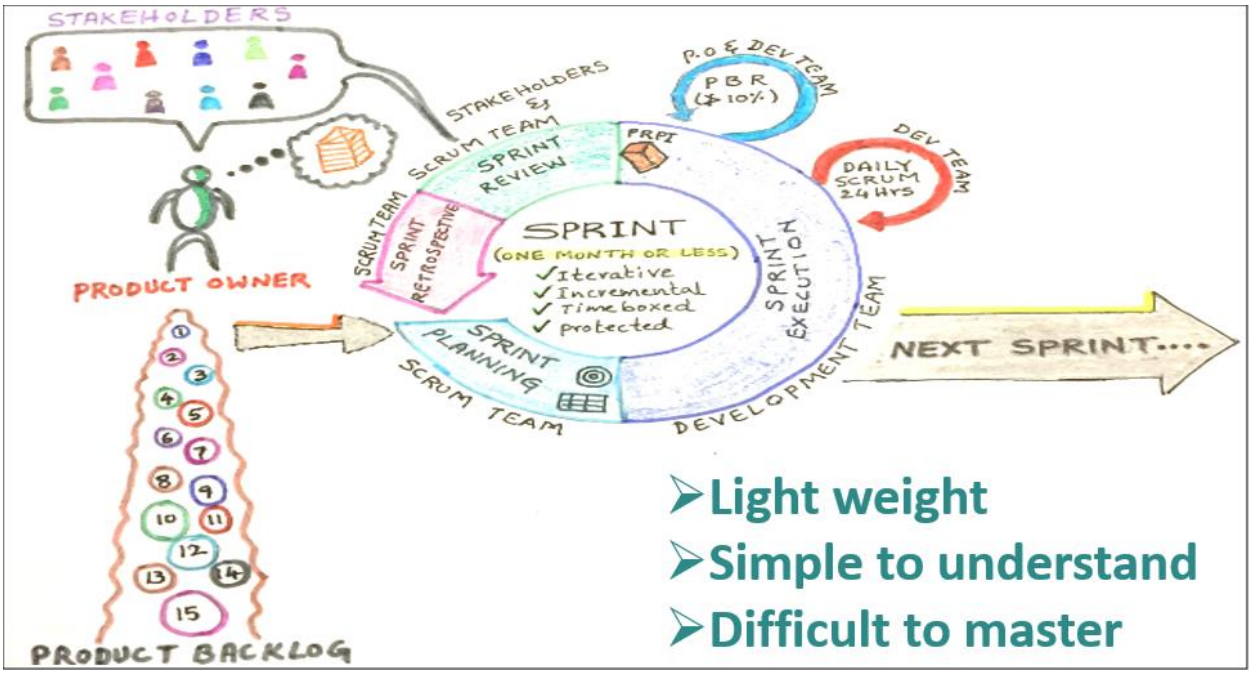
Sprint Planning
Product Owner
Sprint
Scrum Master
Product Backlog
Development Team
Sprint Review
Product Increment
Daily Scrum
Sprint Backlog
Sprint Retrospective

SPACE FOR NOTES:

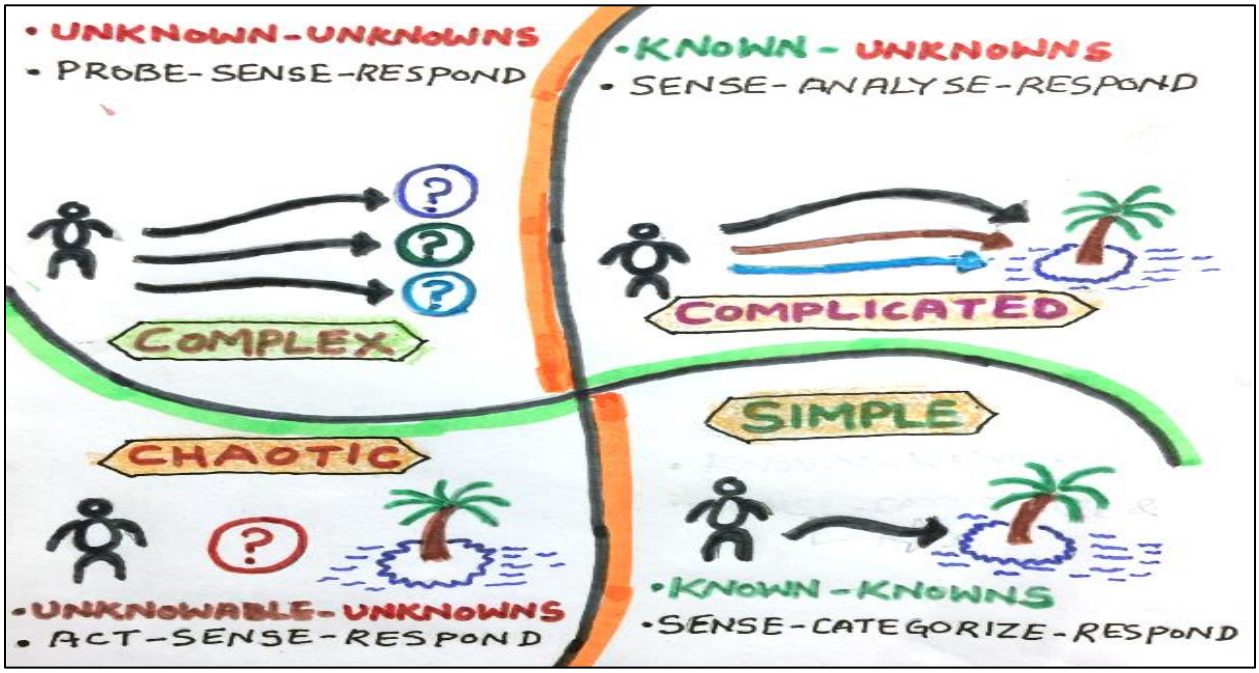
Is Scrum Agile?

Yes, Scrum is Agile but Agile is not ONLY Scrum. Agile is an umbrella of methods as shown below.

Overview of Scrum Framework



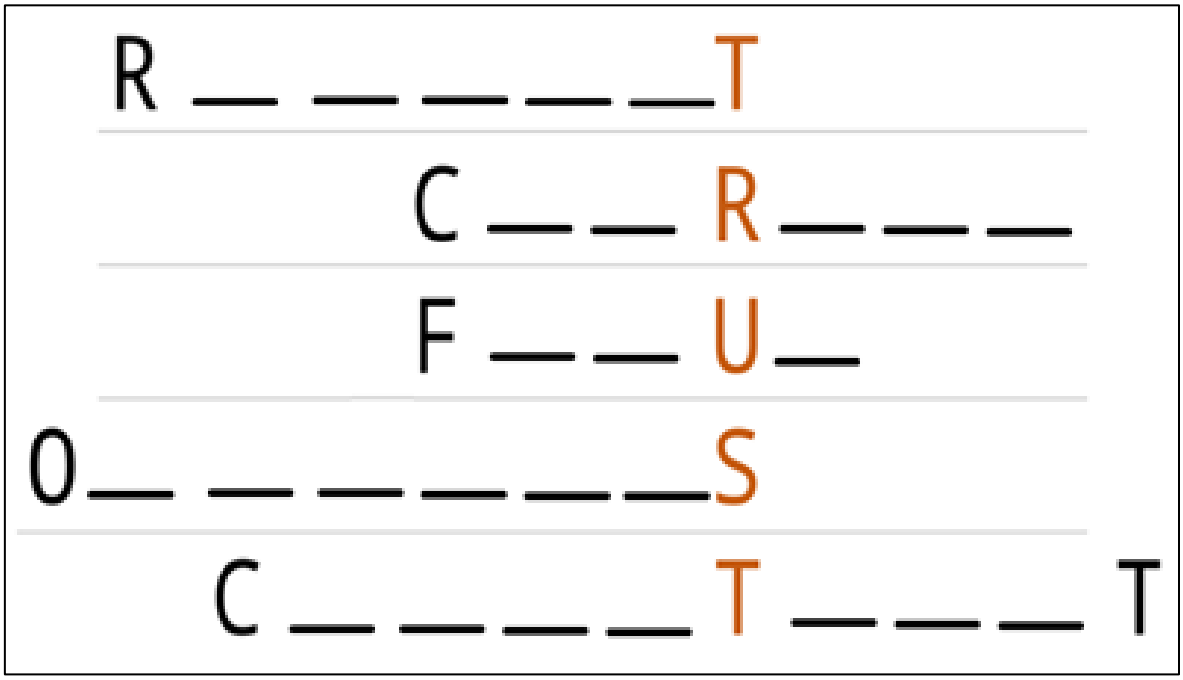
Where can you use Scrum?



Scrum works effectively in Complex environment. The reason being, continuous inspect and adaption and validated learning helps to deliver the value faster. Scrum can also be used in other environments, but it is best suited for "Complex" environment.

SPACE FOR NOTES:

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**.*

Focus: Scrum team should focus current work, on quality, delivering releasable increment, one thing at a time, continuous improvement, teamwork, collaboration, value delivery, eliminate waste, outcome rather than output and so on

Openness: They need to maintain transparency across all levels and keep the artifacts visible and accessible to all stakeholders.

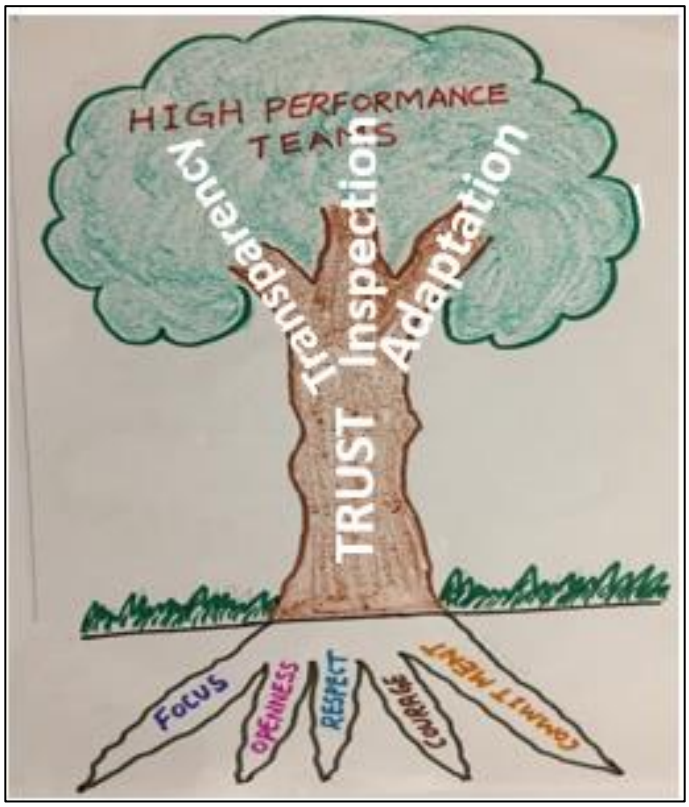
Respect: They need to respect each other in the team, respect their definition of done, respect the practices that improve their delivery.

Commitment: They must show commitment to give their best to meet the sprint goal within the timebox. Committed for team success and not individual success.

Courage: They should have courage to:

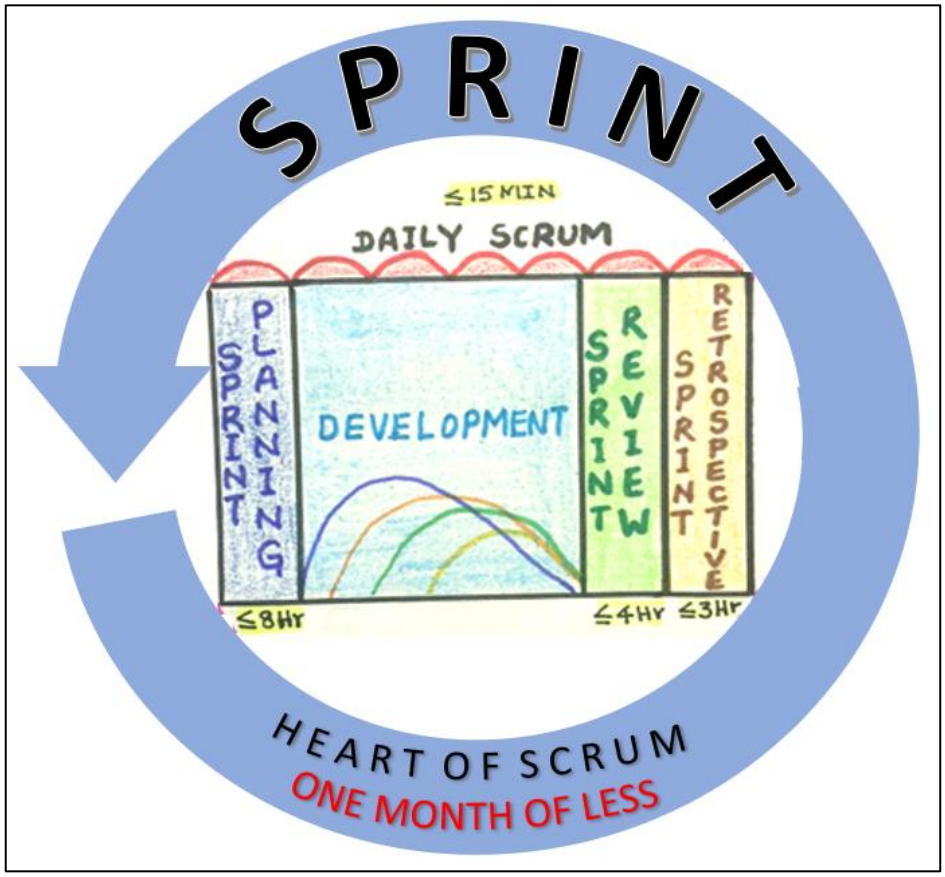
- Make decisions on their own

- Do things differently
- Say “No” when needed.



SPACE FOR NOTES:

Sprint – The "Heart" of the Scrum



Note: Durations mentioned above for the events are based on one-month sprint. However, except the daily scrum, remaining three events' duration will vary based on sprint length (duration).

Product Backlog Refinement is an ongoing activity and done during the sprint. Product Owner leads it and Development team supports. They will decide how and when they want to conduct it. It can take not more than 10% of the Development team's capacity in the Sprint

SPACE FOR NOTES:

Exercise: Understand the Sprint based on the table below

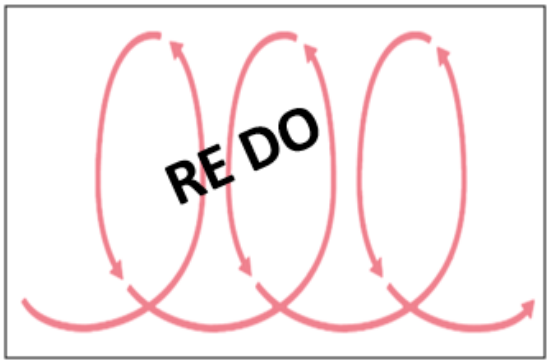
Fill the below blanks with the numbers of the phrase on the second column	
A basic unit of ---------- is called as "Sprint"	1 heart
Team produces --------- at the end of every Sprint	2 changes
Each sprint will have specific -------	3 needs to be built
Each Sprint is ---------- (i.e. Can only take allowed max duration, 1 to 4 weeks)	4 fixed
No ------- between sprints	5 changed during
Sprint is the ---------- of Scrum	6 Potentially releasable product increment
All the events of the ---------- are timeboxed	7 development in Scrum
Each sprint has a definition of what --------, a design and a plan	8 goal are accepted
A new sprint ---------- after previous sprint	9 gap
The duration of sprint is ----------	10 time boxed
No changes that impact the sprint ---------- during the sprint	11 predictability
Same duration for all sprints gives better ----------	12 sprint
Quality goals cannot be --------- the sprint	13 events
No ---------- in team composition	14 starts immediately

Note: Answers are available at the end in Appendix

SPACE FOR NOTES:

Sprint Characteristics:

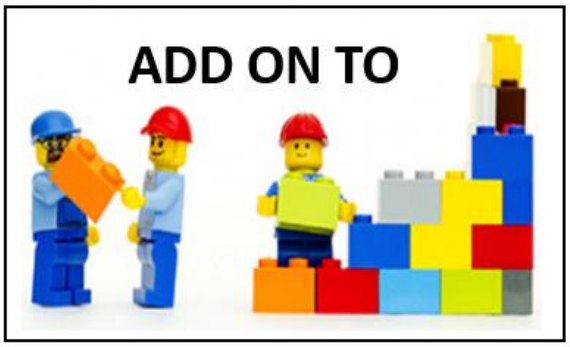
Iterative:



"Iterative" means that we build a partial version of a product and then expand that version through successive short time periods of development followed by reviews and adaptations. That is, making progress through successive refinement!

- *Iteration refers to the cyclic nature of a process in which activities are repeated in a structured manner; until the result is "complete".*

Incremental:



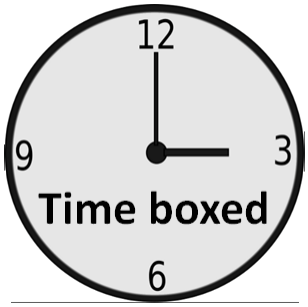
"Incremental" refers to the quantifiable outcome in every timeboxed duration (such as Sprint). An incremental process is one in which software is built and delivered in pieces. Each piece, or increment, represents a complete subset of functionality. The increment may be either small or large. Incremental way adds value incrementally.

SPACE FOR NOTES:

Why iterative & incremental?

- Reduced risk
- Flexibility in managing changes
- Manageable complexity
- Predictable pace of delivery
- Clear visibility of working increments
- Better quality
- Constant collaboration with customers
- Frequent feedback
- Useable product increment
- High value delivered early
- Eliminate waste

Timeboxed:



Timeboxing is allotting a fixed, maximum unit of time for an activity. That unit of time is called a time box. The goal of timeboxing is to define and limit the amount of time dedicated to an activity.

In Scrum, timeboxing is a critical component of all five events. Sprint duration is timeboxed, will not change. All the events of sprint are also timeboxed and vary according to the sprint duration (except the daily scrum).

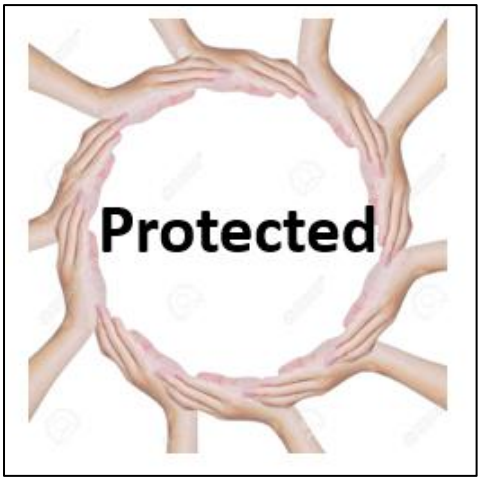
- Why do we have some things (Example: Competitive exam) are time bound?
- Why not we allow everything to take its own time?

Parkinson's law (work expands the time available) and focus is lost if there is no time boxing.

- **Why should we prefer to shorter time durations for Sprints?**

Shorter duration sprints provide early feedback.

Protected:



No changes are made that would affect the Sprint Goal. Development Team composition remains constant - the Team members can change only between sprints, not during sprint. Quality goals do not decrease. Scope may be clarified and re-negotiated between the Product Owner and Development Team as more is learned. Product backlog refinement/grooming meeting will be conducted (preferably in the middle of the sprint)

Sprint Cancellation

A Sprint can be cancelled before the Sprint time-box is over. ***Only the Product Owner has the authority to cancel the Sprint***, PO can take inputs from the stakeholders, the Development

Team, or the Scrum Master. A sprint can be cancelled if the Sprint Goal is obsolete. This may be due to:

- ❑ The strategic changes in the organization
- ❑ Market changes that might impact the product

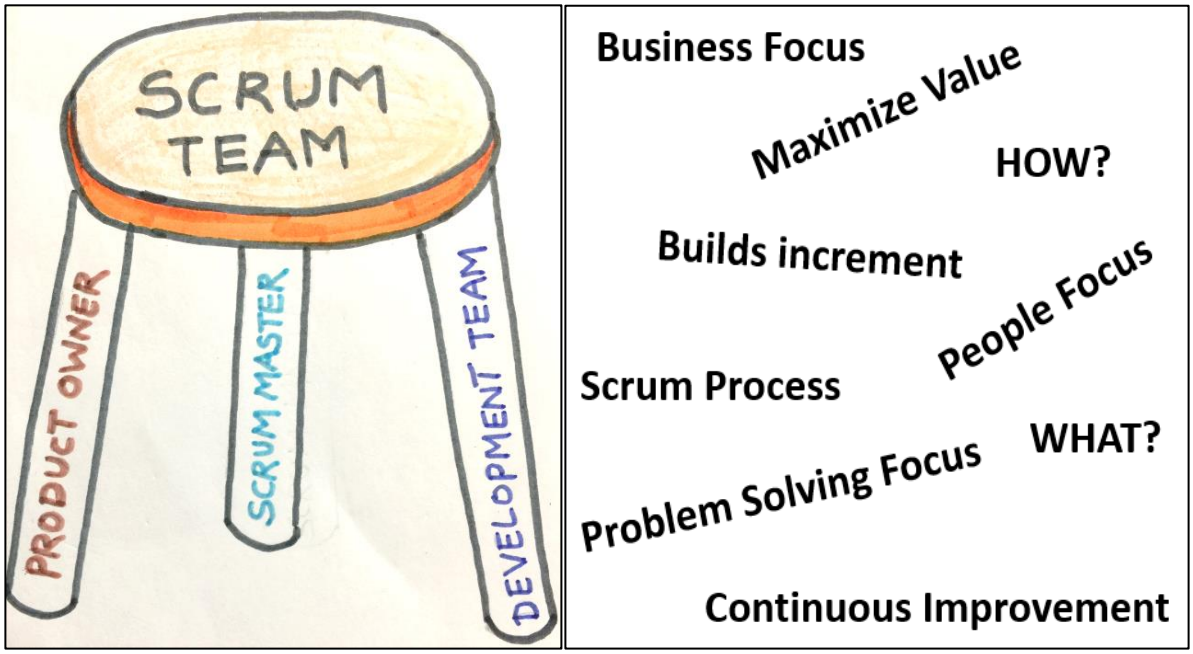
When a sprint is cancelled, the completed (Done) backlog items (work) would be reviewed. If the completed stories are potentially releasable then the Product Owner accepts. The incomplete backlog items are re-estimated and placed back in Product Backlog for future reference and prioritization. In case of a Sprint is cancelled for some technical reasons but the work needs to continue then the next sprint planning should be conducted immediately

SPACE FOR NOTES:

SCRUM

ROLES

Scrum Team:



Map the right-hand side characteristics to the three roles on the left side

The Scrum Team consists of a *Product Owner*, the *Development Team*, and a *Scrum Master*. The Development team also includes members like Testers, designers etc. The team model in Scrum is designed to optimize: Flexibility, Creativity and Productivity. Scrum teams deliver products iteratively and incrementally to maximize the opportunities for feedback.

Scrum teams are:

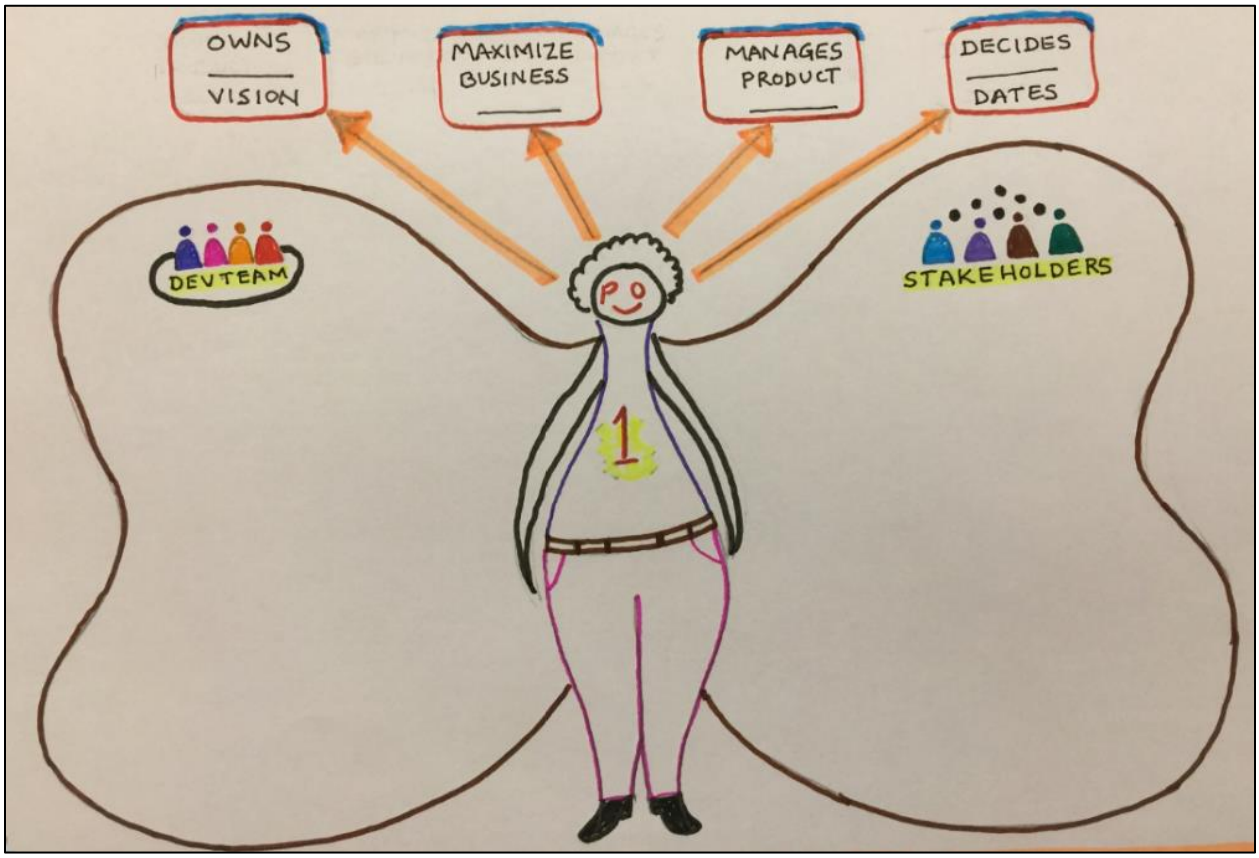
Self-organized: They choose the way they work on their own to accomplish the results without depending on others not part of the team.

Cross-functional: As a team, they have all the skills required to create a potentially releasable product increment.

SPACE FOR NOTES:

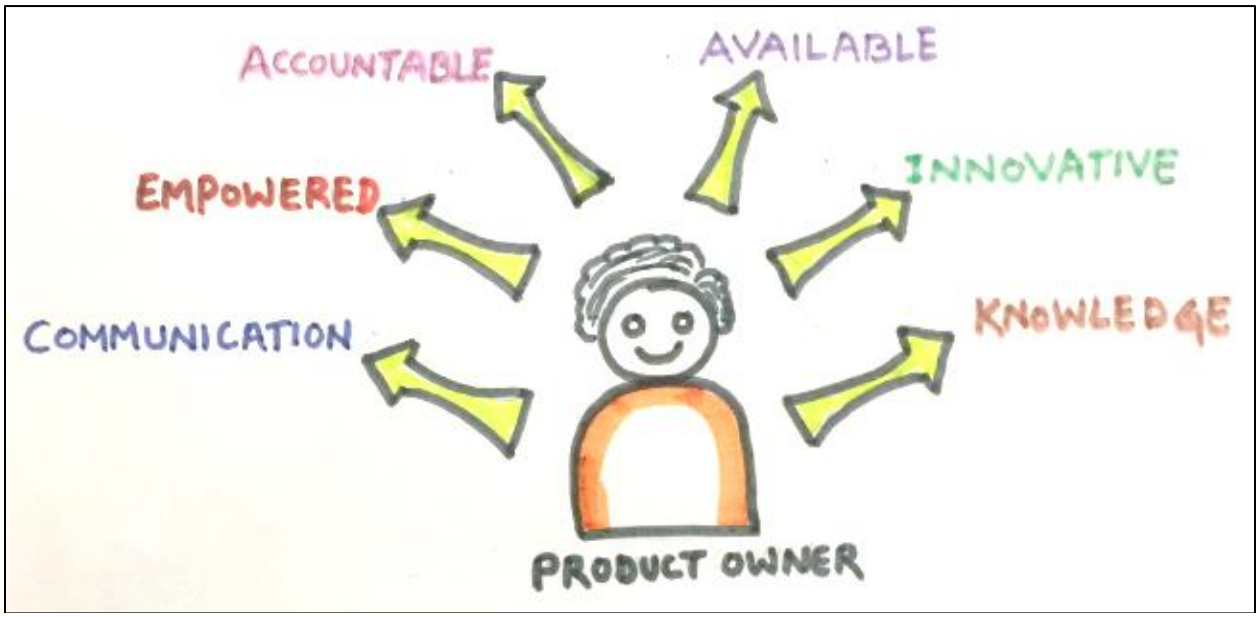
Scrum Roles & Responsibilities

PRODUCT OWNER:



- **Product Owner Responsibilities:**
  - Single person, not a committee
  - Responsible for Product vision & Roadmap
  - Manages Product Backlog (Content, Availability & Ordering)
  - Ensures profitability (ROI)
  - Works closely with Development team and Stakeholders
  - Represents the customer
  - Tracks progress of product through collaboration with Dev team
  - Reviews product backlog items and provide feedback to Development team
  - Provides clarifications to the Development Team
  - Leads Product Backlog refinement
  - Decides when to release the Product increment

- **Product Owner Characteristics:**



- **Communication:** Effectively communicate with Development team and Stakeholders
- **Empowered:** To make product prioritization and release related decisions
- **Accountable:** For the return on investment and product success
- **Available:** To the development team and stakeholders
- **Innovative:** To create unique features that increase the value of the product
- **Knowledgeable:** Domain knowledge of the Product

Product VISION describes, ***why we are building the product and what is the desired end state***. Vision can be created using an Elevator speech as described below template.

FOR <Target Customers>
WHO <Statement of Their Need>
THE <Product Name> **IS A** <Product Category>
THAT <Key benefits, Compelling reasons to buy>
UNLIKE <Primary competitor/Alternate Product>
OUR PRODUCT <Unique differentiation Factor>

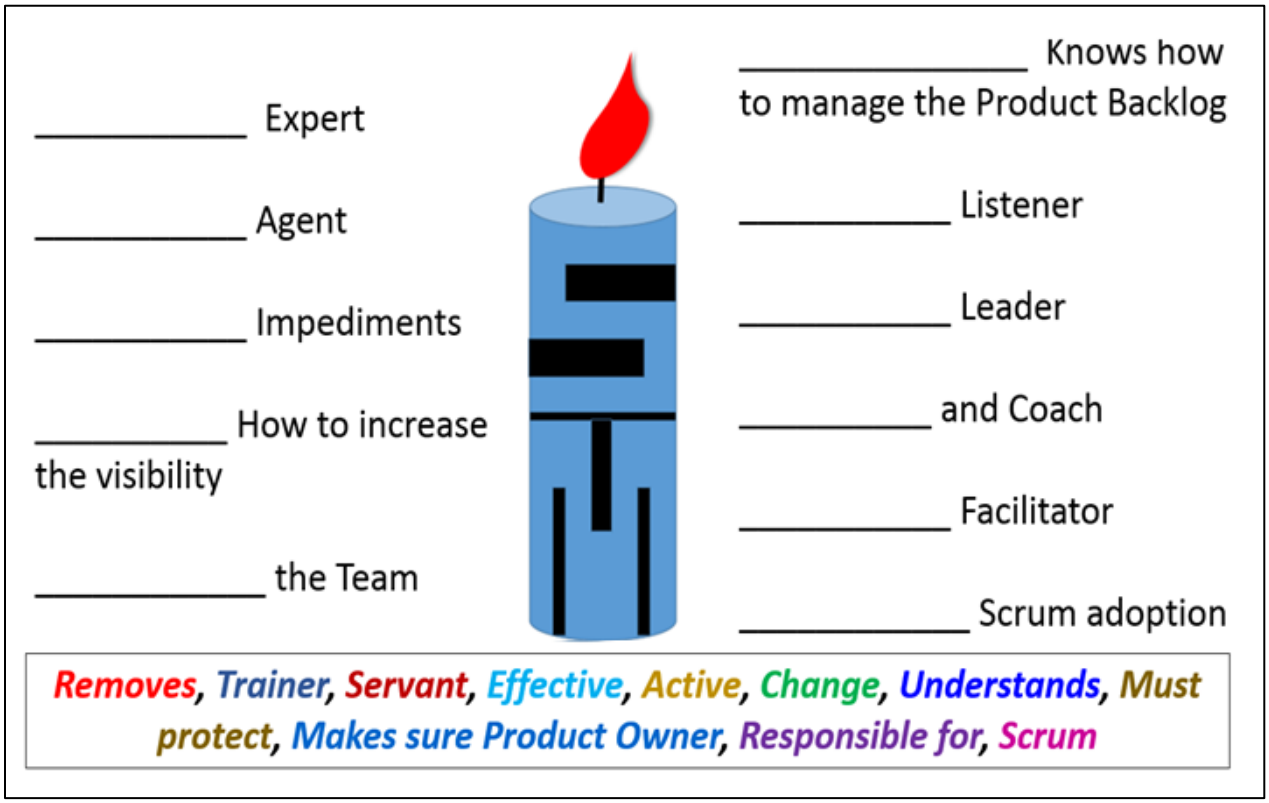
Exercise – PO Role Recap

Statement Related to Product Owner Role	Myth/Fact (M/F)
The P. O. must tell the Dev Team members how they must do their work	
The P. O. should decide the release dates of the product	
The P. O. must be the project customer himself	
The P. O. is the only one that can change priority of the Product Backlog items	
The best P. O.'s already worked as Project Managers	
The P. O. must balance the needs and desires of the different project stakeholders	
The P. O. must involve in daily scrum to guide the Dev team on their work	
The P. O. and the Scrum Master must never be the same person	
The presence of the P. O. is not mandatory at the Sprint Planning meeting	
To be more effective in the role, the P. O. must collaborate closely with the Dev Team	
The P.O has shared responsibility for ROI with Development team	
The P. O. must not estimate Product Backlog items along with the Dev Team	
The P.O can be a committee that collectively shares the P.O responsibilities	
The P. O. is responsible for defining which is the right product to be developed	

Note: Answers can be found at the end of the book

SPACE FOR NOTES:

SCRUM MASTER:

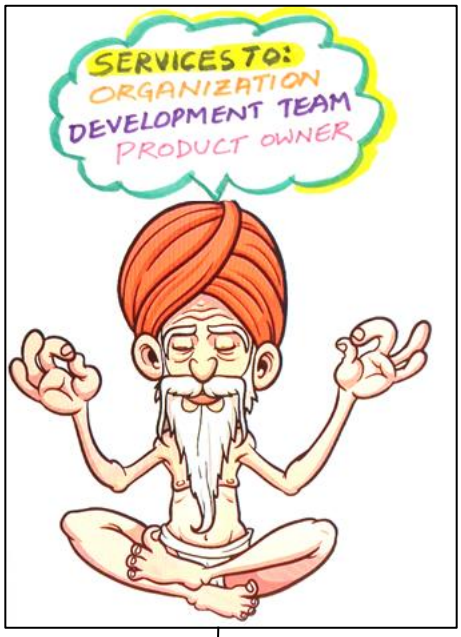


Note: Answer can be found in the end at Appendix section

- Responsible for Scrum implementation according to Scrum Guide (no customizations)
- Helps others understand Scrum theory, practices, rules and values
- Enables close cooperation across all roles
- Removes impediments
- Shields the team from external interfaces
- Facilitates Scrum events (based on the need)
- Improve the ability to improve across the organization
- Coaches the Development Team& PO and Leadership
- Helps Dev team to be innovative
- Deals with conflicts appropriately
- A change agent
- Servant leader

SPACE FOR NOTES:

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.



PRODUCT OWNER	DEVELOPMENT TEAM	ORGANIZATION

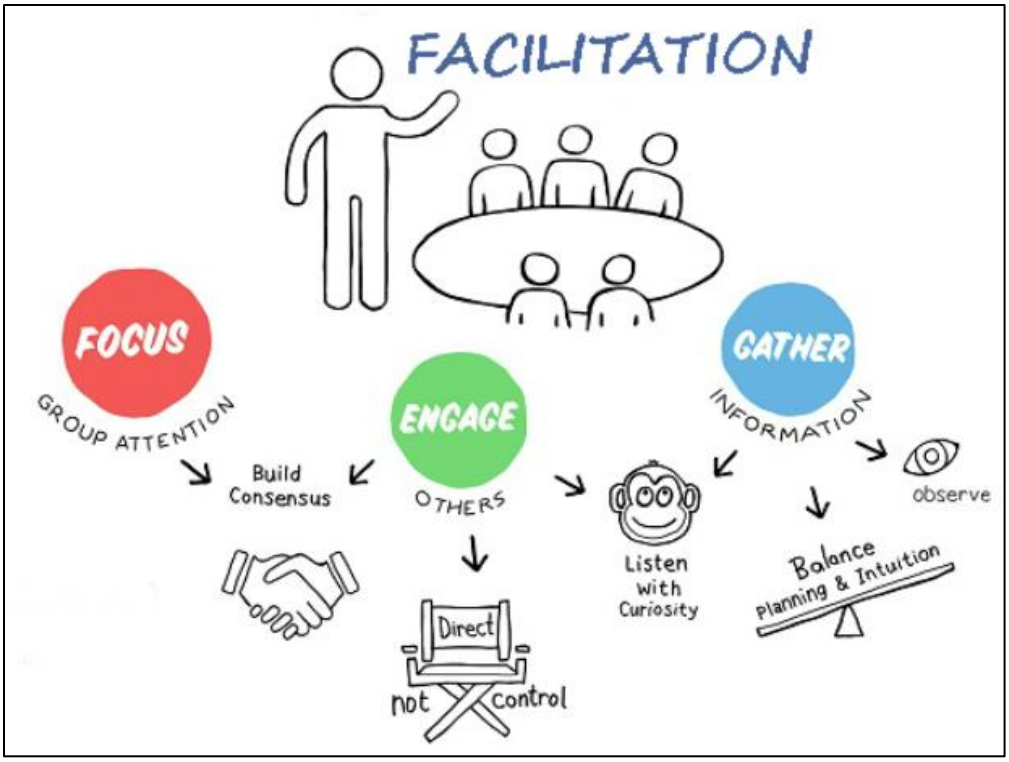
Scrum Master – As Servant Leader

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 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.)

Use "DOT VOTING" Technique to choose TOP 3 characteristics of a Facilitator:

F OCUSED
A CTIVE LISTENER
C ONCENSUS DRIVEN
I NTERACTIVE
L EGITIMATE
I NNOVATIVE
T RUSTWORTHY
A DAPTIVE
T IME KEEPING
O PEN MINDED
R ESPECT

Scrum Master as a Coach

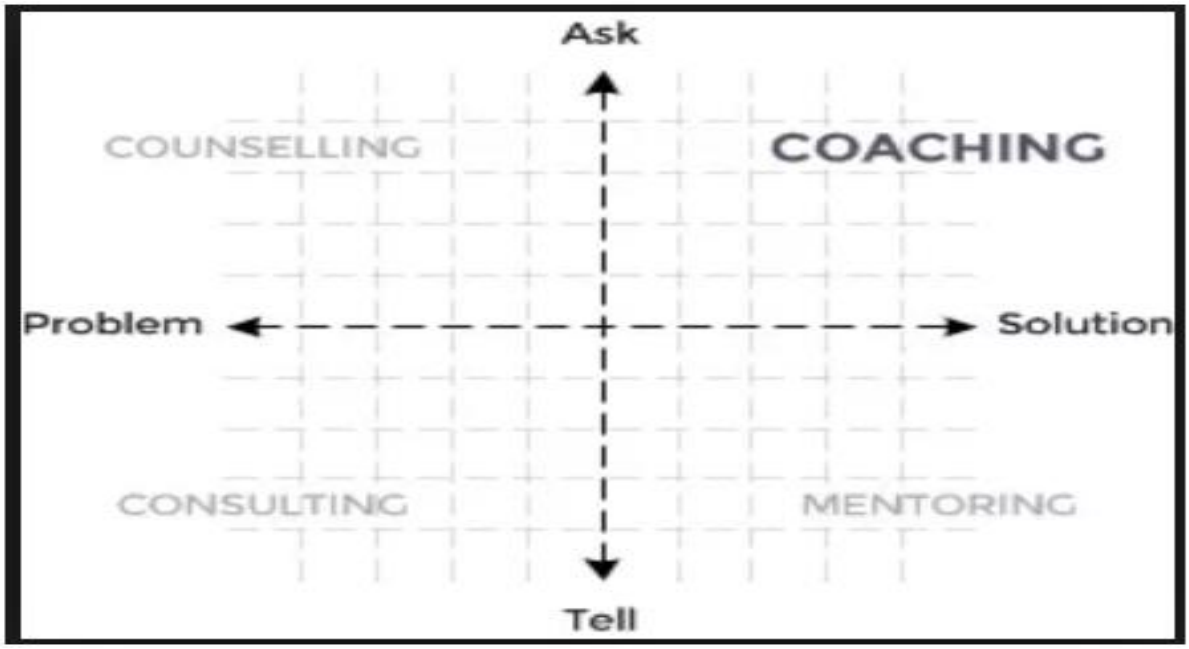
Coaching is unlocking a person's potential to maximize their own performance. It is helping them to learn on their own rather than teaching them. This means, coaching is all about "Asking" instead of "Telling". While asking, powerful questions will be used.

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.

How can Scrum Master Become Good Coaches?

- Lead by example
- Keep up the balance
- Set realistic pace
- Be cautious about your language
- Open to learn
- Accept feedback



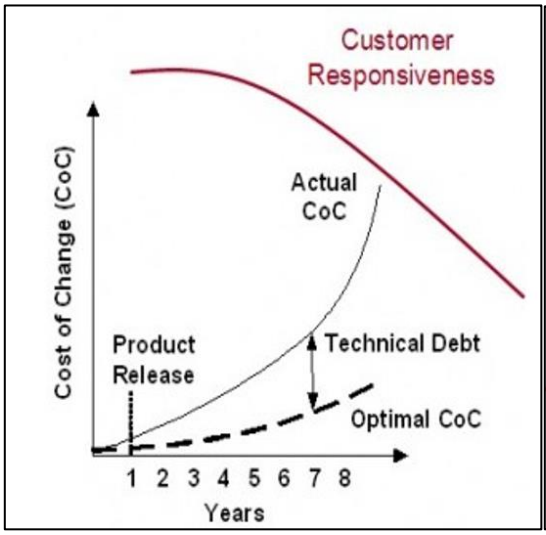
SPACE FOR NOTES:

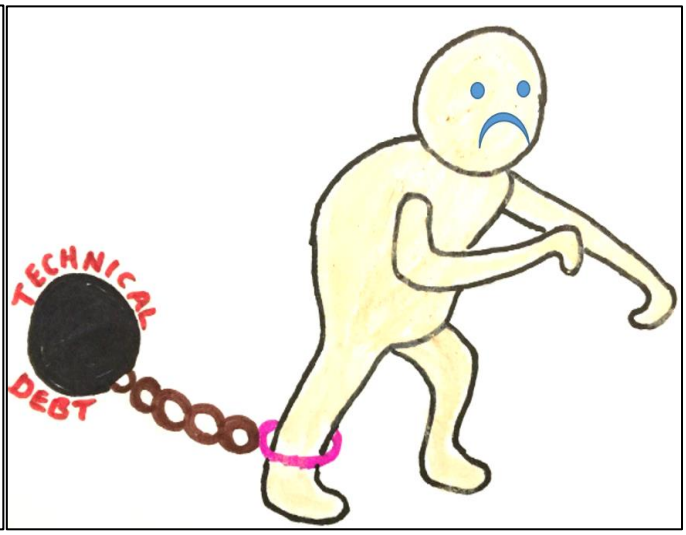
Technical Debt

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



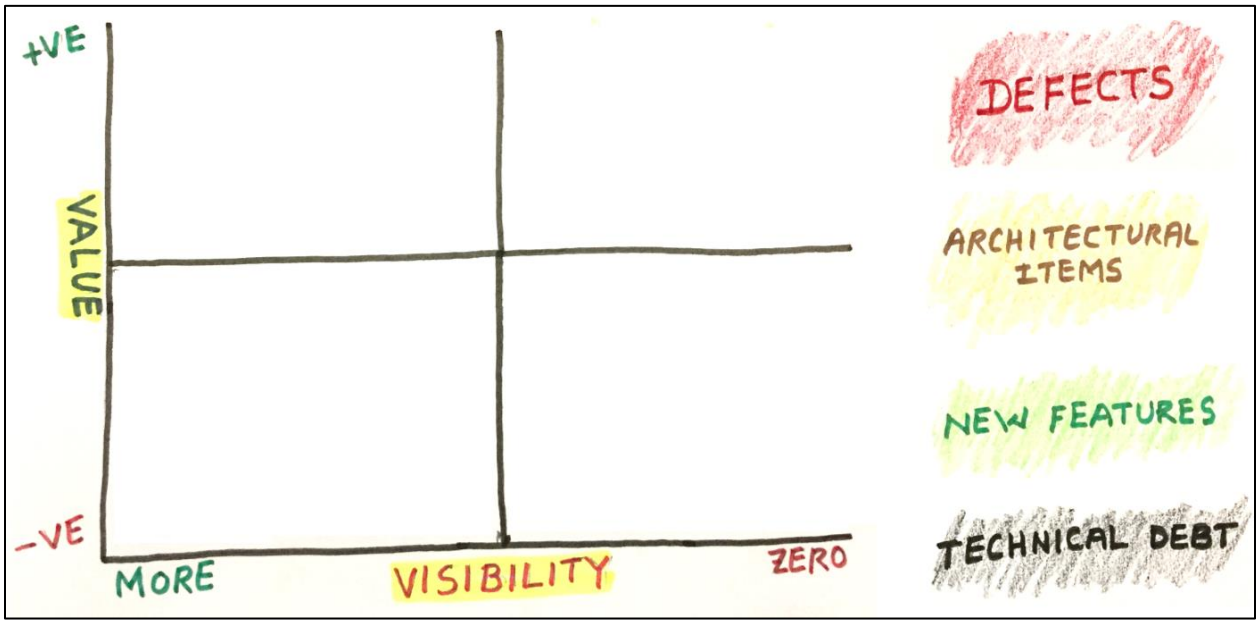


Technical Debt is the difference between the

OPTIMAL cost and the ACTUAL cost of change

SPACE FOR NOTES:

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



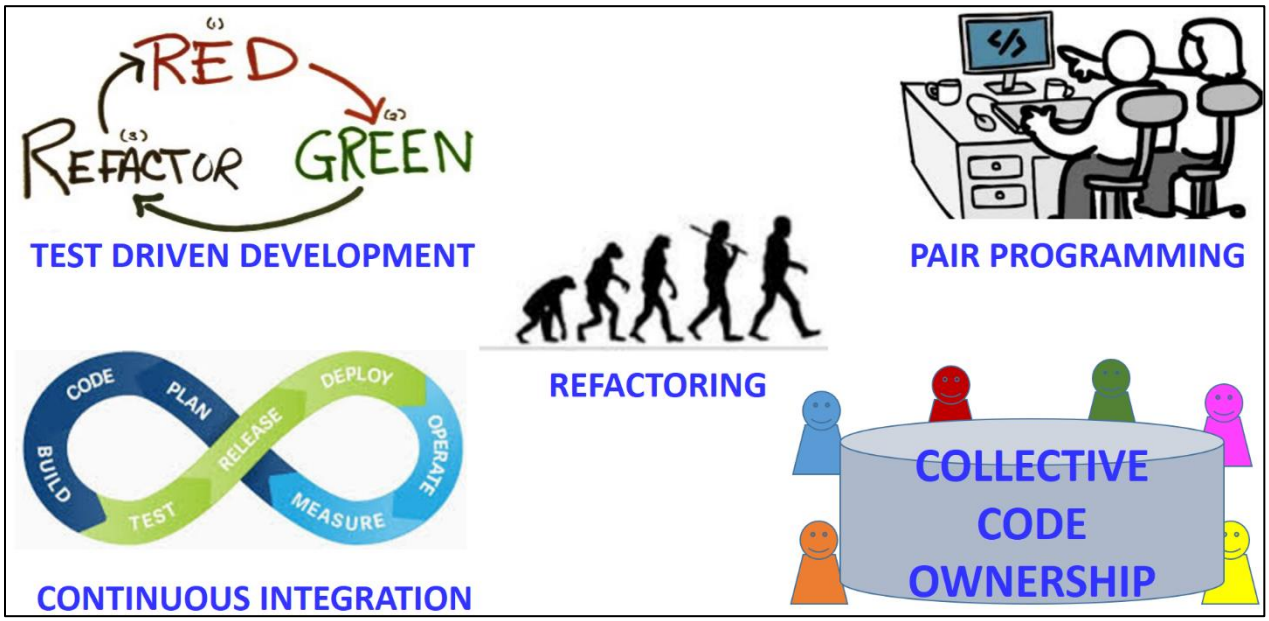
Note: Answers can be found at the end of the book

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).



Test Driven Development (TDD):

TDD is a unit testing practice in which first test will be written and the rite code to just pass the test and refactor the code. This is also called as red green refactor

Red: Create a test and make it fail.

Green: Make the test pass by any means necessary.

Refactor: Change the code to remove duplication in your project and to improve the design while ensuring that all tests still pass.

The Red/Green/Refactor cycle is repeated very quickly for each new unit of code.

TDD helps as safety net for the code, it gets confidence to make any change at any time by anyone in the team.

Continuous Integration (CI):

Continuous Integration (CI) is the practice, in software engineering, of merging all developer workspaces with a shared mainline ***several times a day***. It was first named and proposed as part of eXtreme Programming (XP). Its main aim is to prevent integration problems.

CI makes sure the software checked in to mainline is always in a state that can be deployed to users and makes the actual deployment process very rapid.

Integration issues can be identified early and can be fixed early using CI.

Refactoring:

Changing the design of the code without changing its behavior. That means the WHAT part remains same in the code and HOW part improved.

Example: team can write code for a feature using if…elseif….elseif…with many elseif but once it works, they change it to select- case model.

Pair programming:

Pair programming is, all code is produced by two people programming on one task on one workstation. One programmer has control over the workstation and is thinking mostly about the coding in detail. The other programmer is more focused on the big picture, and is continually reviewing the code that is being produced by the first programmer.

- Programmers trade roles regularly.
- DRIVER & NAVIGATOR

Collective code ownership:

All team members are equally responsible in owning the code. Anyone can make changes to any part of the code base. This will improve the knowledge of the code across tam members. Collective code ownership can be achieved through practices such as pair programming, swarming.

SPACE FOR NOTES:

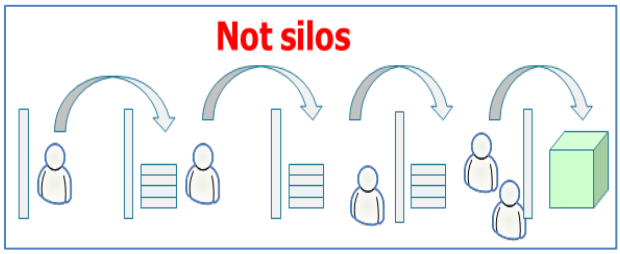
Exercise – Scrum Master Role Recap

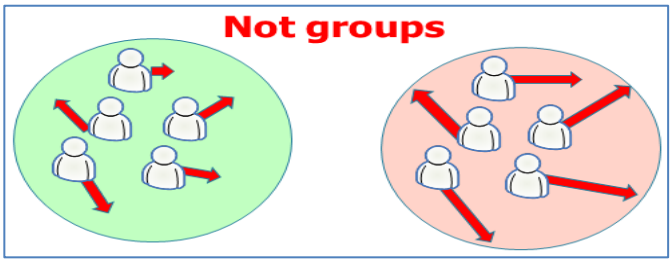
Statement Related to Scrum Master Role	Myth/Fact (M/F)
The best ScrumMasters already worked as a Dev Team member	
The ScrumMaster must have courage to do his job	
The best ScrumMasters already worked as Project Managers	
The ScrumMaster is the most important role in a Scrum project	
The ScrumMaster must have good interpersonal skills	
The ScrumMaster is one of the decision makers for the project schedule	
The ScrumMaster and the Dev Team estimate the Product Backlog items	
The ScrumMaster must protect the Dev Team from external interference	
The ScrumMaster usually cannot change the organization she works for	
The ScrumMaster is responsible for ensuring that Scrum is correctly used	
Ideally Scrum Masters should work full time in this role to obtain effective results	
The ScrumMaster must manage the work of the Dev Team	

Note: Answers can be found at the end of the book

SPACE FOR NOTES:

DEVELOPMENT TEAM:





Characteristics:

- Accountable as team
- Empowered
- Self-organized
- Cross-functional
- No other titles
- No sub teams
- Sized between 3 – 9

Prerogatives:

- Have right to choose how much work to be pulled into Sprint
- Has right to do everything to achieve the Sprint goal
- Estimation for Product Backlog items is done by only Development team
- Decides the architecture and design of the increment

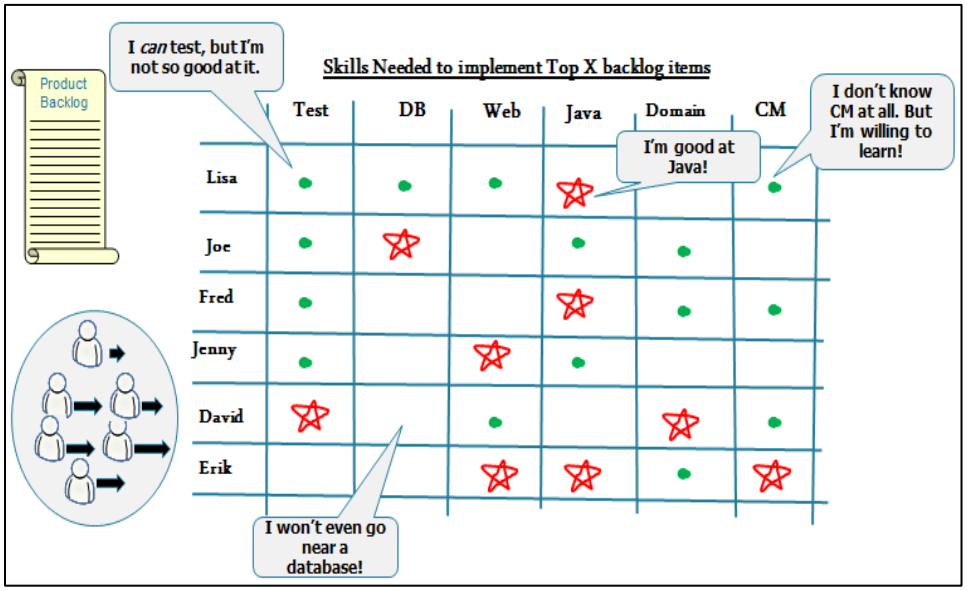
Responsibilities:

- Organizes themselves (their work), do not wait for work to be assigned
- Responsible to convert the PB into working software
- Reduces technical debt
- Tracks the progress of Sprint work on their own (Example: By updating Burndown chart daily)
- Helps Produce Owner for Product Backlog management (Estimation, Backlog refinement)
- Responsible for sprint goal

- Self-disciplined
- Transparent
- Respond to changes quickly
- Analyze stories, design, develop and test (as a team end to end ownership)
- Owns Definition of Done
- Resolve team level impediments on their own

Crossfunctional team

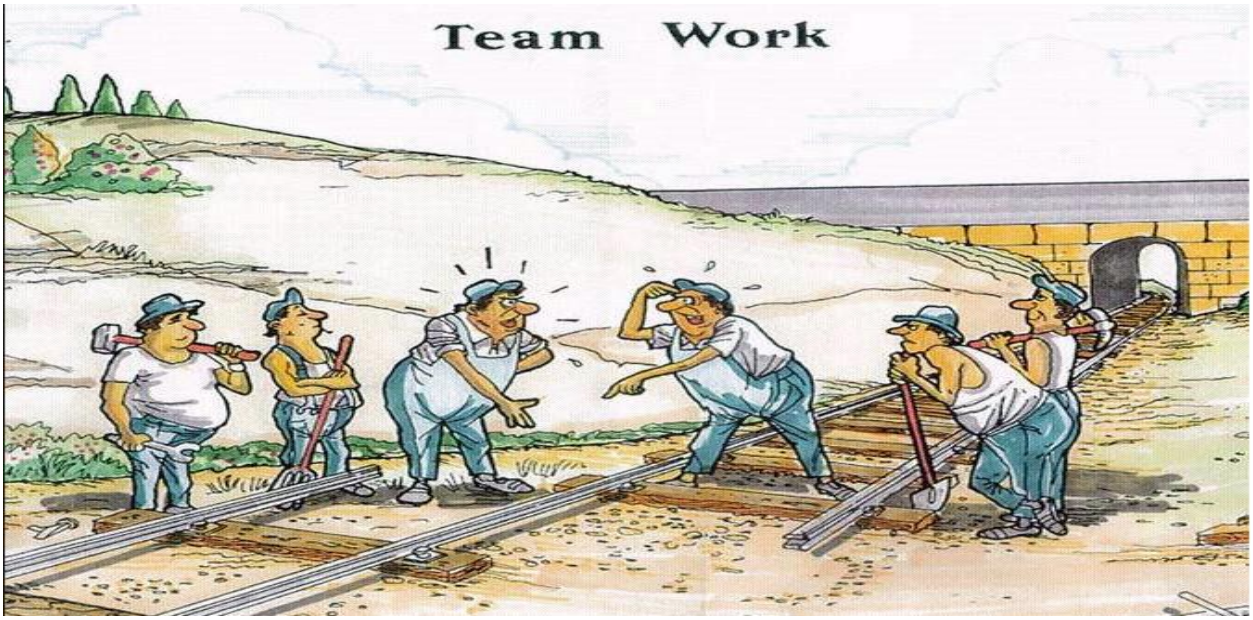
Doesn't mean everyone has to know everything (generalised specialists is the goal)



Reference: Henrik Kneiberg

SPACE FOR NOTES:

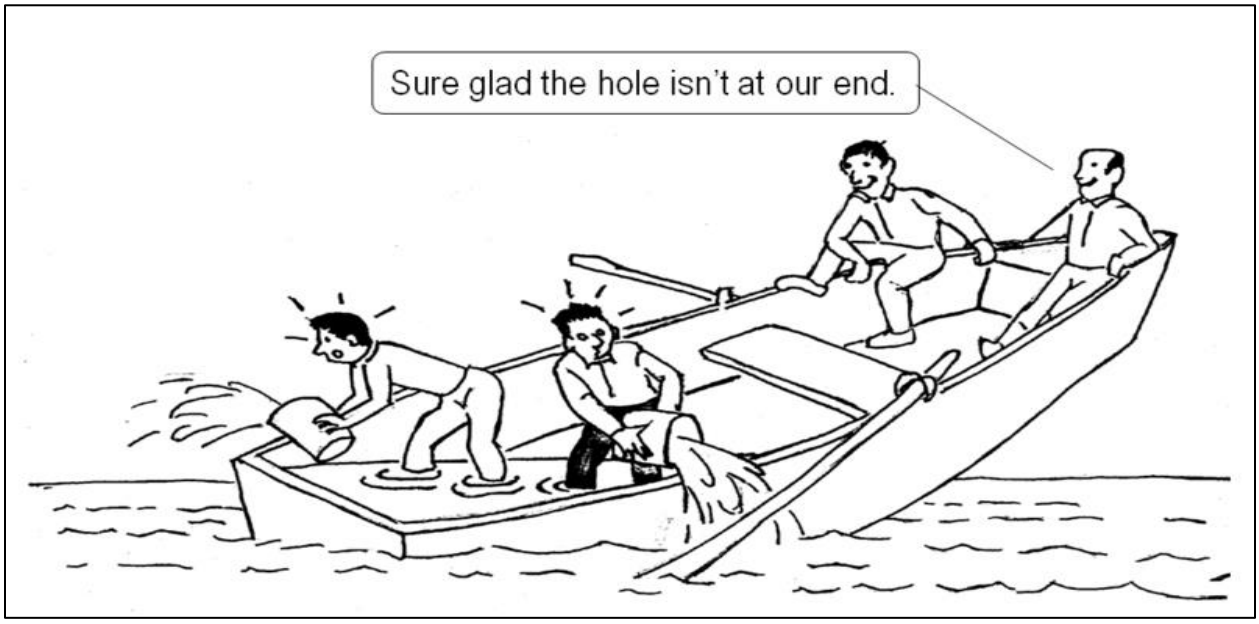
Is this Teamwork?



Whose fault?

High performance team

Collective responsibility



SPACE FOR NOTES:

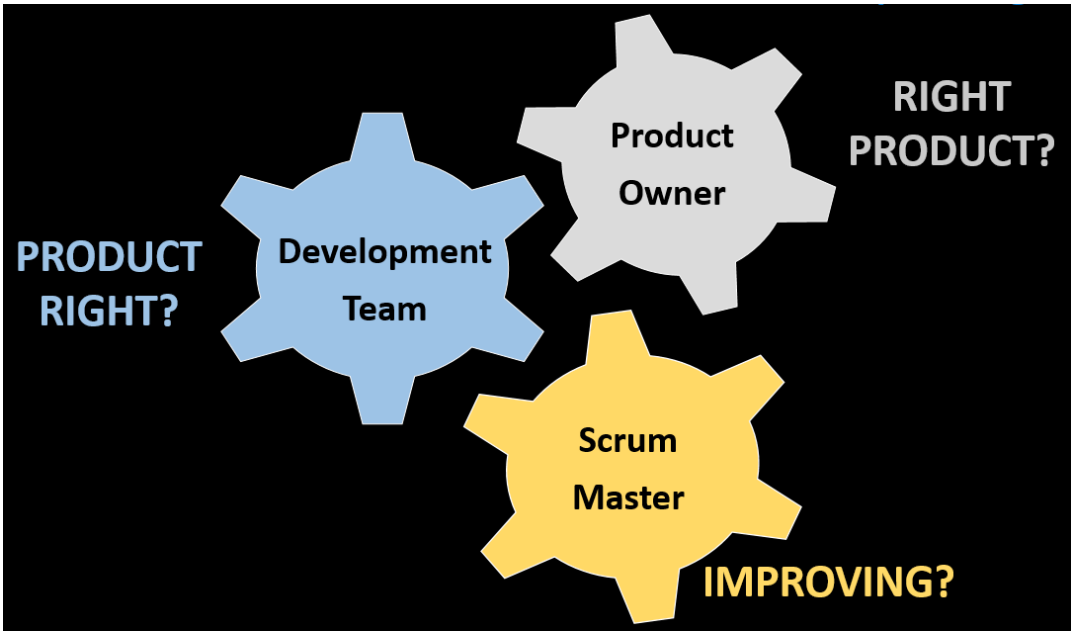
Exercise – Development Team Role Recap

Statement Related to Development Team Role	Myth/Fact (M/F)
The Dev Team is responsible for the quality of the product increment it generates	
Very small Dev Teams may not take benefit from using Scrum	
Only Development team can decide how much work can be pulled into the Sprint	
The Dev Team must plan themselves and manage its work on a Sprint	
The Dev Team must have all skills needed to generate the product increment	
Ideally, the Dev Team size should be between 3 and 9 members	
The Dev Team must inform the impediments to the Scrum Master at the Daily Scrum only	
The Dev Team must never interact directly with the project customer	
The Dev Team technical leader dictates the best technical approaches	
The Dev Team must keep the P. O. out of their way during the Sprint	
The best Dev Teams are able to work in several projects at the same time	
The Dev team can extend Sprint duration if they cannot complete the work of the sprint	

Note: Answers can be found at the end of the book

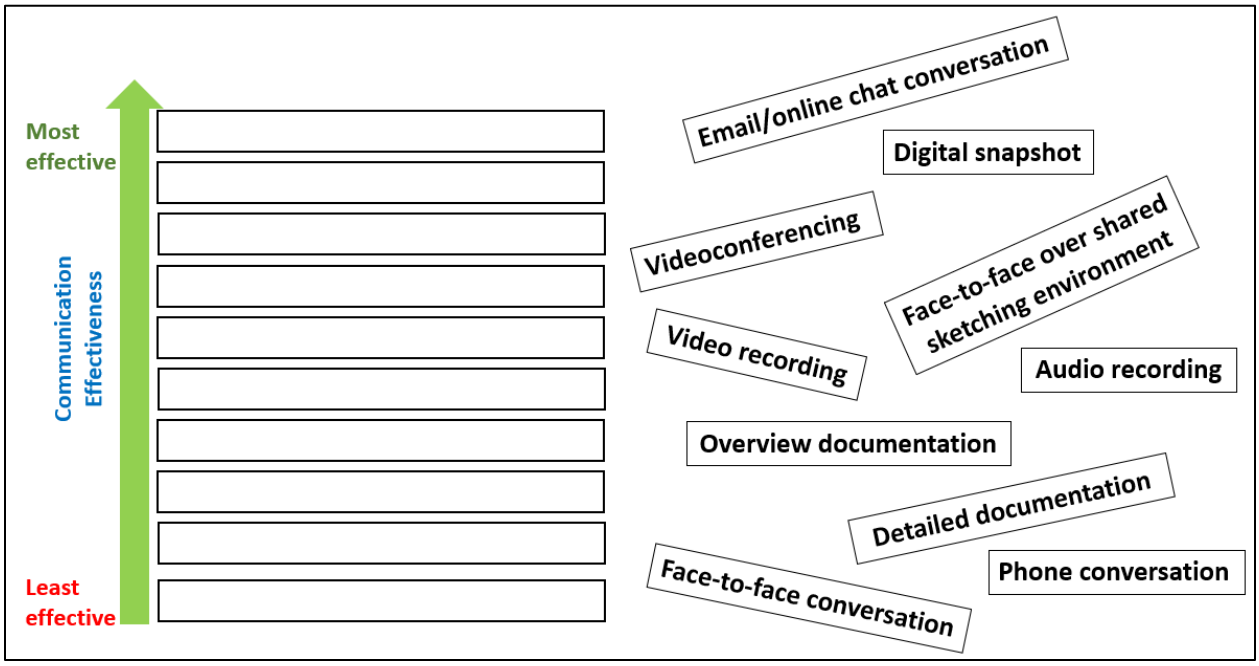
How the 3 roles are related?

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.



SPACE FOR NOTES:

Effective Communication:



Map the right-side items in the boxes on left-side based on their effectiveness

Answers are available at the end in Appendix

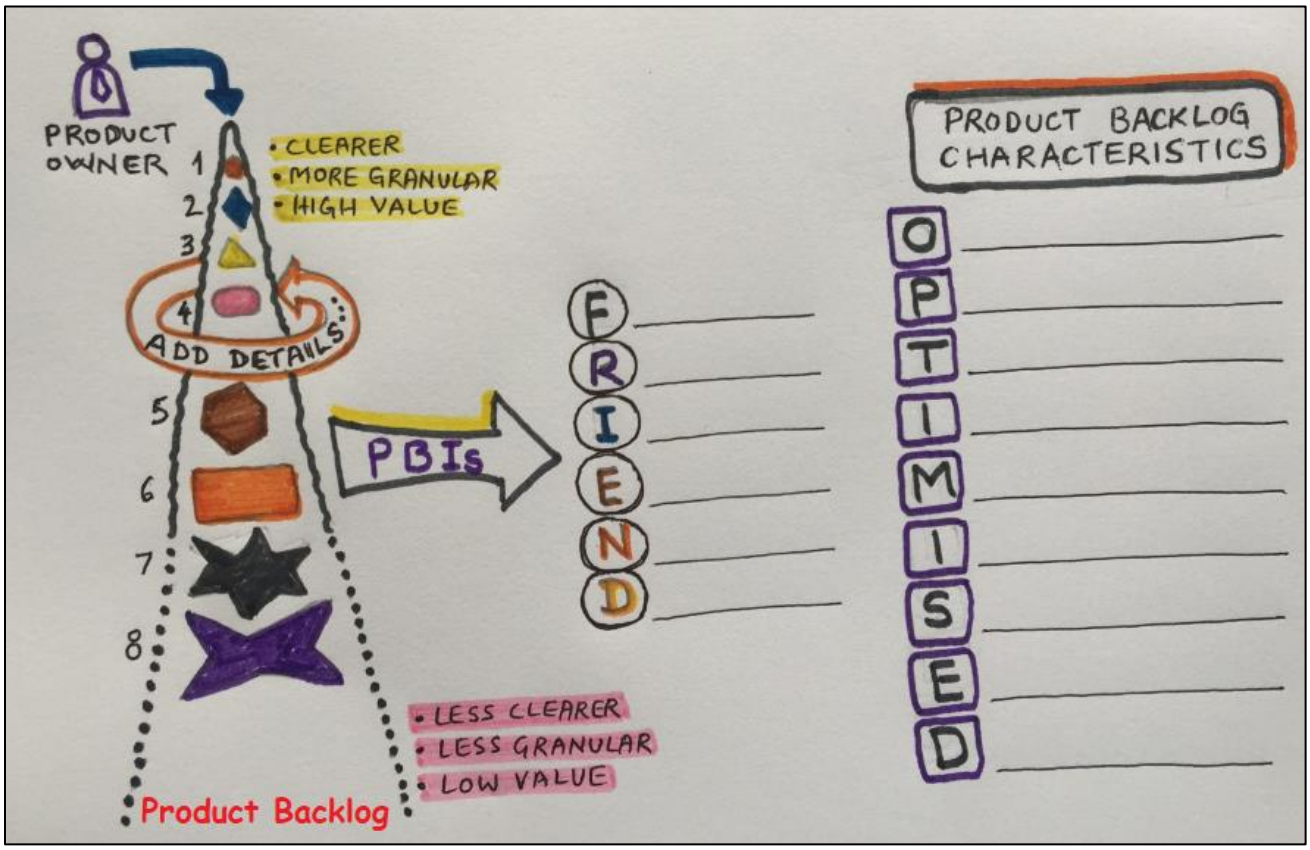
SPACE FOR NOTES:

SCRUM ARTIFACTS

Scrum Artifacts:

Scrum has 3 artifacts: Product Backlog, Sprint Backlog and Product Increment. These artifacts either represent work or Value and enhances transparency and provide opportunities to inspect and adaptation. These artifacts provide shared understanding across all the members.

Product Backlog:



Product Backlog is an ordered list of everything that is known to be part of the Product. It is the single source of requirements of the Product. Product owner owns and manages the Product Backlog in terms of: Content, Availability and ordering. Anyone can suggest items to the Product backlog. The Product Backlog evolves over time as and when more information is known about the product. It gets changed dynamically depending up on the need (Example: new items added, items removed, items split, reordered etc.). The Product backlog exists as long as the product exists. As the product is released and used by customers, based on the feedback the Product Backlog becomes larger and more exhaustive list.

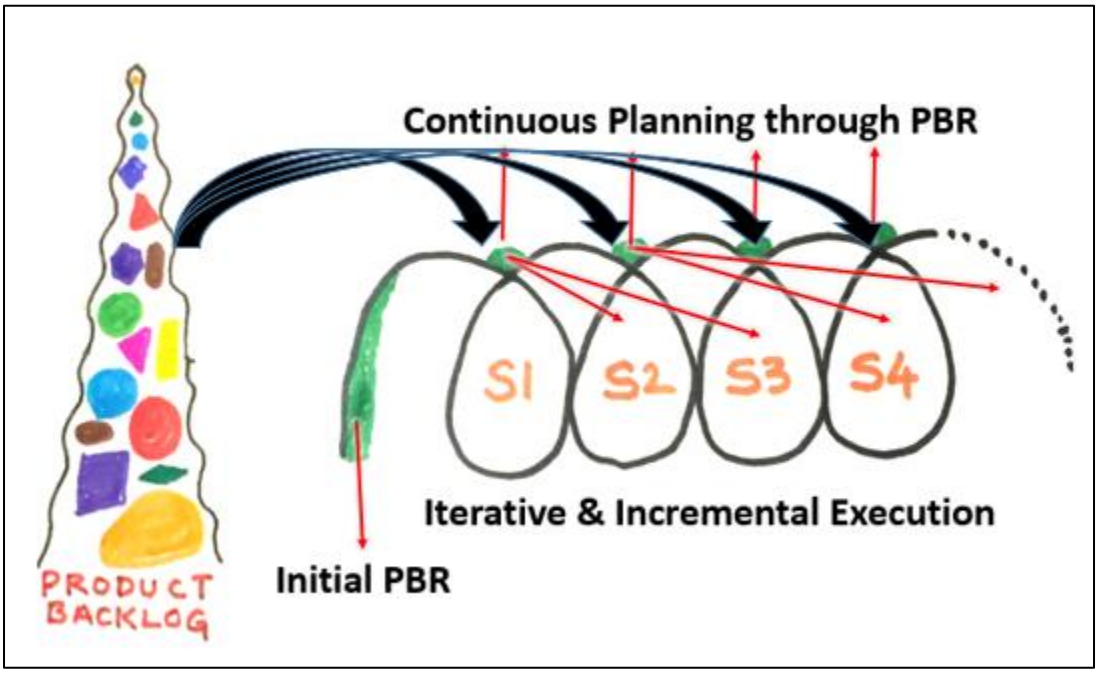
Items contained in Product Backlog are called as “Product Backlog Items (PBI)”. They may have Description, Estimate, Value and Order. Development team is responsible to estimate the PBIs. Value and Estimate will help Product Owner to order the Product Backlog. PBIs may also have

test descriptions that help understand the completeness when “Done”. They may be referred to as “conditions of satisfaction”.

Multiple teams often work on single Product Backlog and they may use an attribute in Product Backlog to specify the team working on the PBIs.

SPACE FOR NOTES:

Product Backlog Refinement:



Product Backlog refinement (PBR) in short, is an ongoing activity in which the Product Owner and Development Team members collaborate on the upcoming Product Backlog Items to make sure they are "Ready" for the next Sprint planning. This activity happens in the current to discuss items for the next one or two sprints. Focus is to discuss them and perform following activities:

- 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
- ….

The Scrum team decides how 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.

SPACE FOR NOTES:

Product Backlog Recap: Tick the correct box and cross the wrong one (***Answers @ end***)

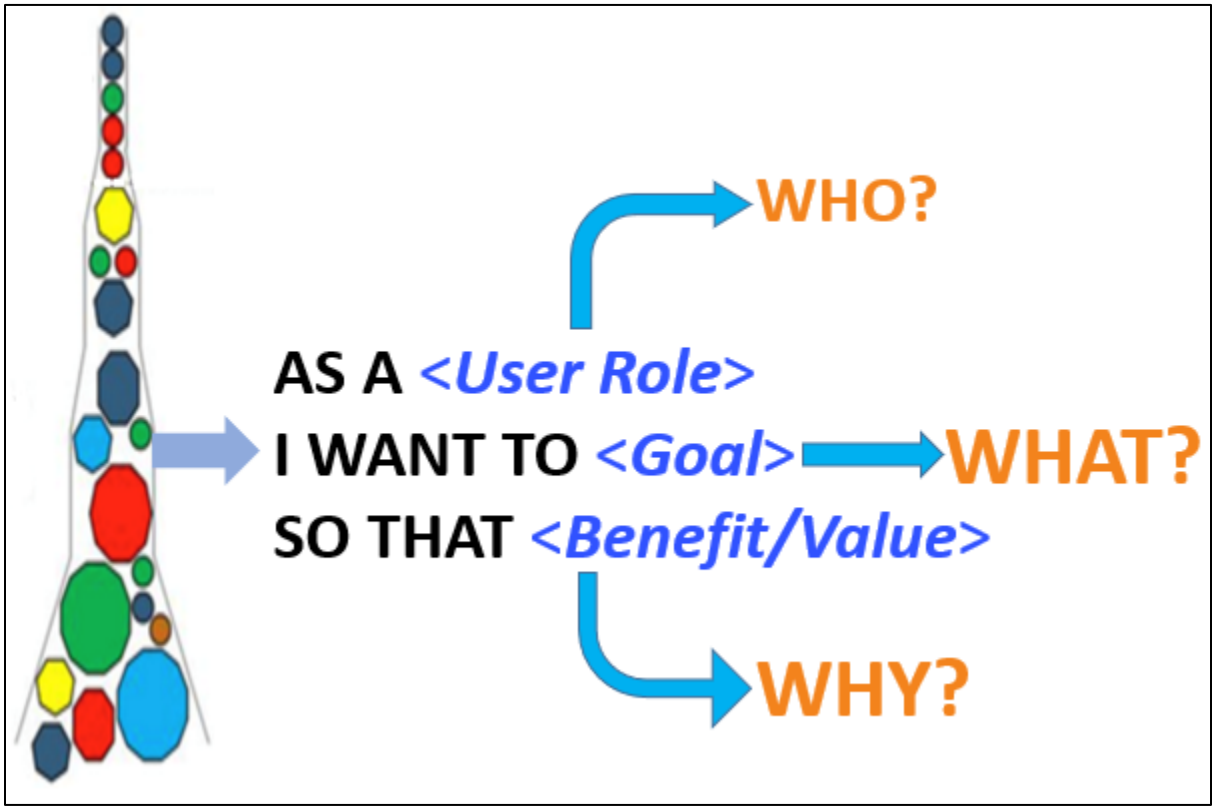
The Development team is responsible for estimating backlog items	Anyone can add items to backlog but PO is responsible to prioritize them
Dev team may work on critical engineering items without placing them in backlog	PO keeps the product backlog somewhere secretly so that no one can access it
Once an item is placed in Product Backlog, it will never be reordered	Product Backlog items usually have: Description, Order, Estimate and Value
Higher order items are usually clearer and more detailed than lower order items in PB	The Product backlog is always sorted by small values at the top and large items at bottom
Product backlog may contain functional, non-functional, infrastructural & defect items	Product backlog is incomplete
Product backlog is owned by Development team	Scrum Master will prioritize Product backlog in the absence of PO
Product backlog enhances the transparency	More than one team can work for a single Product Backlog

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

From Mike Cohn's definition "A user story is a short, simple description of a feature told from the perspective of the person who desires the new capability, usually a User or Customer of the system"

In XP it is called as "User story" and in SCRUM it is called as "Product Backlog Item (PBI)"

User stories typically written in the following format:



SPACE FOR NOTES:

Example:

What is the significance of this format? (Reference: Mike Cohn's blog)

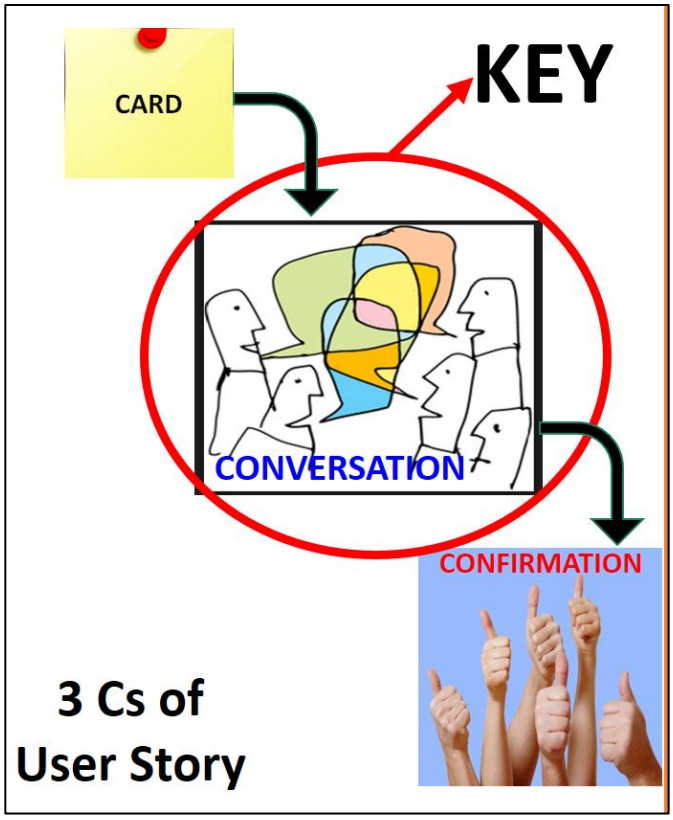
Reason-1: When requirements are put in the 1st person I'm tempted to say magical happens. When you start saying "As a <….> I want<…> the intensity of imagination Will be high

Reason-2: Having a structure to the stories actually helps the product owner prioritize. If the product backlog is a jumble of things like: Fix exception handing, Let users make reservations, Users want to see photos, Show room size options … and so on, the product owner has to work harder to understand what the feature is, who benefits from it, and what the value of it is.

Reason-3: Writing stories with this template actually suppresses the information content of the story because there is so much boilerplate in the text.

According to Ron Jeffries, User Story comprises of: 3Cs (Card, Conversation, Confirmation)

- **Card (Describes the goal)**
- **Conversation (Collaboration to get more understanding)**
- **Confirmation (Acceptance criteria)**



- User stories are invitations for conversations.
- User stories emphasize verbal rather than written communication.
- User stories are comprehensible by both business people and the developers.
- User stories are the right size for planning and estimating
- User stories work for iterative development.
- User stories encourage deferring detail until the best understanding is available.
- Encourage deferring details through conversation
- Try to use "Specification by example" wherever required but not mandatory for all

"Conversation" is the key to get clear understanding through face-to-face communication

"Confirmation" is related to the acceptance criteria for the particular user story. Also called as "conditions of satisfaction" with respect to the functionality.

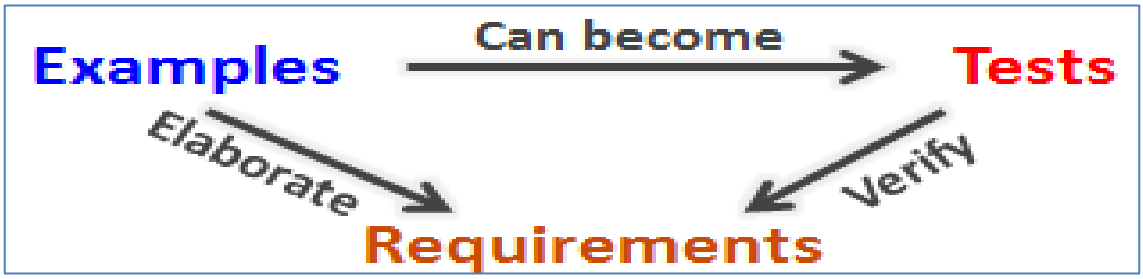
Key Practices:

Teams that apply Specification by example successfully commonly apply the following process patterns:

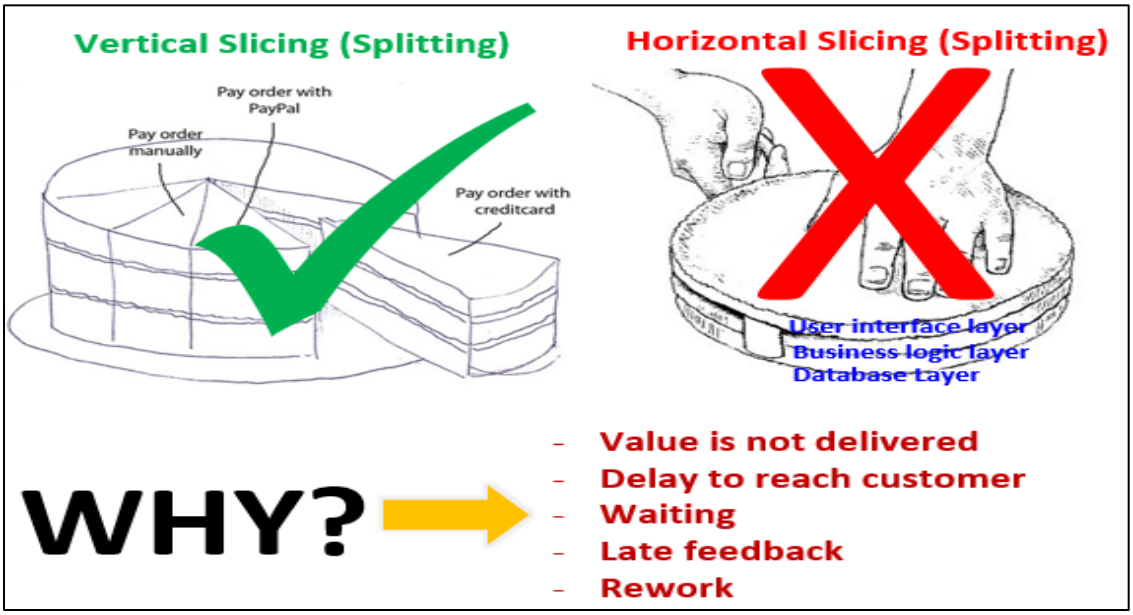
- Deriving scope from goals
- Specifying collaboratively - through all-team specification workshops, smaller meetings or teleconference reviews
- Illustrating requirements using examples

- Refining specifications
- Automating tests based on examples
- Validating the underlying software frequently using the tests
- Evolving a documentation system from specifications with examples to support future development

Specification by Example:



Split stories if they are oversized:



- ❑ Stories come in different sizes
- ❑ Try to make them as simple as possible by splitting
- ❑ How do you split user stories?
- ❑ **Workflow steps:** Look for steps in the workflow that chain different roles/personas together or very distinctly different functions that can be independently done.

  Example: Content Authoring & Approval

- As an author, I want to be able to submit my article.
- As an editor, I would like to get notified when an article has been submitted so that I can review it.
- As an editor, I need to approve an article.
- As an editor, I would like to be able to request more information

❑ **Operations pattern:** Focus along Operations (think high level methods or CRUD type operations).

Example: As a shop keeper I want to manage the products being sold in my online store so that I can sell what people want to buy. This could be become:

- As a shop keeper I want to add and remove products from my online store so that I can sell what people want to buy.
- As a shop keeper I want to edit product details in my online store so that I can avoid recreating a product to fix a typo … (and so on)
- *Look for generic verbs such as "manage" or "administer" which hints at multiple more detailed operations. (Sort of use-case like…)*

❑ **Business rules variations pattern:** Find ways to split stories based on business rules deviations.

Example: Payment currency must be specific to purchase location

- Cash payment denomination amount must not be greater than . . .
- Payment change amount is calculated as . . .
- Receipt bar code is designed using . . .

Warning: these variations could be articulating various acceptance criteria.

❑ **Simple → Complex pattern:** Look for general stories that hide complexity, when the definition of the acceptance criteria uncovers varied ways to approach this, then you can split along this variance...

Example: As a loan applicant, I want to calculate my mortgage payments might be made more specific in ways such as …

- … calculate payments manually

- … using an online spreadsheet template
- … using an online calculator

❑ **Variations in data pattern:** Choose data objects that may have variations based on roles and actions.

Example: Suppose that there are data objects called Product, Payment, and Receipt. In this instance, the idea would be to focus on specific data types for each object type. So, for Product, there might be data types such as Book, DVD, and Gift Card.

You can then work with the Product Owner to identify the data type or types with the highest business value and split the story accordingly.

❑ **Mock UI pattern:** Create a prototype/whiteboard of the UI when it is known to be complex; ID areas where related functionality is present and story out each area. Throw away the UI.

❑ **Data entry methods patter:**Examining various options for data entry at the UI can be done. Each option is essentially the same story and thus is a refactoring of a prior story. Great for identifying MVF options…

Example: As a user, I can search for flights between two destinations.

- …using simple date input.
- …with a fancy calendar UI.

❑ **Defer performance pattern:**Look for Opportunities to Defer Work for later (and thus refactor of the code). “Simplicity--the art of maximizing the amount of work not done--is essential.” – Agile Manifesto Principle #10

Example: Focus on user functionality 1st then various –’ilities’:

- Scalability
- Performance

Caution: “Continuous attention to technical excellence and good design enhances agility.” – Agile Manifesto Principle #9

❑ **Breakout a spike pattern:** Often Stories are not necessarily complex, but just have many unknowns. Turn possible acceptance criteria into questions and investigate these.

Example: Suppose you didn't understand how PayPal worked…

***Turn**:*

As a seller I want to collect payment with PayPal as it is universally accepted, and this will increase how I get paid.

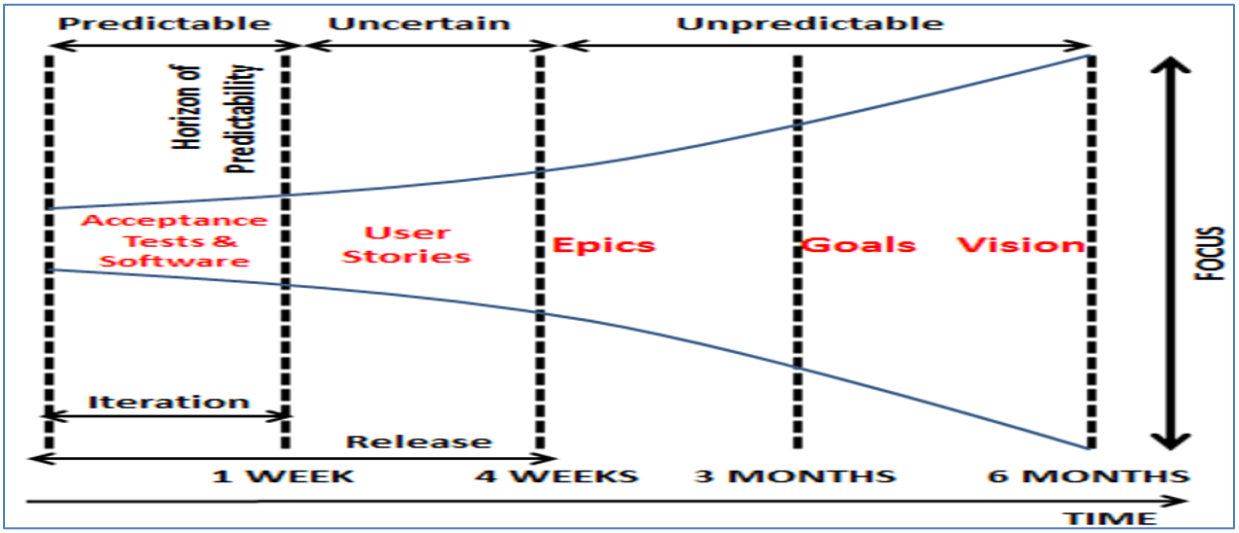
into…

- Spike: *How does PayPal work?*
- *How do I know I have a successful payment?*
- *How do I know when a payment is unsuccessful and what options can I present to the buyer?*
- Once I answer these I can create a valid story (or set of stories).

IMPORTANT:

- ❑ Do not split into tasks until Sprint Planning
- ❑ Do not split stories like waterfall phased

Visibility of User stories against the timeline:



SPACE FOR NOTES:

What is a Good User Story?

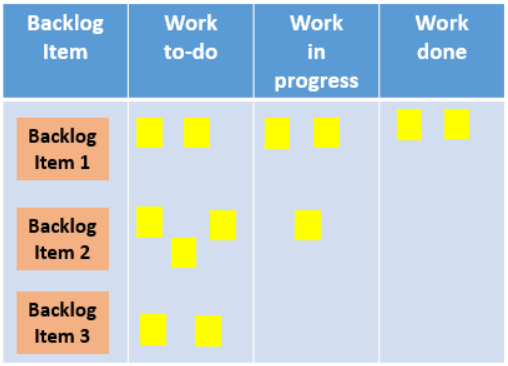
In order for a user story to be clear and unambiguous, it should satisfy INVEST principle as described below table. Whenever you are writing a story, ensure that it satisfies the "INVEST" principle. If it is not satisfying, then try to split as per the guidelines provided above so that it will meet "INVEST" principle.

In general, "Epics" (A big user story) may not satisfy "INVEST" principle. However, they will be groomed as part of "Product Backlog Grooming" meeting so then Epics will be divided into small user stories by then they will satisfy "INVEST". This is called "Rolling wave planning" or "Progressive Elaboration".

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.**

SPACE FOR NOTES:

Sprint Backlog:



- Contains the backlog items & a plan to deliver the increment
- Output of Sprint planning meeting
- A Forecast by Development team on what they can deliver
- Owned by Development team
- 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, Dev team adds it to the Sprint Backlog
- As work is completed, remaining work is updated
- Unnecessary elements are removed
- Highly visible and frequently updated during the sprint
- Keeps the sprint work transparent

The Sprint Backlog makes visible all the work that the Development Team identifies as necessary to meet the Sprint Goal. To ensure continuous improvement, it includes at least one high priority process improvement identified in the previous Retrospective meeting.

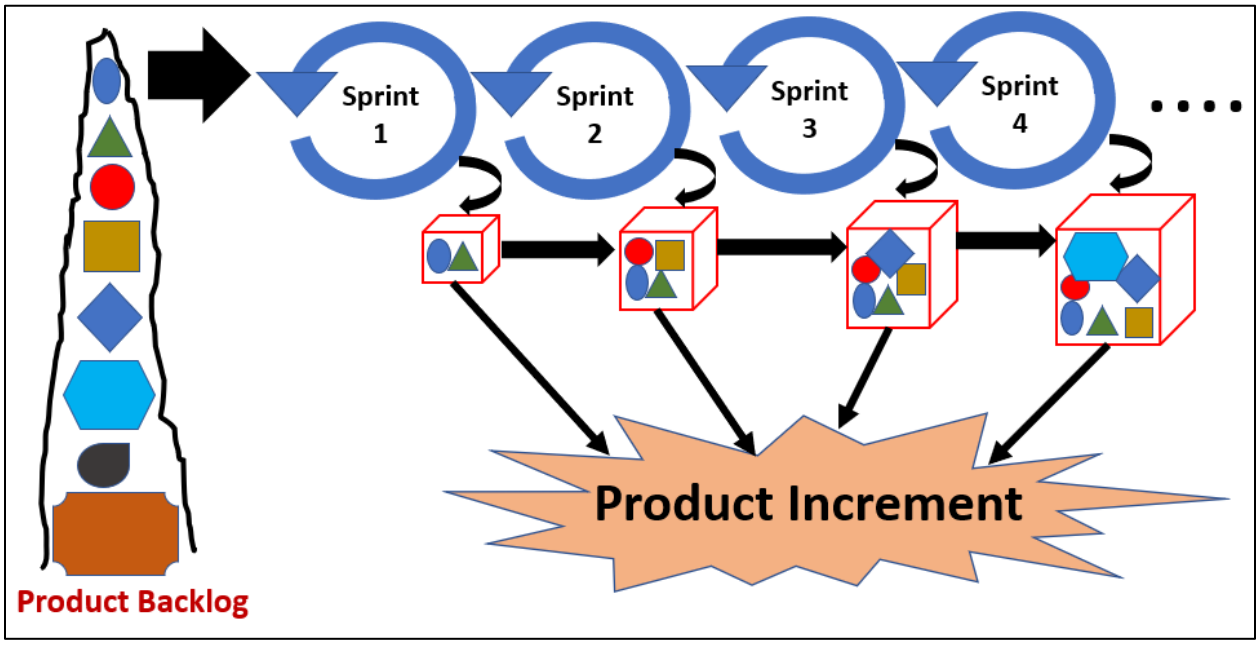
The Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum. The Development Team modifies the Sprint Backlog throughout the Sprint, and the Sprint Backlog emerges during the Sprint. This emergence occurs as the Development Team works through the plan and learns more about the work needed to achieve the Sprint Goal.

The Sprint Backlog is a highly visible, real-time picture of the work that the Development Team plans to accomplish during the Sprint, and it belongs solely to the Development Team. The Development team usually also cover the items that are required to meet Definition of done in Sprint Backlog.

Only the Development Team can change its Sprint Backlog during a Sprint

SPACE FOR NOTES:

Product Increment



The Increment is the sum of all the Product Backlog items completed during a Sprint and the value of the increments of all previous Sprints. At the end of a Sprint, the new Increment must be "Done," which means it must be in useable condition and meet the Scrum Team's definition of "Done".

An increment is a body of inspectable, done work that supports empiricism at the end of the Sprint. The increment is a step toward a vision or goal. The increment must be in useable condition regardless of whether the Product Owner decides to release it.

Potentially Releasable Product Increment helps to:

- *Elicit feedback and validate assumptions*
- *Increases the Business Value*
- *Reduces the Risk by receiving early feedback*
- *Will create transparency to the stakeholders*

***** Only members of the Development Team create the Increment *****

SPACE FOR NOTES:

AGILE ESTIMATION

(Not part of Scrum)

What is Estimation & Why Estimation?

Estimation is predicting the Size, Cost, Schedule etc…

Estimation can never be accurate. It is a guess value derived based certain conditions, parameters and criteria.

Estimation in Agile is different than traditional. We believe in "It is okay to be roughly accurate rather than precisely wrong"

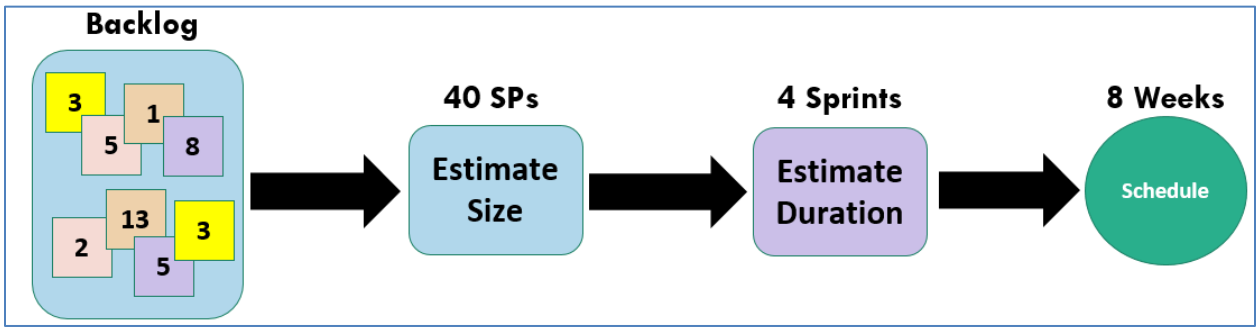
Two factors of Estimation:

Accuracy – How something is close to reality

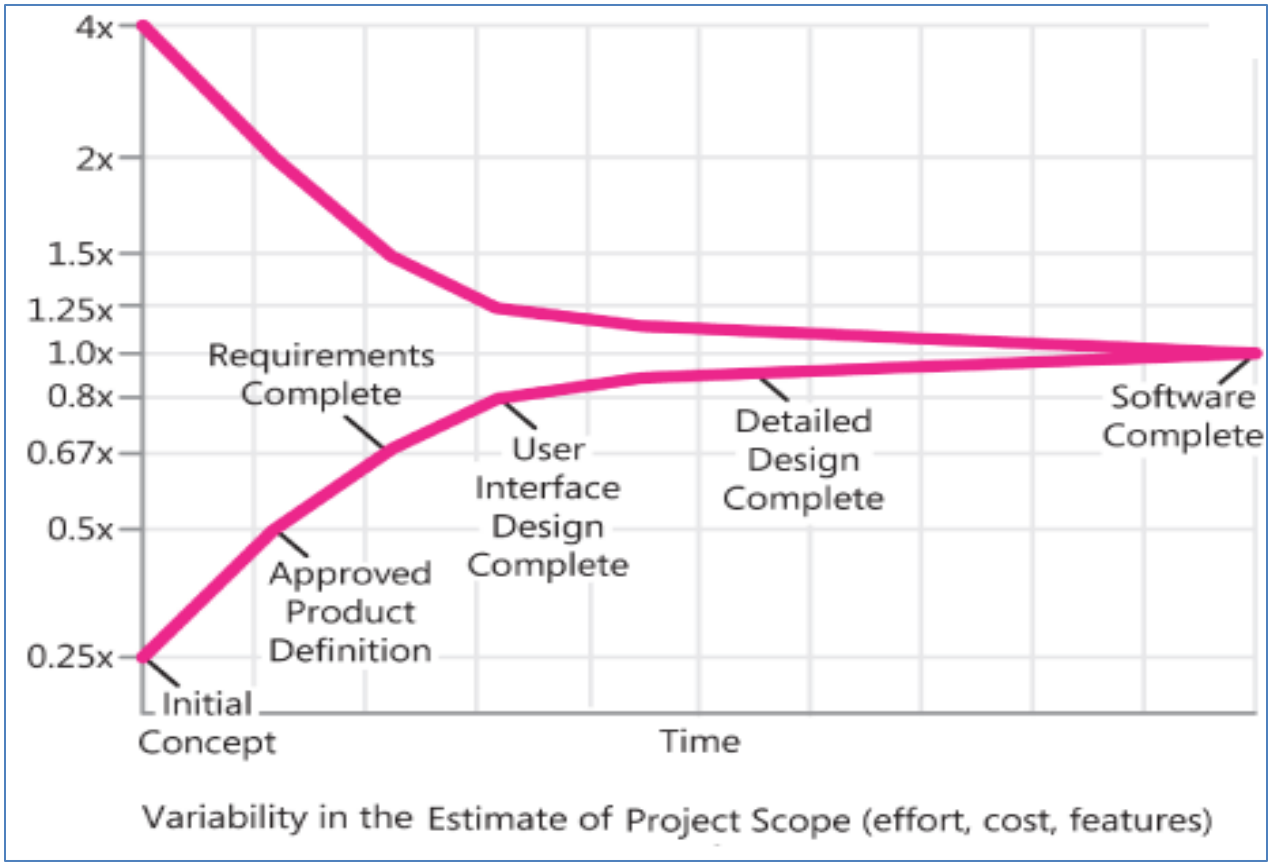
Precision – Degree to which repeated measurements shown (consistency)

We generally do estimation in order to forecast the time or cost or can a given scope be delivered in a given time etc

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.

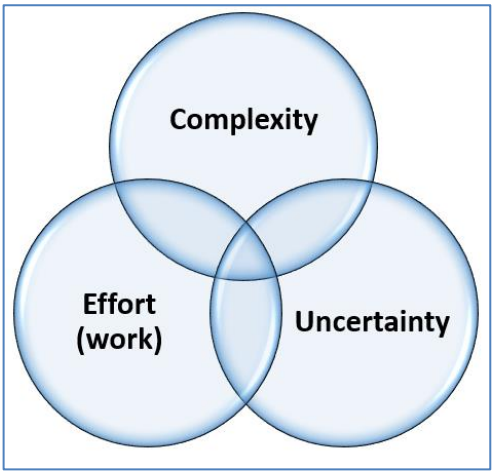


SPACE FOR NOTES:



Agile Estimation Technique (Story Point)

"Definition Of Done" plays a vital role in the estimation process, because DoD gives the entire picture of scope of work involved in the iteration/project.



Three factors that influence the Story Point estimation:

Complexity: Gives the level of complexity of the story/feature (Functional and Technical)

Uncertainty: Gives the level of Risk (unknowns) involved in the story/feature

Effort: How much effort (work) the story/feature may need to achieve DoD

Estimate the SIZE in Story Points and Estimate the EFFORT in Task Hours

Why size of the Backlog item is important?

Assume that you go to a new restaurant where you want to order a cup or bowl of soup. You can have either a full or half bowl of soup. When you order half bowl soup, you will clearly know that it is smaller than full bowl.

You are not ordering exact size such as "Get me 330 ml soup".

When you know the size you can predict the quantity to the closest accuracy.

So, size will help to arrive at the duration for software projects. It is the relative measure.

In Agile projects, the size is applied to the "User Stories". Each story will be estimated with a size Story Points. This size is relative size to other stories.

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, ∞

Up to 13 is called "Definite Range" and beyond 13 is called "Indefinite Range". If any story is estimated beyond 13 points, it is better to split it into multiple stories and get the estimate within "Definite range".

How to Estimate Story Points and Why?

- 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)

- **Why Story Points?**
    - Story Points help drive cross functional behavior
    - Story points estimate do not decay
    - 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

Why only Fibonacci sequence and why not 1,2,3,4,5,6,…OR 1,2,4,8,16,…?

The gaps in the Fibonacci series become appropriately larger as the numbers increase. A 1 point gap from 1 to 2 and from 2 to 3 seems appropriate just as the gaps from 3 to 5 and 5 to 8. The other two sequences is either spaced with only 1 number gap or double the gap which is not very useful.

What if you have a story that you think fall between 5 and 8 (say 7)?

You can place it in the nearest bucket (in this case 8).

Explanation about Estimation Symbols:

? : I don't understand the story enough to estimate with confidence

∞ : I understand but it will take longer than we will ever have

 : If I don't get a bio. break first, then my estimate will be whatever you want it to be

0 : It can be done quicker than it will take to have this conversation

Baseline User Story/Stories

Before the team starts the Story Point estimate, the team should first come up with a "Baseline Story" which is very clear in scope, not very small or very large but a medium size. This story is called "Baseline Story". For all the remaining stories the team can take the baseline story for comparison the relative size. It is also sometimes recommended to have more than one story as "Baseline Story" say a story with 3 story points and another story with story points 8 so when you compare the remaining stories you can compare to see if the new story is <3 or >8 or in between 3 and 8 story points.

Story points help to define the team's velocity so that it will help to forecast the future iteration's velocity and then forecast the release duration.

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.

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

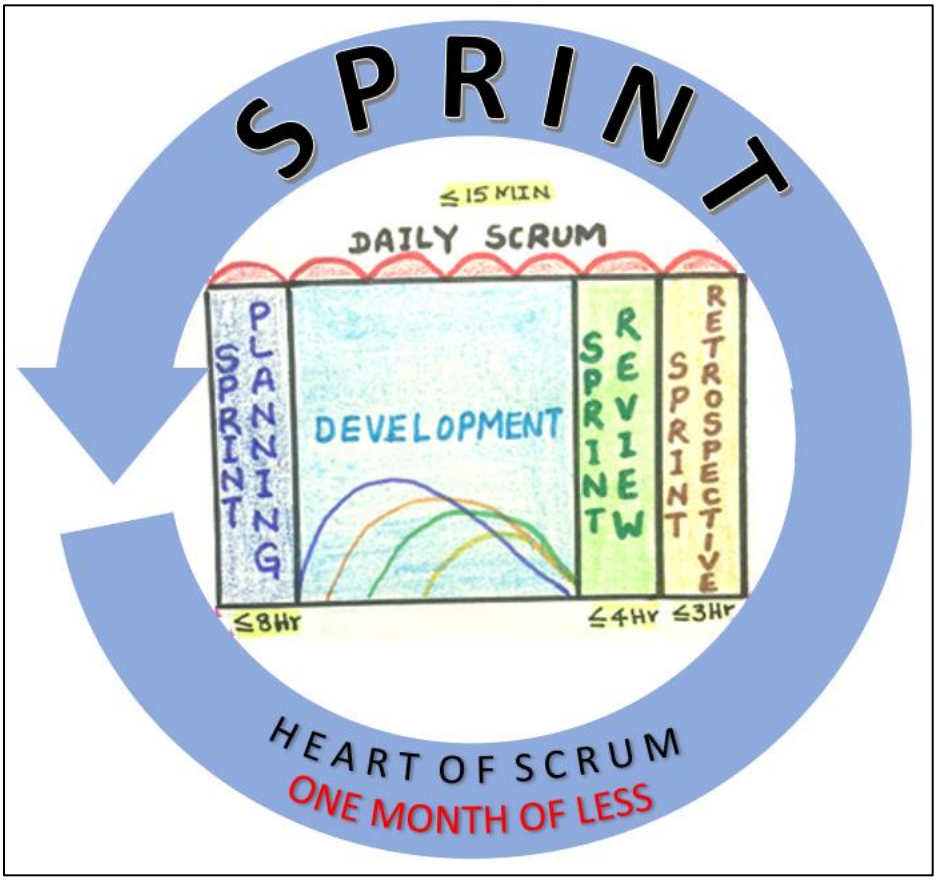
Distributed Agile teams can play the Planning Poker online for the estimation

SPACE FOR NOTES:

SCRUM

EVENTS

Scrum Events:



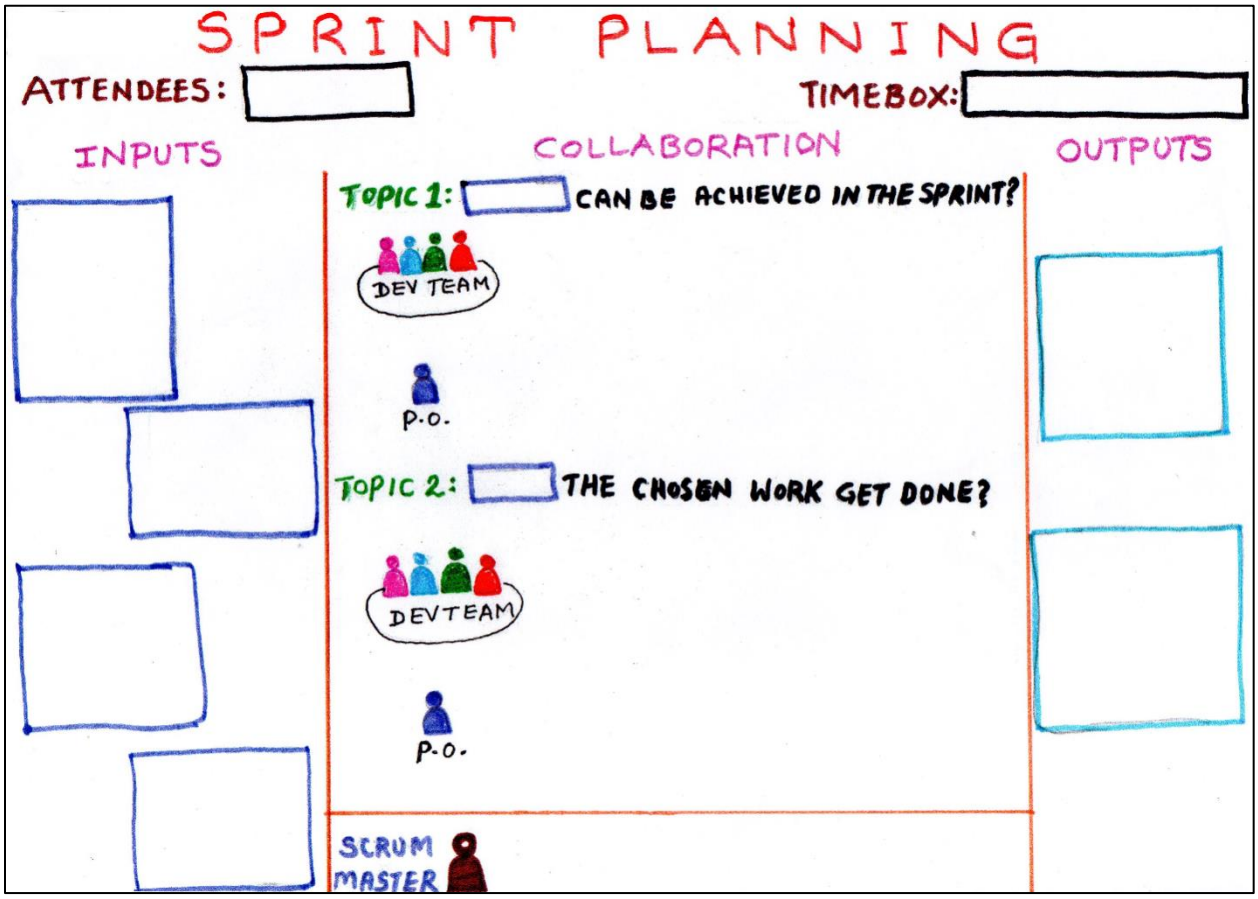
Sprint is a container of the other 4 events. Sprint itself and the 4 events are timeboxed. These vents are formal opportunities to inspect and adapt specific items; they are:

1. **Sprint Planning** to inspect the **Product Backlog** and **Plan the work** for the Sprint
2. **Daily Scrum** to inspect the **Sprint goal & Sprint Backlog** 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**

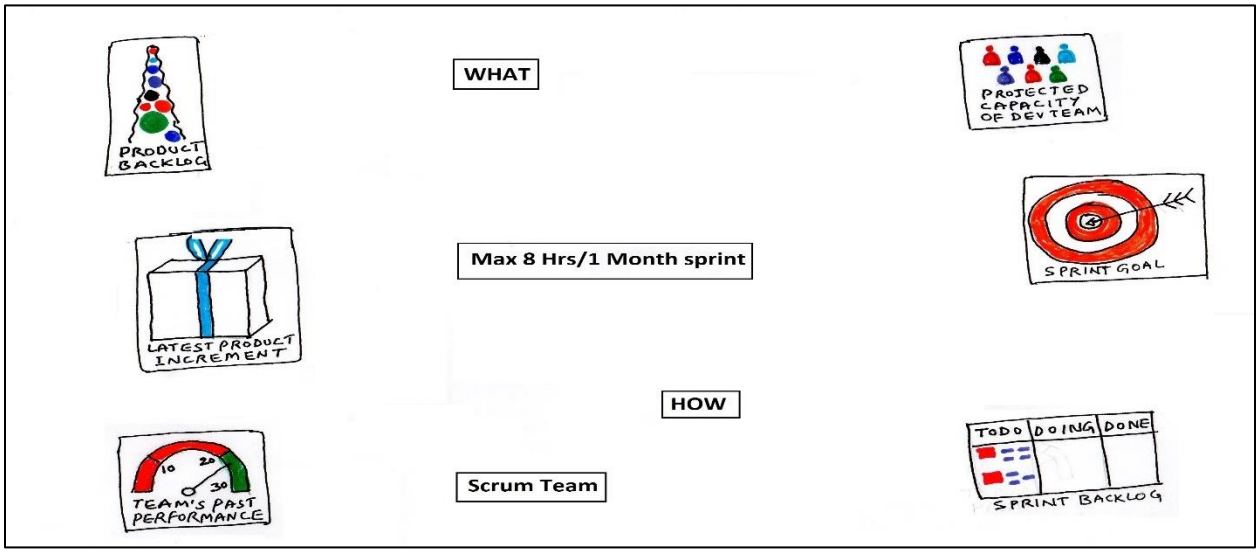
SPACE FOR NOTES:

Sprint Planning:

Fill the boxes in below diagram with appropriate items using the clues given underneath that.



Clues:



Sprint Planning

The work to be performed in the Sprint is planned at the Sprint Planning. This plan is created by the collaborative work of the entire Scrum Team.

Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. The Scrum Master teaches the Scrum Team to keep it within the time-box.

Sprint Planning answers the following:

- ***What can be delivered in the Increment resulting from the upcoming Sprint?***
- ***How will the work needed to deliver the Increment be achieved?***

Inputs for the Sprint planning: Product backlog, Latest product increment, Projected Capacity of Development team during the Sprint, and Development Team's past performance (If they use Story point estimation then this is Velocity).

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

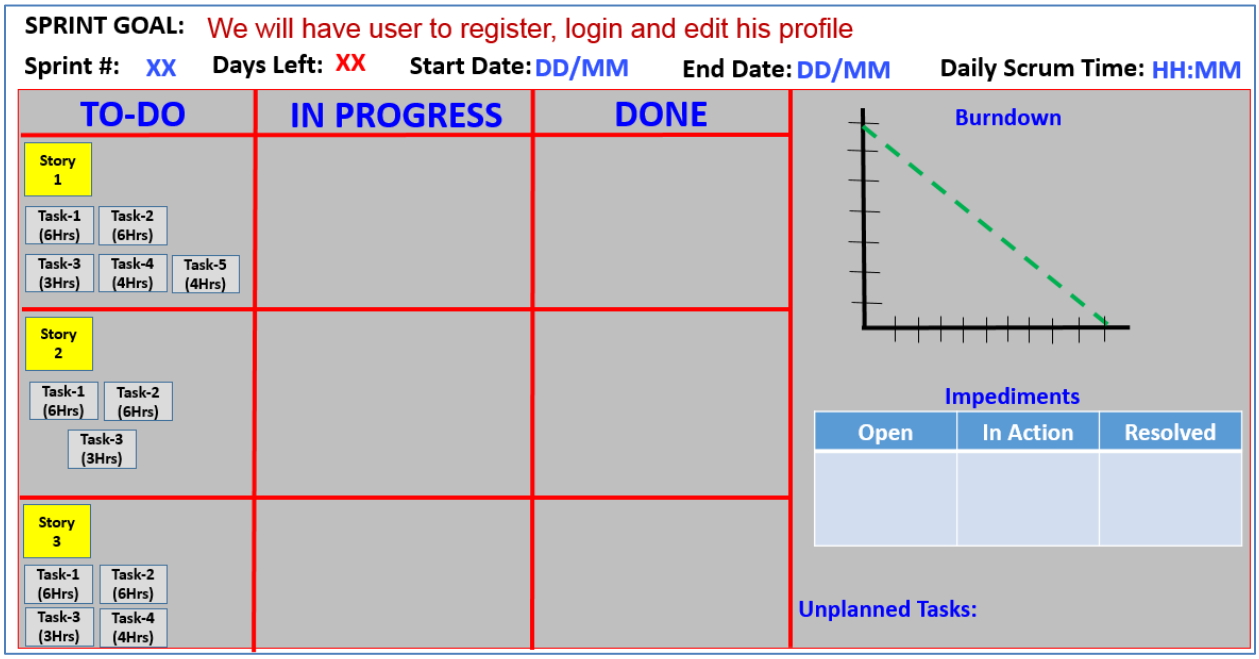
ROLE	Contribution in WHAT part of Sprint Planning
 **Product Owner**	- Discusses the tentative objective for the Sprint
 **Development Team**	- Forecasts the functionality that will be built in the Sprint - Only Development team decides how much work can be pulled into Sprint - No one can force the Development team on the selection of amount of work to be pulled into the Sprint (# of Product Backlog Items)
 **Scrum Team**	- The entire Scrum team collaborates to understand the work of the Sprint - Crafts the Sprint Goal (one of the two outputs of the Sprint Planning) - 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 WHAT 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. - 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. - The Development team usually starts the design the system for the increment that they create in the Sprint. - 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. - 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. - 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. - 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.
 **Product Owner**	- Product owner can clarify the selected product backlog items and make trade-offs
 **Scrum Master**	- Ensures the event takes place - Makes sure attendees understand the purpose of Sprint planning - Teaches the Scrum team to keep it within the timebox

SPACE FOR NOTES:

At the end of the sprint planning, the sprint backlog may look like below (this is a sample only)



SPACE FOR NOTES:

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

Sprint goal should be clear, measurable, achievable. Product increment created with a clear Sprint goal is a step towards meeting the Vision or goal. Sprint goal helps Product owner to decide on whether a sprint must be cancelled before the timebox expires. If the Sprint goal becomes obsolete, then the Product owner and Development team collaborates, and the Product owner makes the call. Development team uses Sprint goal in the Daily Scrum to see the progress of the Sprint. Sprint goal is the outcome that will be achieved by implementing the product backlog items selected for the sprint, which is the output. The Sprint goal can be used in Sprint review to measure the success of the Sprint.

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***

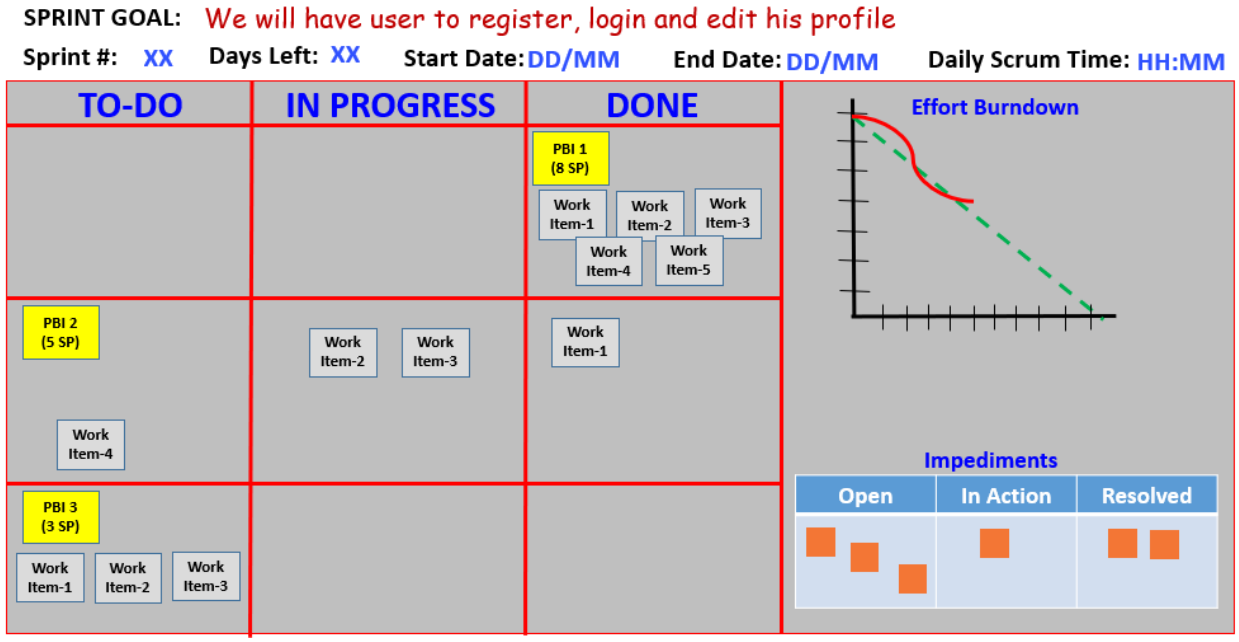
Above Sprint goals are not clear, and they do not help Development team to work as a single team, with focus. They also do not provide “Why” the increment is built. They do not provide any flexibility in terms of the functionality.

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.

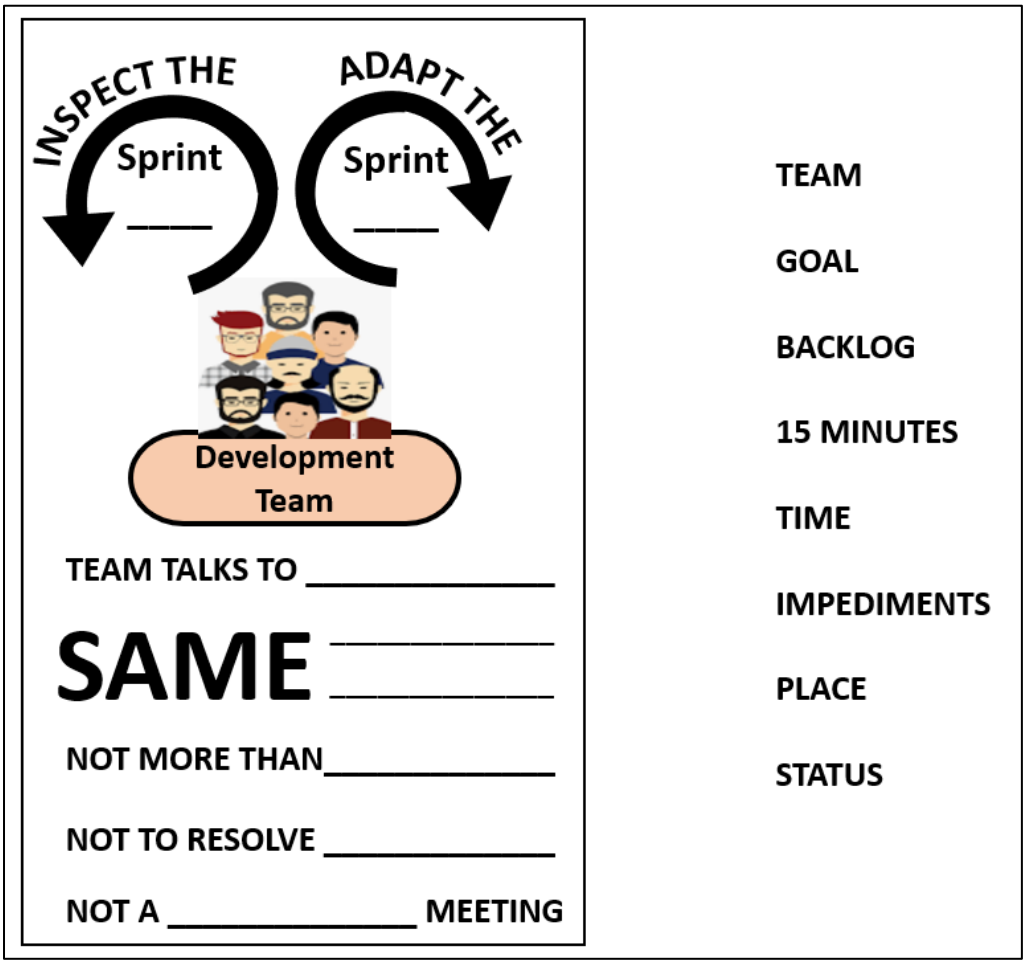
SPACE FOR NOTES:

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



SPACE FOR NOTES:

Daily Scrum



Map the right-side points to the blanks on the left side picture. (Answers are available at Appendix)

The Daily Scrum is a key inspect and adapt meeting in Scrum which is note more than 15-minute time-boxed event. In this meeting the Development Team synchronizes activities and creates an action plan for the next 24 hours. It should be held every working day of the Sprint. Development team inspects the progress towards Sprint Goal and adapt the Sprint backlog with plan for the next 24 hours. All the Development team members participate in the Daily Scrum meeting. Development team decides the structure of the Daily Scrum. Sometimes it may be a question-based conversation or sometimes it may be a discussion oriented. If they use question-based approach, then every member of the Development team covers the following questions. Who will start does not matter, anyone can start. It need not to go in a particular order also, whatever approach the Development team feels right, they will take that.

- ✓ **What did I do yesterday that helped the Development Team meet the Sprint Goal?**
- ✓ **What will I do today to help the Development Team meet the Sprint Goal?**

- ✓ **Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?**

If there is a need for further discussion or to discuss about impediments resolution, the Development team can meet immediately after the Daily Scrum.

Scrum master ensures this meeting happens and participants understand it's purpose and also teaches them how to keep it within the timebox of maximum 15 minutes. The Development team is responsible to have this meeting even though the Scrum master is not available. This is the Development team's internal meeting so if any others not part of the team attends, Scrum master ensures that they will not interrupt the Development team during the meeting.

Product owner is optional for Daily Scrum but if Development team wants the Product owner to be part of the Daily Scrum, then the Product owner joins.

Purpose of Daily standup is NOT for status update, it is for Planning, Knowledge sharing & to highlight the impediments

Status is a byproduct of it

Advantages of Daily Scrum Meeting:

- Improves the communication and collaboration within Development team
- Sets the day's direction
- Eliminate other meetings
- To identify any obstacles, the team comes across
- Quick decision making
- Improves Development team's level of knowledge
- Collectively re-commit to the project as a team
- Encourages self-organization

SPACE FOR NOTES:

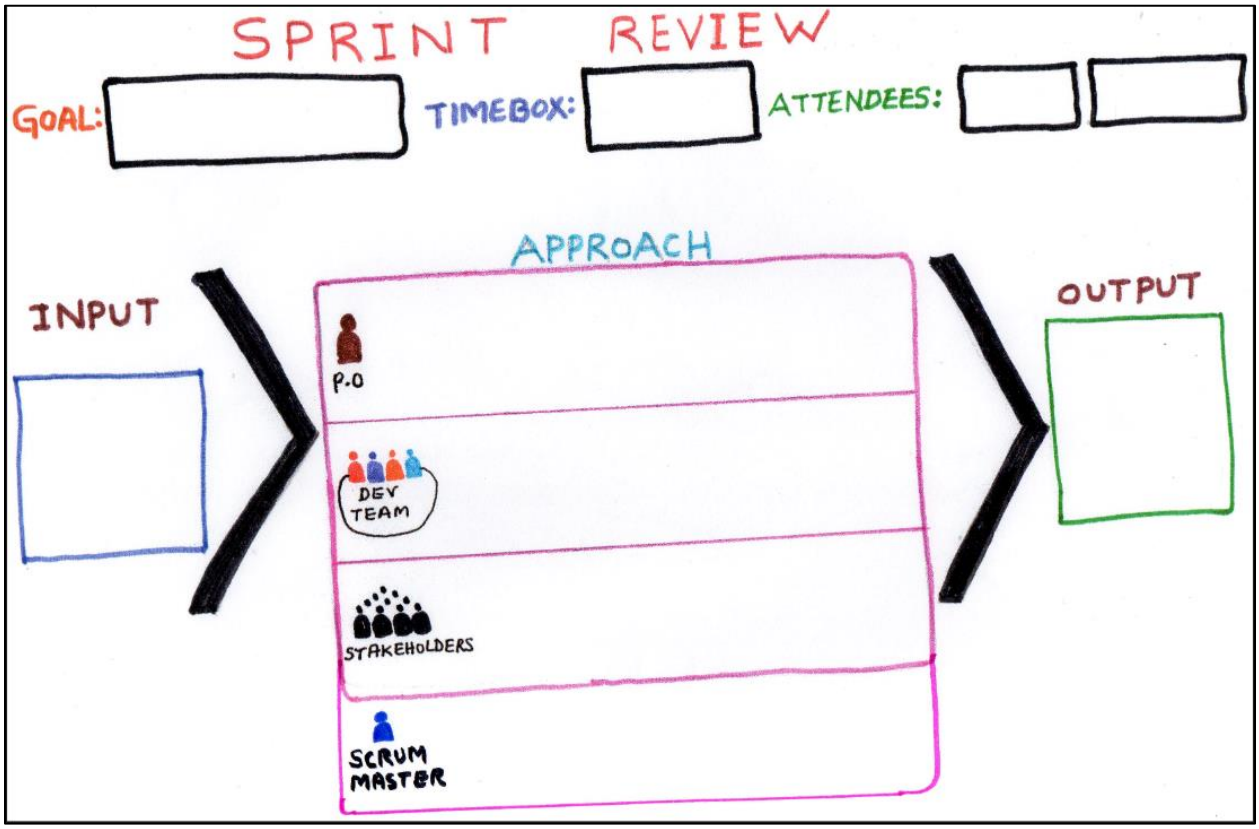
What makes Daily Scrum effective

Note: These are some general recommendations and not prescribed by Scrum Guide.

- ❑ Do not get into prolonged discussions as it may end up taking longer than 15 minutes.
- ❑ 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.
- ❑ All Development team members are mandatory for the daily Scrum.
- ❑ Team members should be punctual for the meeting and it should start on time so that waiting can be avoided. Always 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.
- ❑ Team speaks to team and it's not an update to Scrum Master.

SPACE FOR NOTES:

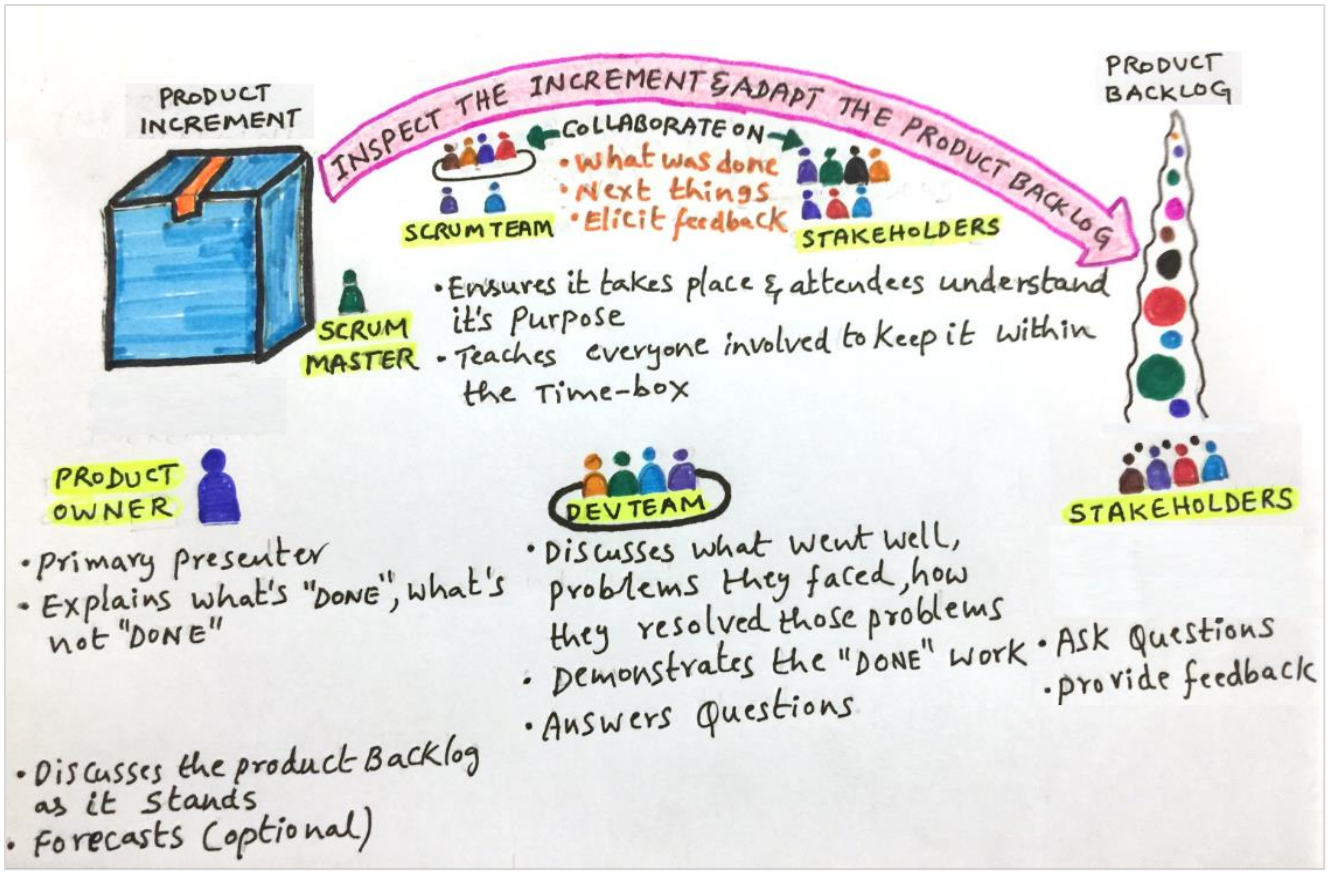
Sprint Review



Map the below items to the appropriate boxes in the above diagram to complete Sprint Review.

MAX 4 Hours for one Month Sprint **Product Increment** **Stakeholders**

Scrum Team **Elicit Feedback** **Updated Product Backlog**

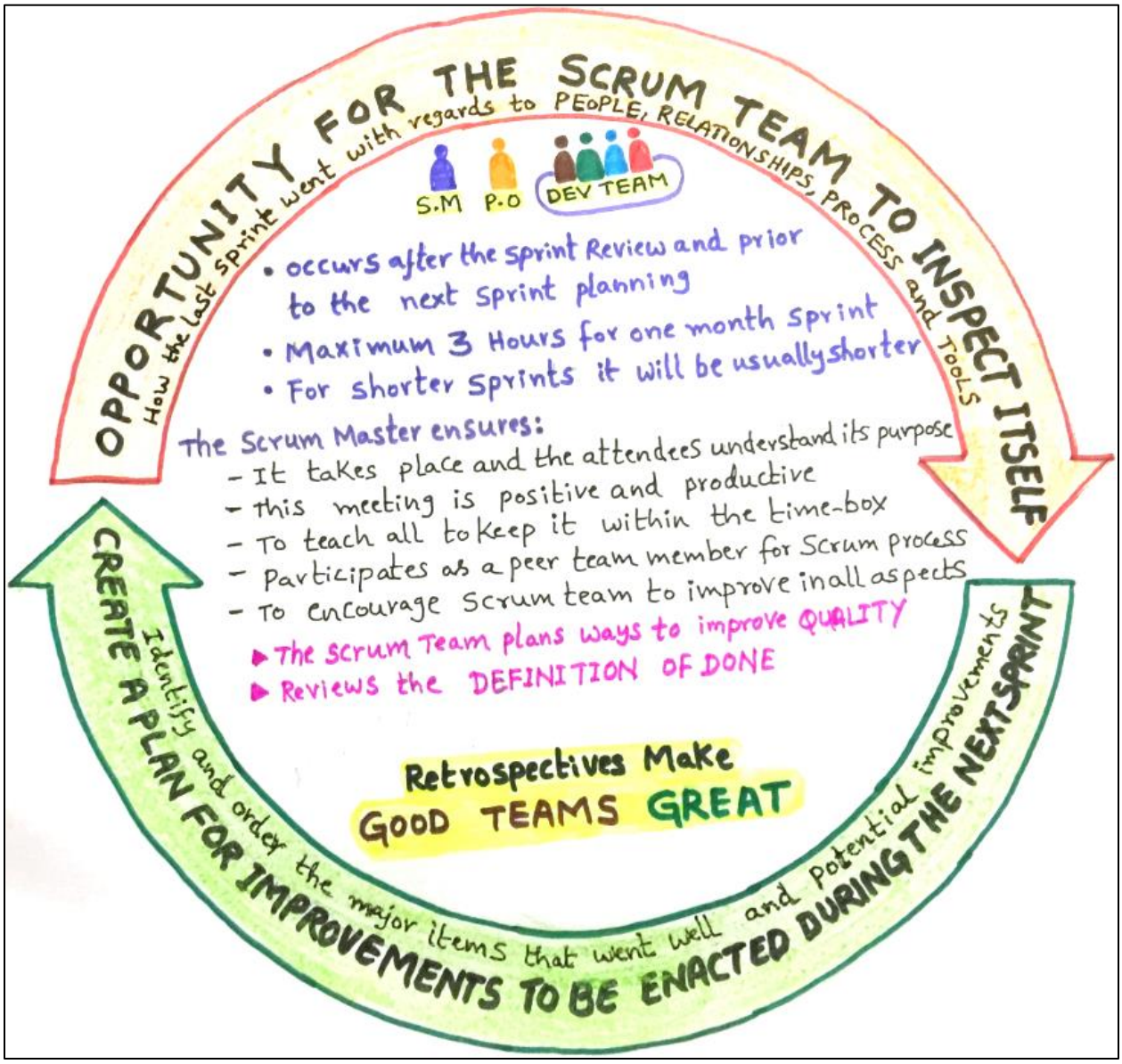


Sprint Review is held at the end of the Sprint before the Sprint Retrospective. Objective is to inspect the product increment and adapt the product backlog based on the need (inputs received from the stakeholders based on the review). During the Sprint review, The Scrum Team and stakeholders collaborate about what was done in the sprint. This is an informal meeting and NOT a status meeting. The presentation (demo) is to elicit feedback on the Product increment and foster collaboration.

Duration of the "Sprint Review Meeting" is *4 hours* for one-month sprint, usually shorter for shorter Sprints. All the participants will collectively collaborate on what to do next to add value, so it will be a key input for next sprint planning. Review of timelines, budget, potential capabilities, market place for the next anticipated release of the product

SPACE FOR NOTES:

Sprint Retrospective



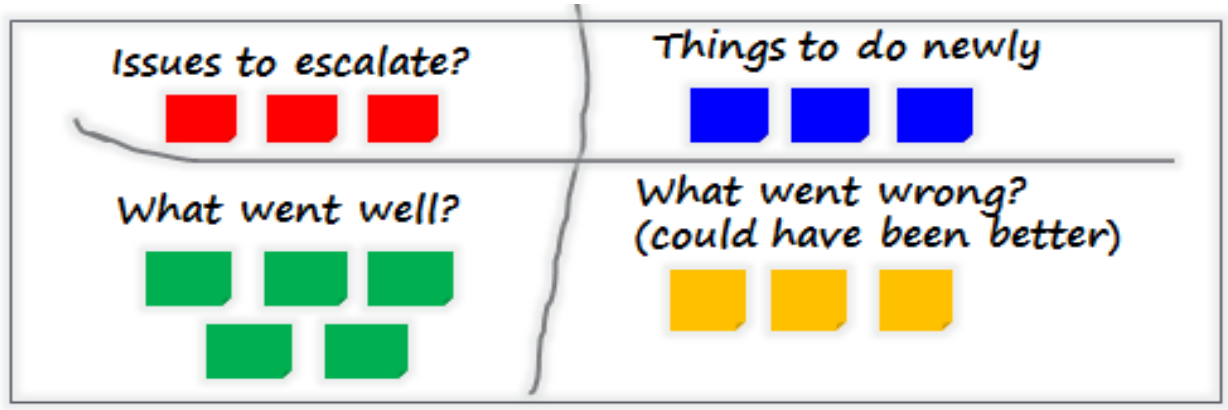
Scrum Master can help Development team to prioritize the improvements identified using some facilitation techniques (example: DOT voting). The improvements identified will be added to the next Sprint backlog so that team will have focus in implementing them. Product owner also provides his feedback to the team and takes feedback from the team at retrospective. This is not a fault finding and blame game. This is an inspect and adapt opportunity for Scrum team to get better.

SPACE FOR NOTES:

Retrospective stages: (Not prescribed Scrum but effective practice)

Set the stage	• Lay the groundwork for the session by reviewing the goal and agenda. • Create an environment for participation by checking in and establishing working agreements.
Gather data	• Review objective and subjective information to create a shared picture. • 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. • 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:



Mike Cohn's approach for Retrospective:

Team is asked on things that they should :

- STOP Doing
- START Doing
- CONTINUE Doing

SPACE FOR NOTES:

Visible Progress through Burndown Chart *(Not prescribed by Scrum)*

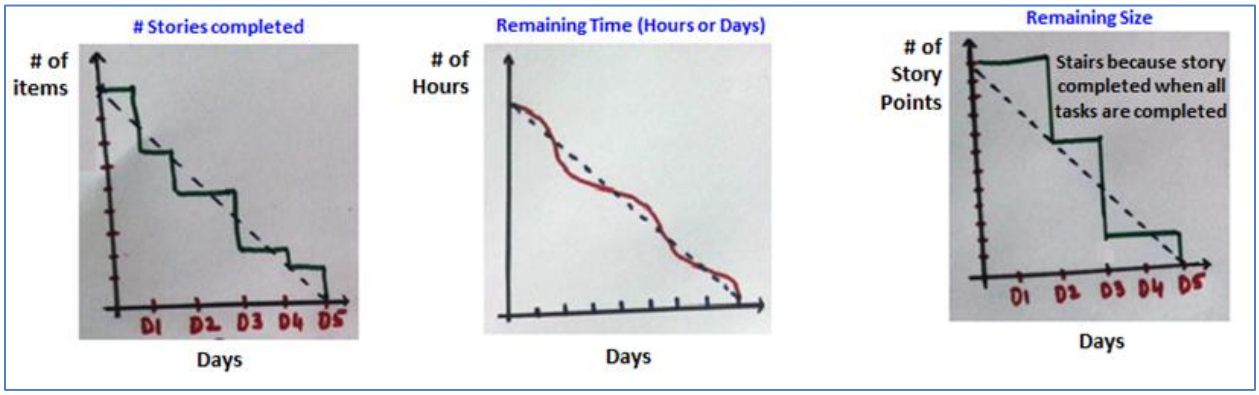
Burndown chart is the tool to show the work remaining in a Sprint. Team updates this as and when the work gets completed. That means whenever a product backlog item meets the "DoD" corresponding size of the backlog item (story points) will be updated on the chart. This helps Development team to track 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.

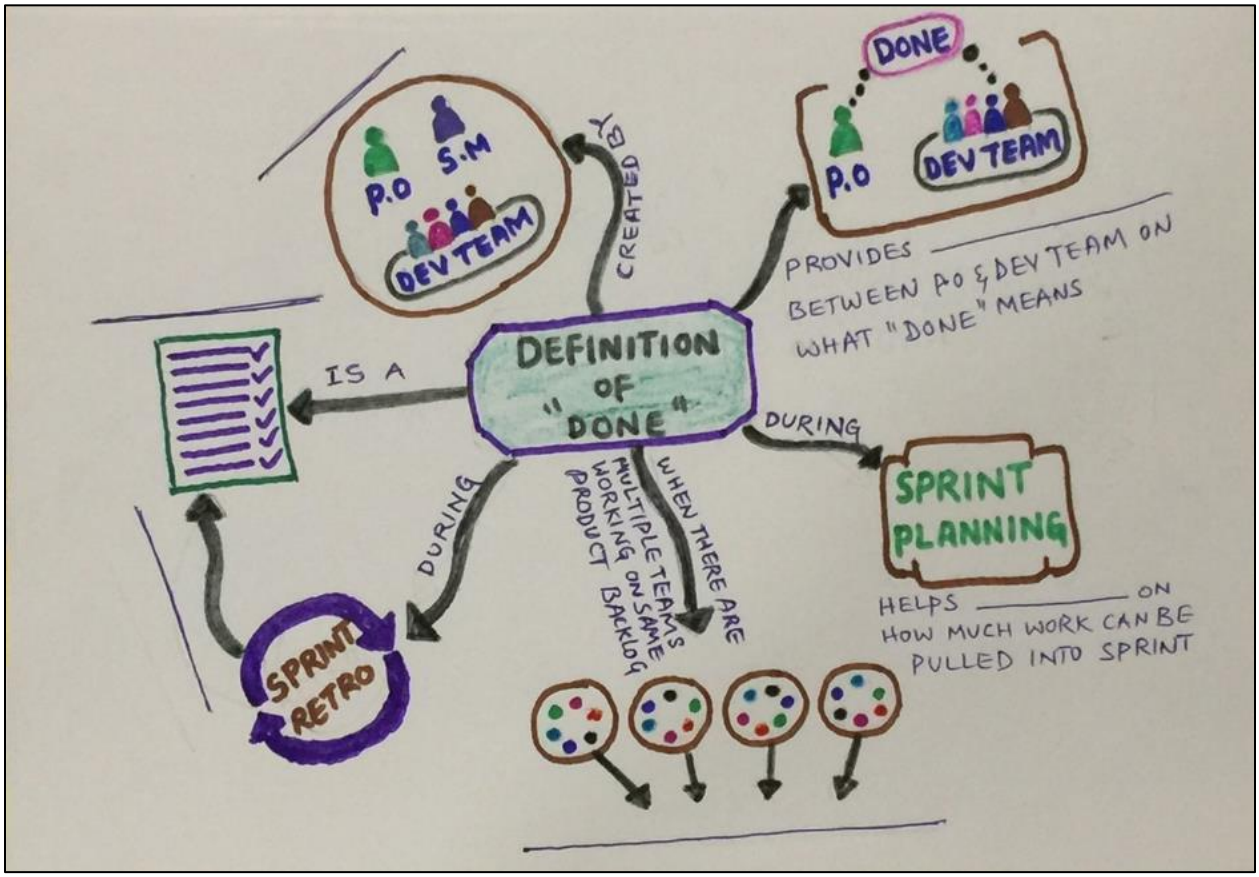
Key points related to Velocity:

- 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

SPACE FOR NOTES:

DEFINITION OF DONE

Definition of Done (DoD)



Fill the below clues into appropriate blanks in the above diagram: ***(Answers are in the Appendix)***

There should be a mutually agreed DOD | Shared understanding | Quality criteria | Scrum Team | Review and strengthen | Development Team

When a Product backlog item or an Increment is described as “Done” everyone (especially Product owner and the Development team) what “Done” means. It may be different for different Scrum teams. The Definition of Done will help to assess when the work is “Complete”. It is a quality criteria defined by Scrum team collaboratively and it will be created before the first Sprint. If the organization has certain quality standards defined, all teams must follow that as minimum Definition of Done. If there is no such standard at organization level, then each Scrum team should define a Definition of Done.

Development team owns the Definition of Done and they review and strengthen it during Sprint Retrospective. If there are multiple teams working on same Product Backlog, all teams should mutually define the Definition of done. Definition of Done helps the Development team to decide how much work can be pulled into the Sprint.

DoD vs Acceptance Criteria:

Here are the jumbled differences between Acceptance Criteria and Definition of done. Arrange them into below table as per the column headings.

Owned by Development Team | Related to functionality | Related to Quality |

Owned by Product owner | Defined at any time | Does not change usually |

Usually defined as soon as possible before sprinting |

Different for every Product backlog item | Enhanced through periodic inspect and adapt

| Applies to entire Product backlog or Increment |

ACCEPTANCE CRITERIA	DEFINITION OF DONE

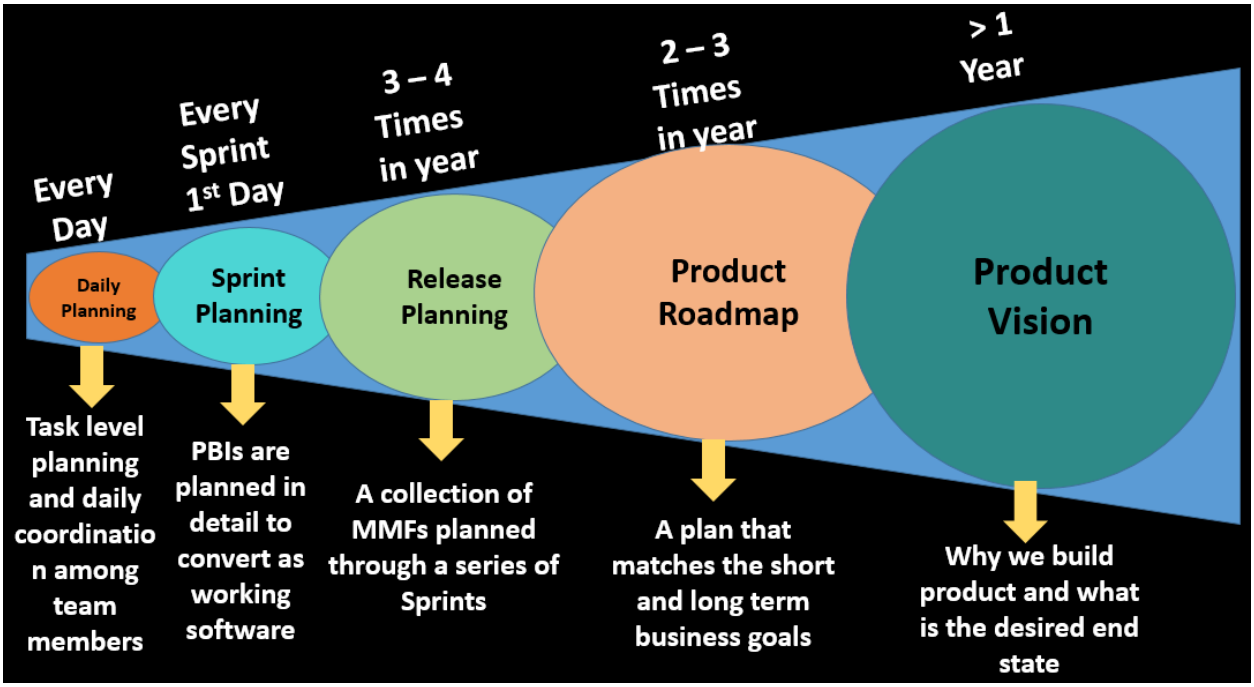
SPACE FOR NOTES:

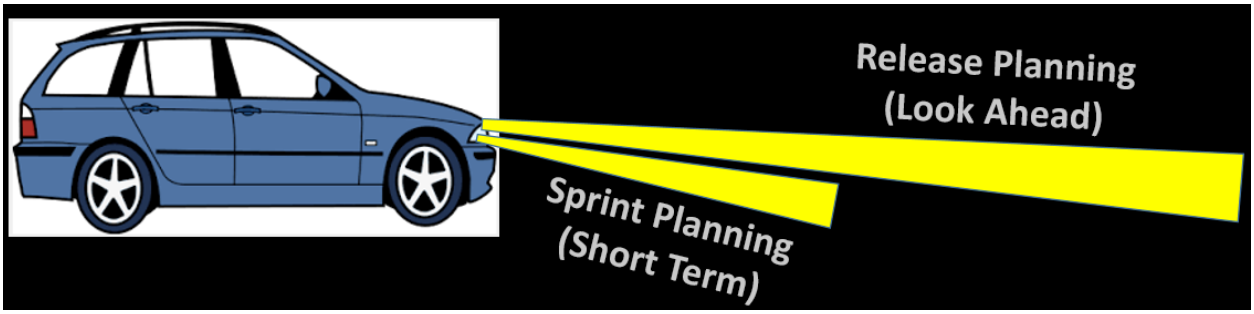
RELEASE PLANNING

(Not part of Scrum)

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:

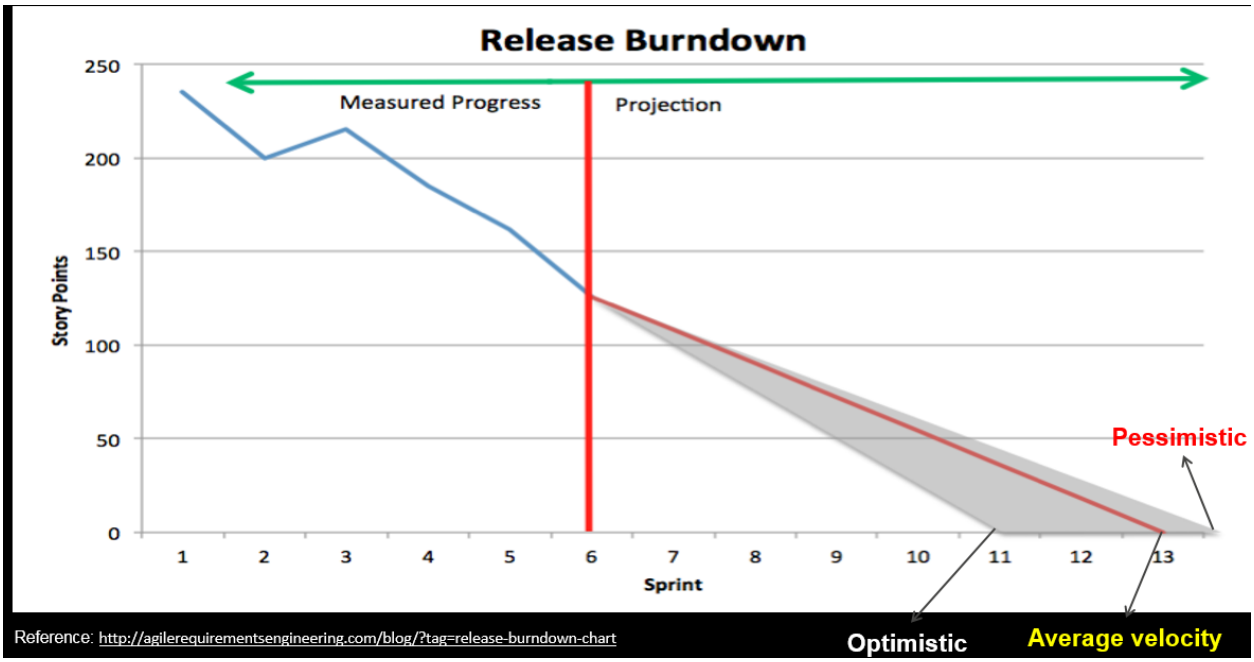
1. Release every 6 weeks

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	

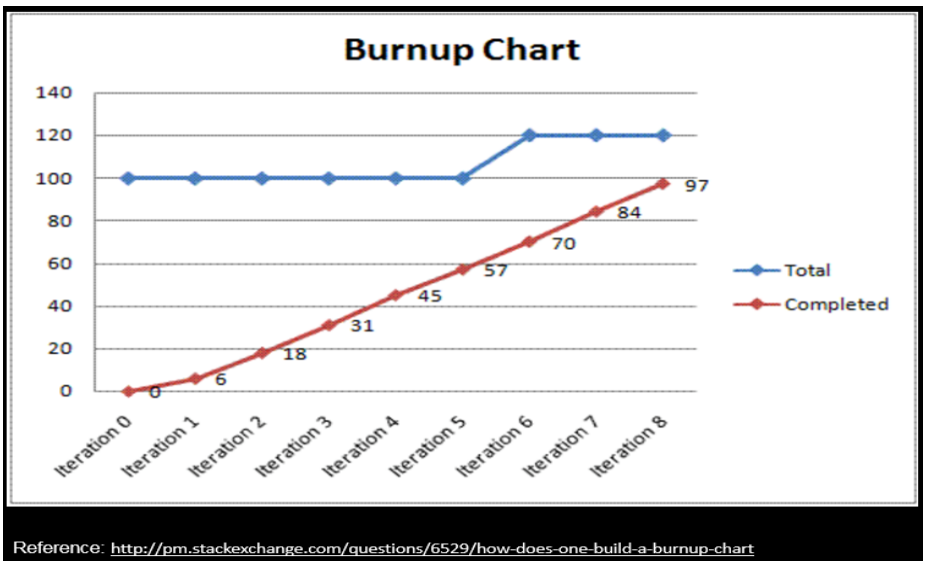
- The duration of Release Planning meeting will vary depending on sprint length. As a general rule of thumb, the team should allocate 5 to 15 percent of the team's total available hours to this meeting early in projects. As the project matures, this time may drop to as low as four hours per sprint.

- 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:



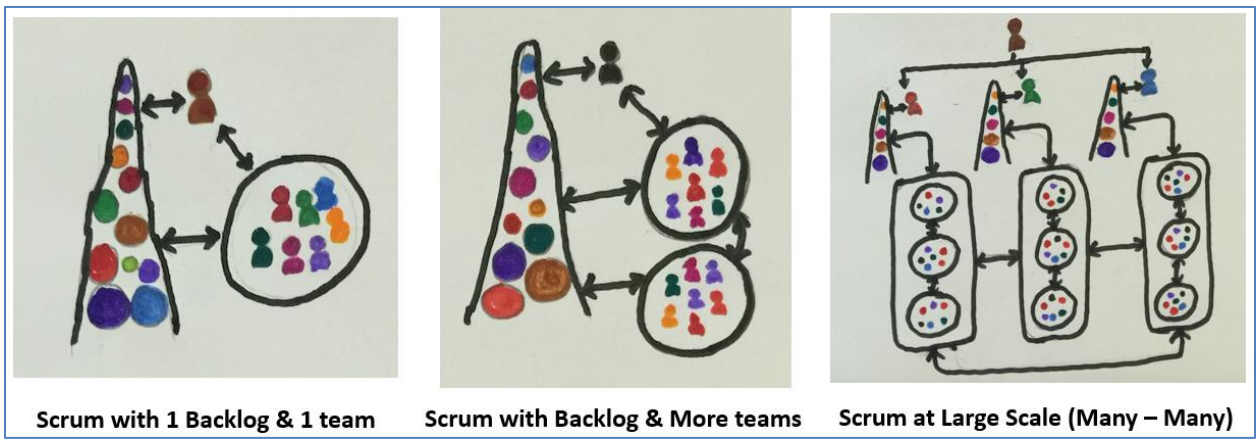
Blue line indicates the overall release backlog size. Till the 5th Sprint, it is same but in Sprint 6 it has increased, that means some new work is added. Red line indicates the cumulative completion of the work in every Sprint. So, at the 8th Sprint, "Remaining work to be complete" can be found by subtracting 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:

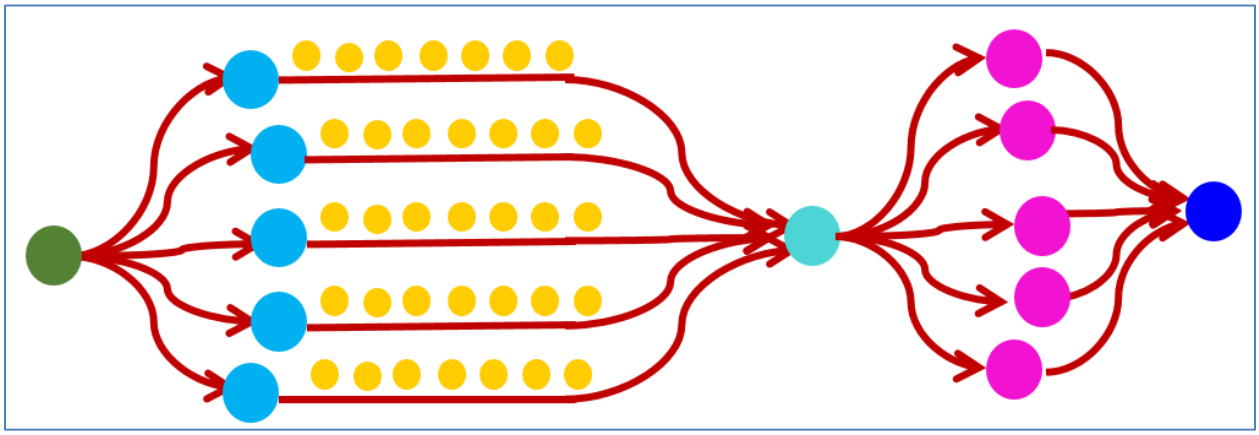
Scaling Scrum (Not part of Scrum)

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.

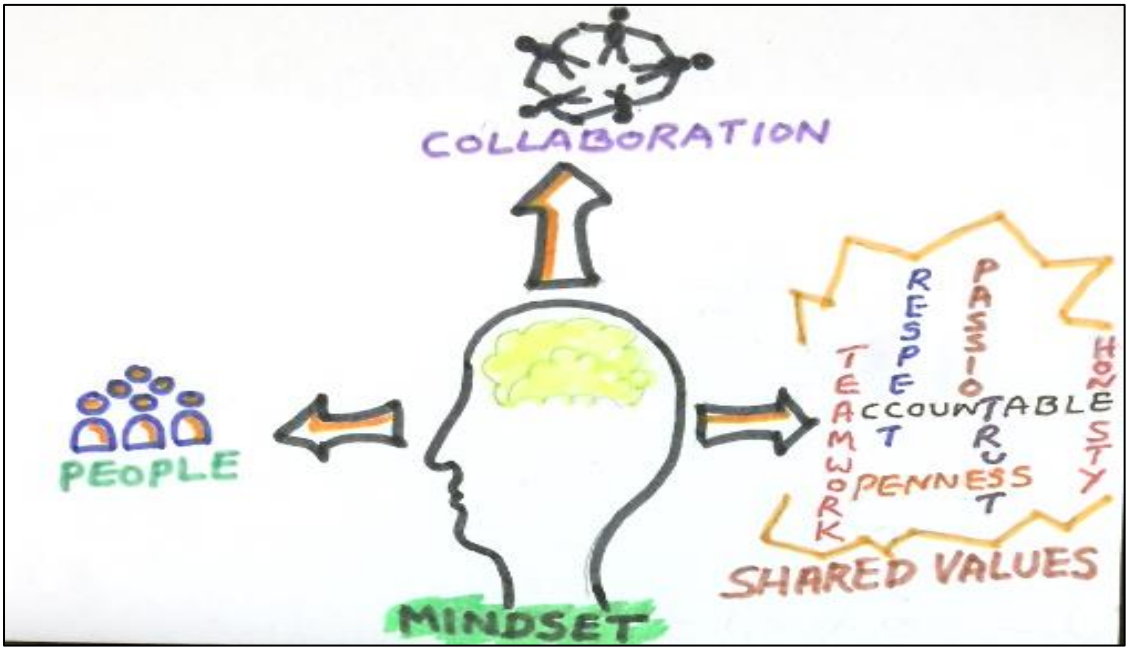
- **Coordination:** Just Talk, Communicate in Code, Travelers, Open Space, and Communities.
- **Overall PBR:** There may be an optional and short overall Product Backlog Refinement (PBR) meeting that includes the one Product Owner and people from all teams. The key purpose is to decide which teams are 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

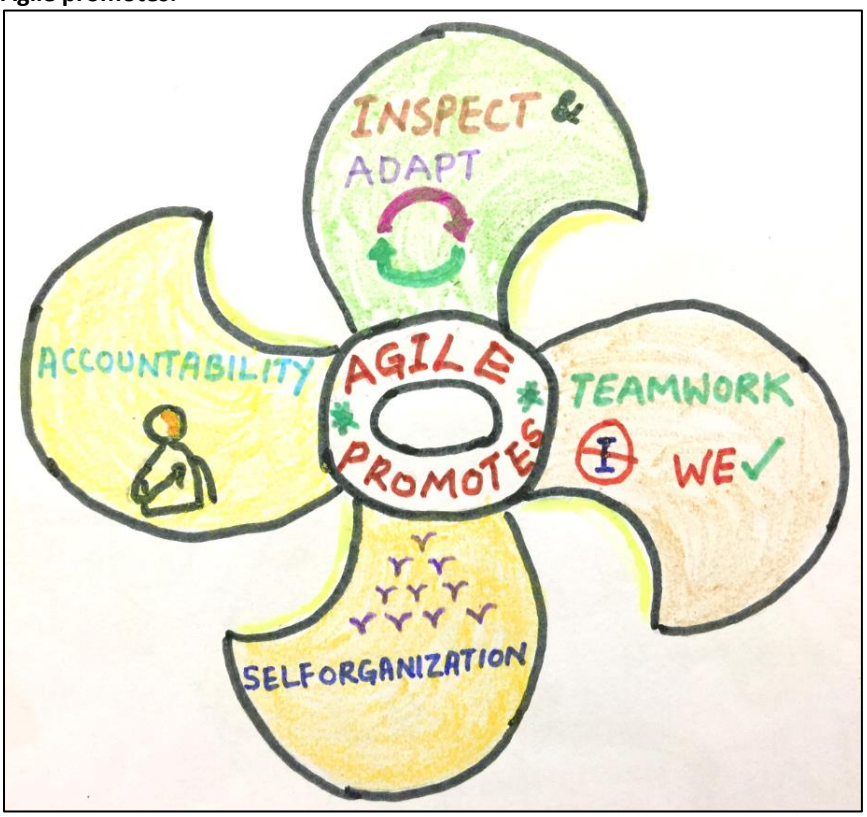
APPENDIX – 1: ANSWERS FOR EXERCISES

1. **Exercise – What is Agile?**



2. **Agile focuses on:**

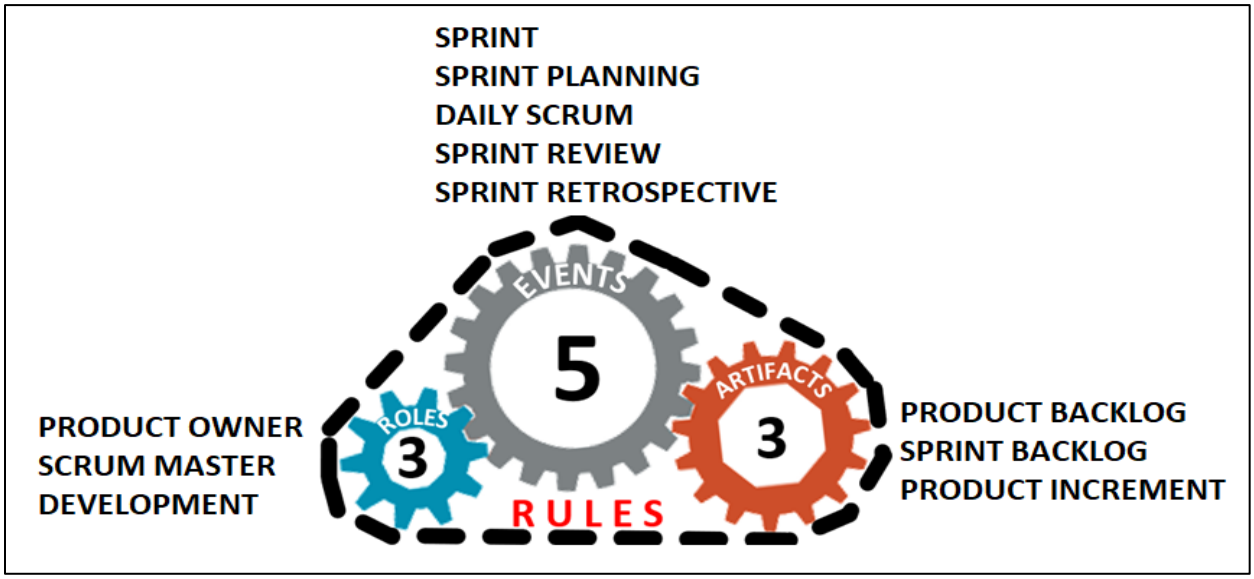
3. **Agile promotes:**



4. **What is Agile and what is not?**

AGILE IS ….	AGILE IS NOT ….
✓ A disciplined and structural approach to iterative development	× Getting the same amount of work in less time and cost
✓ A set of practices that accommodate changing requirements	× A way for the business to change it's direction on the fly
✓ A set of practices that will help improving the quality	× A project management methodology
✓ An umbrella of different iterative and incremental software development frameworks	× A process

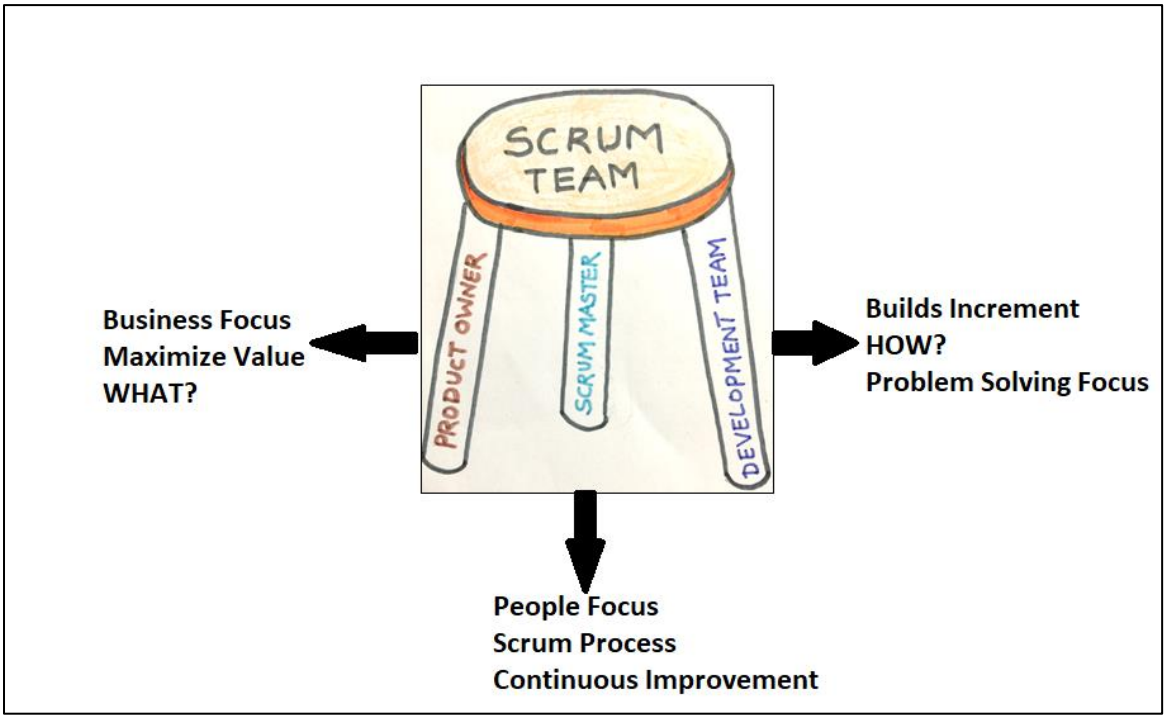
5. **Exercise – Scrum framework**



6. **Exercise – Sprint**

Fill the below blanks with the numbers of the phrase on the second column	
A basic unit of ---------**7**-------- is called as "Sprint"	1 heart
Team produces ----**6**--- at the end of every Sprint	2 changes
Each sprint will have specific --**13**-	3 needs to be built
Each Sprint is --**10**--- (i.e Can only take allowed max duration, 1 to 4 weeks)	4 fixed
No --**9**--- between sprints	5 changed during
Sprint is the -----**1**---- of Scrum	6 PSPI
All the events of the ---**12**--- are timeboxed	7 development in Scrum
Each sprint has a definition of what ---**3**---, a design and a plan	8 goal are accepted
A new sprint ------**14**------ after previous sprint	9 gap
The duration of sprint is ----**4**------	10 time boxed
No changes that impact the sprint ------**8**------- during the sprint	11 predictability
Same duration for all sprints gives better --------**11**-------	12 sprint
Quality goals cannot be ---**5**---- the sprint	13 events
No ---**2**------ in team composition	14 starts immediately

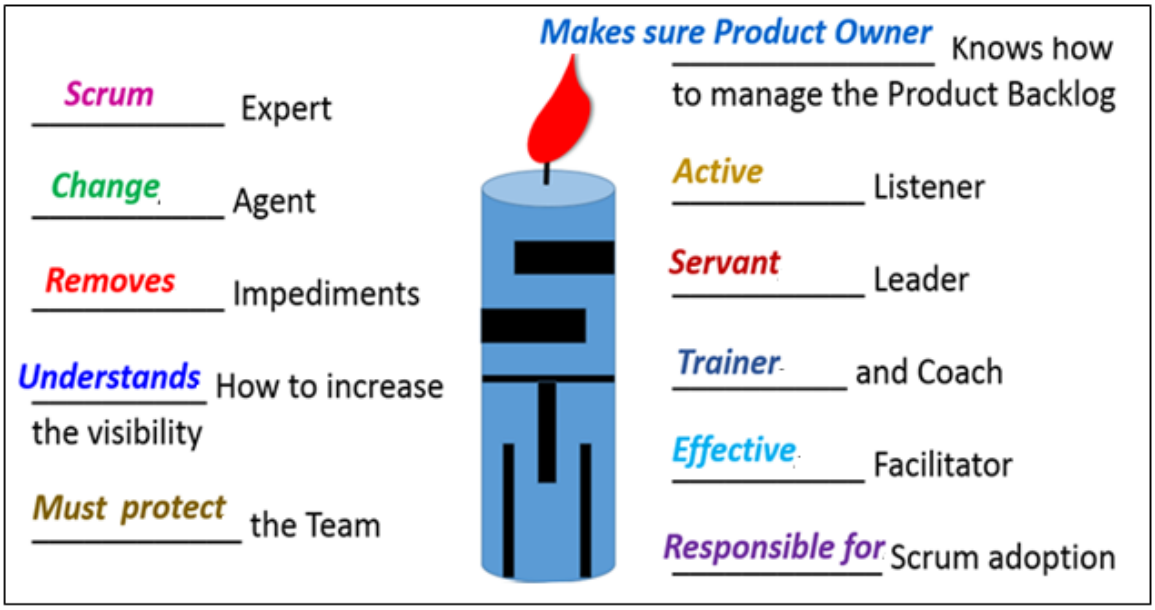
7. Exercise – Scrum Team



8. Exercise – *PO Role Recap:*

Statement Related to Product Owner Role	Myth/Fact
The P. O. must tell the Dev Team members how they must do their work	MYTH
The P. O. should decide the release dates of the product	FACT
The P. O. must be the project customer himself	MYTH
The P. O. is the only one that can change priority of the Product Backlog items	FACT
The best P. O.'s already worked as Project Managers	MYTH
The P. O. must balance the needs and desires of the different project stakeholders	FACT
The P. O. must involve in daily scrum to guide the Dev team on their work	MYTH
The P. O. and the Scrum Master must never be the same person	FACT
The presence of the P. O. is not mandatory at the Sprint Planning meeting	MYTH
To be more effective in the role, the P. O. must collaborate closely with the Dev Team	FACT
The P.O has shared responsibility for ROI with Development team	MYTH
The P. O. must not estimate Product Backlog items along with the Dev Team	FACT
The P.O can be a committee that collectively shares the P.O responsibilities	MYTH
The P. O. is responsible for defining which is the right product to be developed	FACT

9. **Exercise – Scrum Master**



10. **Exercise – Scrum Master services to Organization, Product Owner and Development team**

Scrum Master Service to the Product Owner (not limited to):

- Finding techniques for effective Product Backlog management;
- Clearly communicating vision, goals, and Product Backlog items to the Development Team;
- Teaching the Scrum Team to create clear and concise Product Backlog items;
- Understanding long-term product planning in an empirical environment;
- Understanding and practicing agility; and,
- Facilitating Scrum events as requested or needed.

Scrum Master Service to the Development Team (not limited to):

- Coaching the Development Team in self-organization and cross-functionality;
- Teaching and leading the Development Team to create high-value products;
- Removing impediments to the Development Team's progress;
- Facilitating Scrum events as requested or needed; and,
- Coaching the Development Team in organizational environments in which Scrum is not yet fully adopted and understood.

Scrum Master Service to the Organization:

The Scrum Master serves the organization in several ways, including:

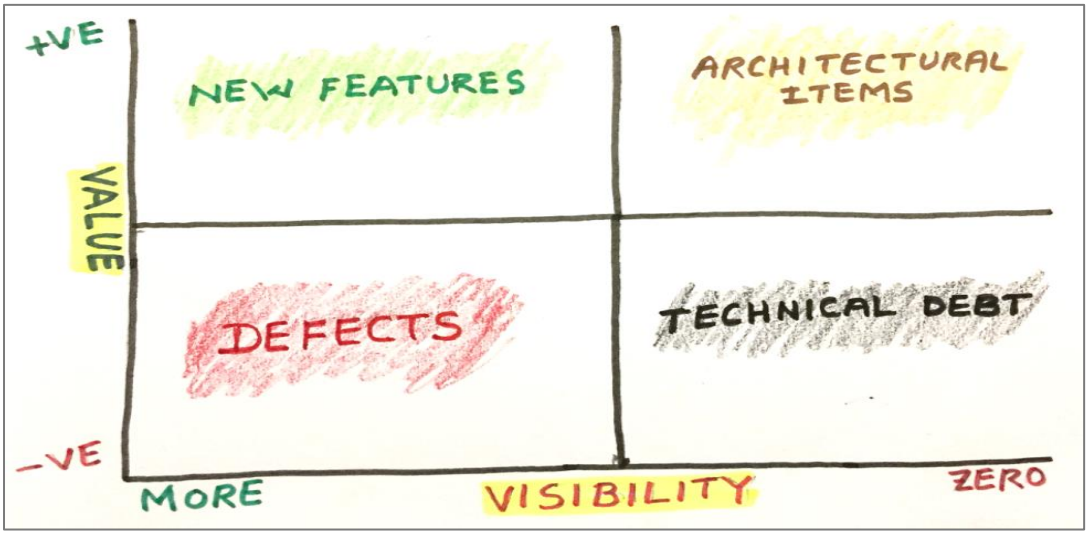
- Leading and coaching the organization in its Scrum adoption;
- Planning Scrum implementations within the organization;
- Helping employees and stakeholders understand and enact Scrum and empirical product development;
- Causing change that increases the productivity of the Scrum Team; and,

Working with other Scrum Masters to increase the effectiveness of the application of Scrum in the organization.

11. **Exercise –*Scrum Master Role recap:***

Statement Related to Scrum Master Role	Myth/Fact
The best Scrum Masters already worked as a Dev Team member	MYTH
The Scrum Master must have courage to do his job	FACT
The best Scrum Masters already worked as Project Managers	MYTH
The Scrum Master is the most important role in a Scrum project	MYTH
The Scrum Master must have good interpersonal skills	FACT
The Scrum Master is one of the decision makers for the project schedule	MYTH
The Scrum Master and the Dev Team estimate the Product Backlog items	MYTH
The Scrum Master must protect the Dev Team from external interference	FACT
The Scrum Master usually cannot change the organization she works for	MYTH
The Scrum Master is responsible for ensuring that Scrum is correctly used	FACT
Ideally Scrum Masters should work full time in this role to obtain effective results	FACT
The Scrum Master must manage the work of the Dev Team	MYTH

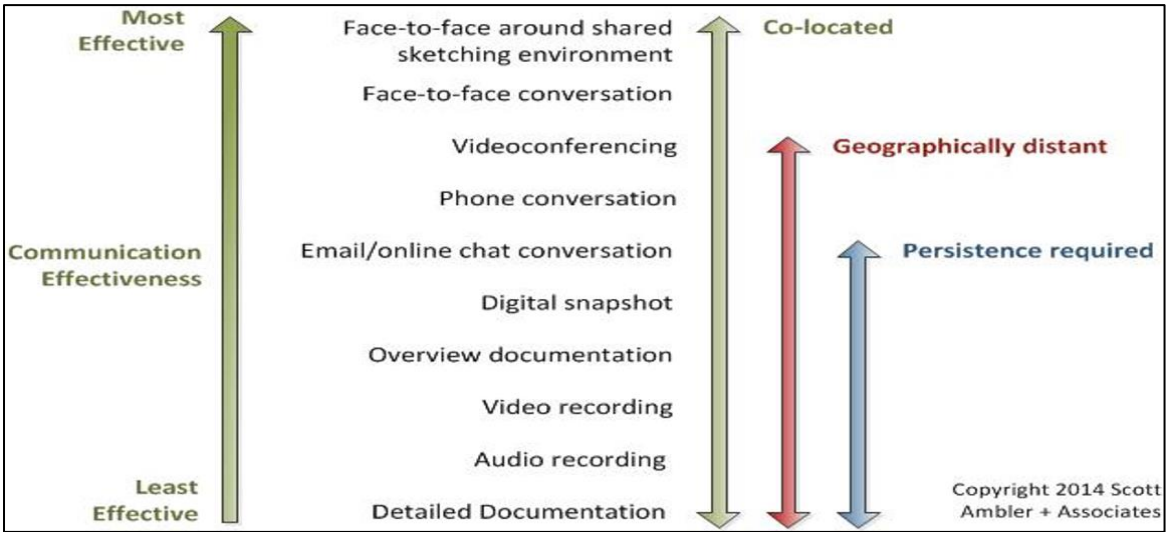
12. **Exercise – Technical debt quadrant**



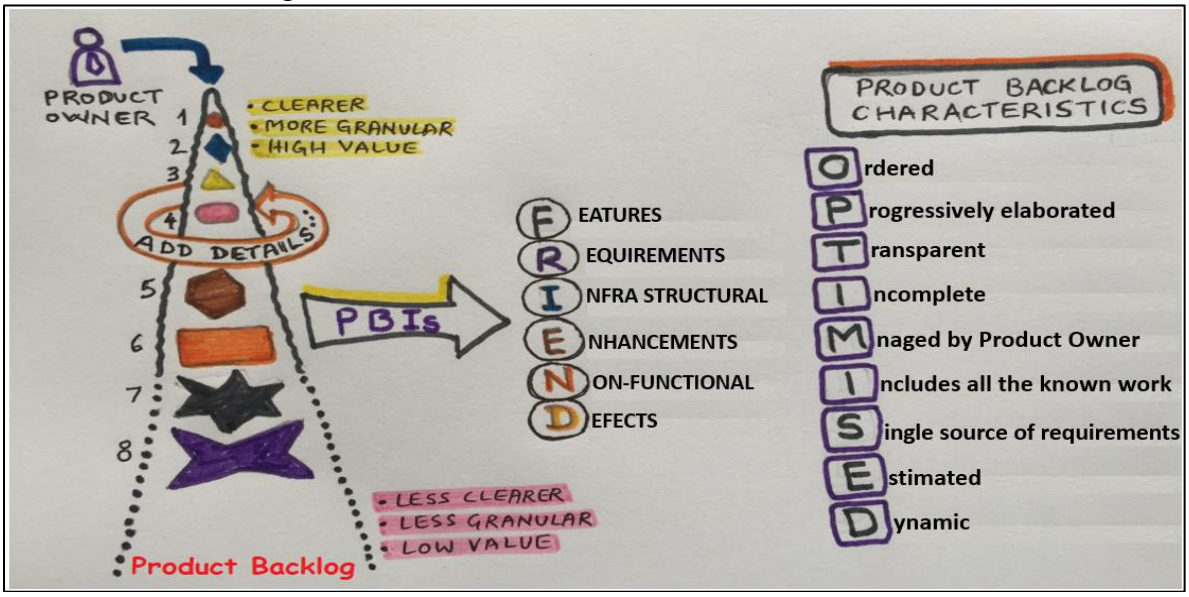
13. **Exercise –*Development Team Role recap:***

Statement Related to Development Team Role	Myth/Fact (M/F)
The Dev Team is responsible for the quality of the product increment it generates	FACT
Very small Dev Teams may not take benefit from using Scrum	FACT
Only Development team can decide how much work can be pulled into the Sprint	FACT
The Dev Team must plan themselves and manage its work on a Sprint	FACT
The Dev Team must have all skills needed to generate the product increment	FACT
Ideally, the Dev Team size should be between 3 and 9 members	FACT
The Dev Team must inform the impediments to the Scrum Master at the Daily Scrum only	MYTH
The Dev Team must never interact directly with the project customer	MYTH
The Dev Team technical leader dictates the best technical approaches	MYTH
The Dev Team must keep the P. O. out of their way during the Sprint	MYTH
The best Dev Teams are able to work in several projects at the same time	MYTH
The Dev team can extend Sprint duration if they cannot complete the work of the sprint	MYTH

14. Exercise – Communication Effectiveness

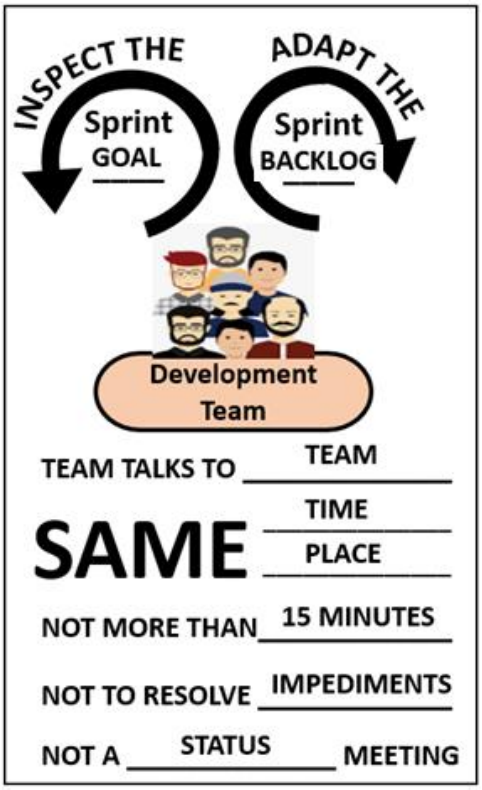


15. Exercise – Product Backlog

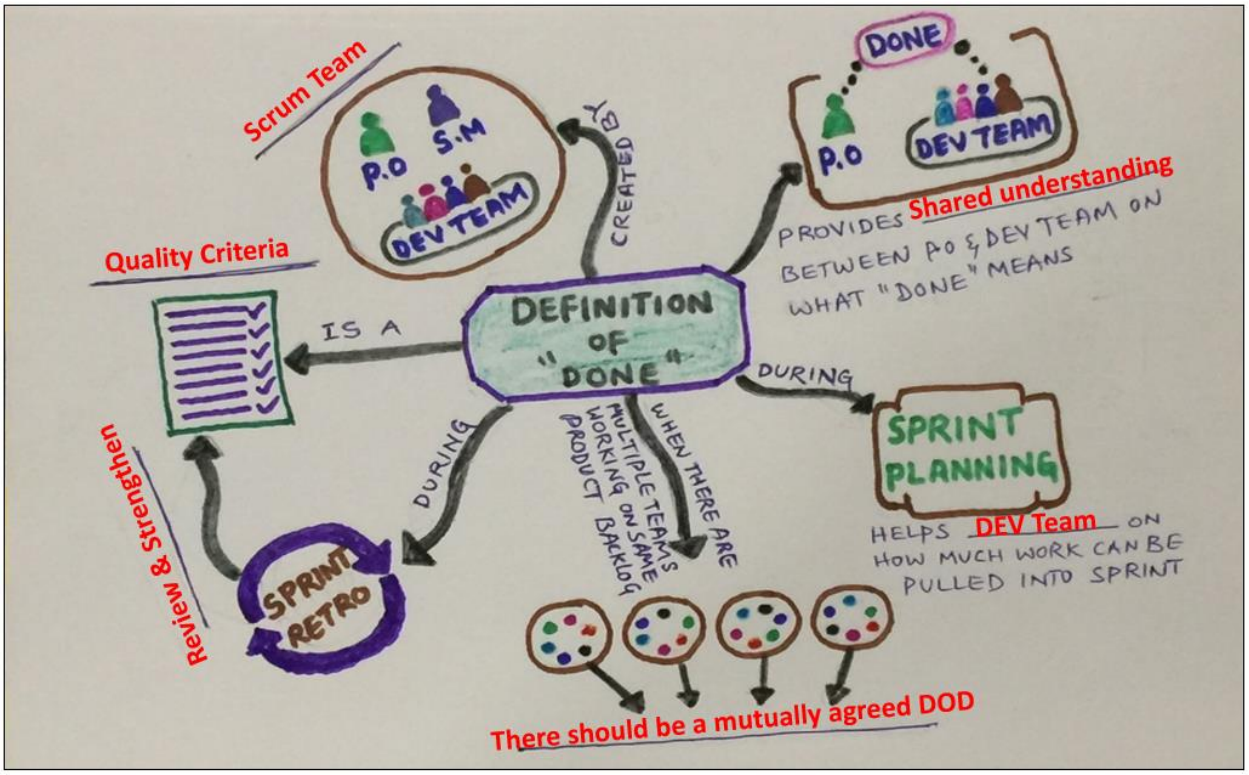


16. **Exercise – Product Backlog Recap**
    All WHITE font boxes are CORRECT and BLACK font boxes are WRONG

17. **Exercise – Daily Scrum**



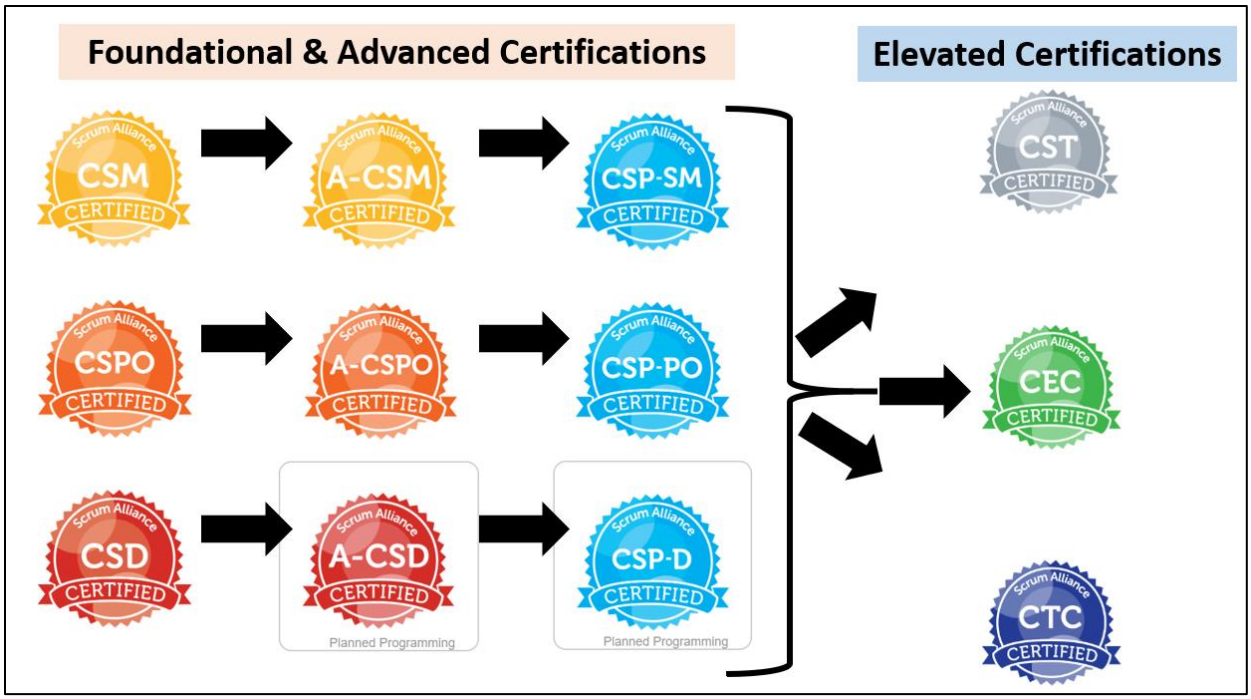
18. Exercise – Definition of Done



19. Exercise - Acceptance Criteria Vs Definition of Done

Acceptance Criteria	Definition of Done
Owned by the Product owner	Owned by Development Team
Related to Functionality	Related to Quality
Defined at any time	Usually defined as soon as possible before sprinting
Doesn't change usually	Enhanced through periodic inspect and adaption
Different for every Product Backlog item	Applies to entire Product backlog and Product increment

APPENDIX – 2: SCRUM ALLIANCE CERTIFICATIONS



CSM is for the Scrum Masters, CSPO is for the Product Owner and CSD is for the Development Team role. These are the foundation level courses for each of the 3 roles of Scrum. Once you achieve any one of these three, you can go to the intermediate level certification in the same path. Example, if you complete CSM then you have to move to A-CSM (Advanced CSM). After the A-CSM you will then focus on advanced certification which is CSP-SM (Certified Scrum professional – Scrum Master). The same applies to other two roles related certifications.

Upon completing the CSP on any path, you can then focus on "Elevated Certifications" which are more advanced. There are two categories of certifications in this path. CST (Certified Scrum Trainer) is to become a Scrum Trainer and to teach all foundation and advanced courses. CTC (Certified Team Coach) and CEC (Certified Enterprise Coach) are for the coaching at team level and to help organizations in transforming into Scrum. Based on your interest, you can choose any or both paths. There is no any order or hierarchy in the elevated certifications. After CSP, you can directly go to CST, or CEC or CTC then CEC. It is up to your interest and passion. But CST is more training focused and CTC and CEC are more of coaching focused.

For more information, please visit: https://www.scrumalliance.org/certifications

APPENDIX – 3: 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 – 4: 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]?

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

☐ 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]

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

---

[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 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 – 5: 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. **Scrum Guide - http://www.scrumguides.org/**
2. **Scrum Alliance – http://www.scrumalliance.org/**
3. **Mike Cohn – http://www.mountaingoatsoftware.com/**
4. **Linked in user Group – https://www.linkedin.com/groups/8305379**
5. **The New New Product Development Game - https://www.antiguosupv.org/wp-content/uploads/2017/07/TheNewNewProductDevelopementGame.pdf**
6. **Soft copy of this material and some additional content: https://drive.google.com/drive/folders/1uW7fyHq3J9NDp_Nva3QVIU-VQkqfYCh2?usp=sharing**
7. **For different retrospective models:**
   - **www.funretrospectives.com**
   - **www.tastycupcakes.com**
   - **www.conteneo.com**

Books: