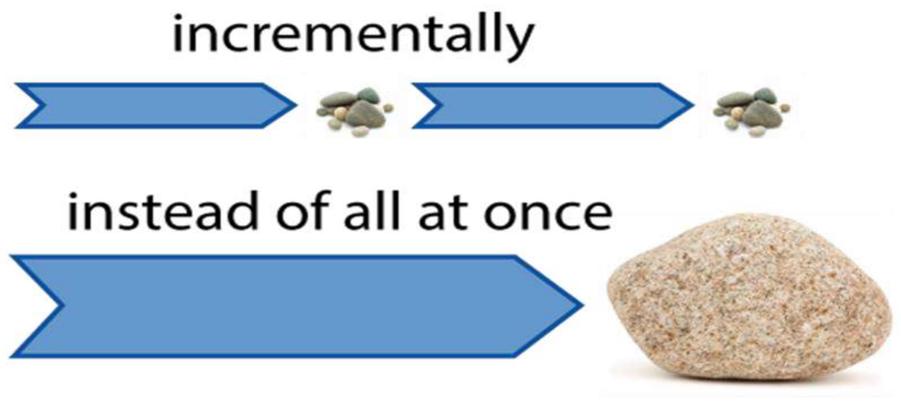
AGILE

What is Agile?

- Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches.
- Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments.
- Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.



- Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.
- It works by breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short two-week cycles called iterations.

History of Agile:

<u>1930s-1940s</u>

- Walter Shewhart a quality expert at Bell Labs Proposed a series of short "plan-do-study-act" (PDSA).
- Later in 1940s Edward Deming, another quality guru promoted this oppressively.
- Later in 1982 Thomas Glib applied the same in S/w Development

<u>1970s-1980s</u>

- 1984- Theory of Constraints. Physicist Eliyahu Goldratt formulates the Theory of Constraints.
- TRW used IID (Iterative and Incremental Development) Missile Software, IBM FSD S/w for the Command Centre for Submarine Barry Bohme promoted spiral model did formalize and make prominent the risk-driven-iterations concept

- Thomas Glib discussed his IID practice—evolutionary project management—and introduced the terms “evolution” and “evolutionary”. He suggested to implement complex project with a series of small steps and each step has a clear measure of success condition

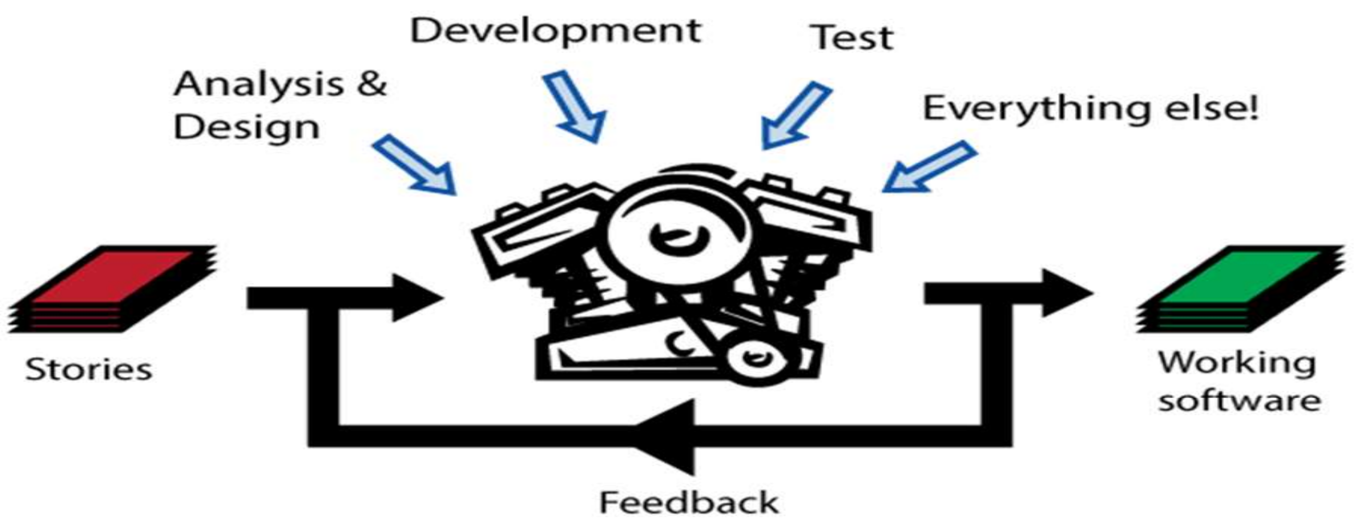
1990s

- 1992-Crystal. Alistair Cockburn invents the Crystal methods.
- 1993-Refactoring
- 1995-pair programming
- 1995-scrum
- 1999-extreme programming
- 1999-continuous Integration
- 2001-Agile Manifesto.
- January 1994, a group of 16 rapid application development (RAD) practitioners met in the UK to discuss the definition of a standard iterative process to support RAD development. This methodology is finally known as Dynamic System Development Method

2001 Onward

- In February 2001, a group of 17 process experts—representing DSDM (Dynamic System Development Method), XP, Scrum, FDD (Feature Driven Development), and others—interested in promoting modern, simple IID methods and principles met in Utah to discuss common ground.
- Team produce 4 agile Values and 12 Principles. These Agile Values and Principle form the basis for Agile Software Development.

Iterations:



- An Agile iteration is a short one-to-two-week period where a team takes a couple of their customers' most important user stories and builds them completely as running-tested-software.
- This means everything happens during an iteration. Analysis, design, coding, testing. It all happens here. The beauty of working this way, is every couple of weeks the customer gets something of great value (working software), but it's also a great way to track progress (measuring the rate at which the team can turn user stories into production ready working software).

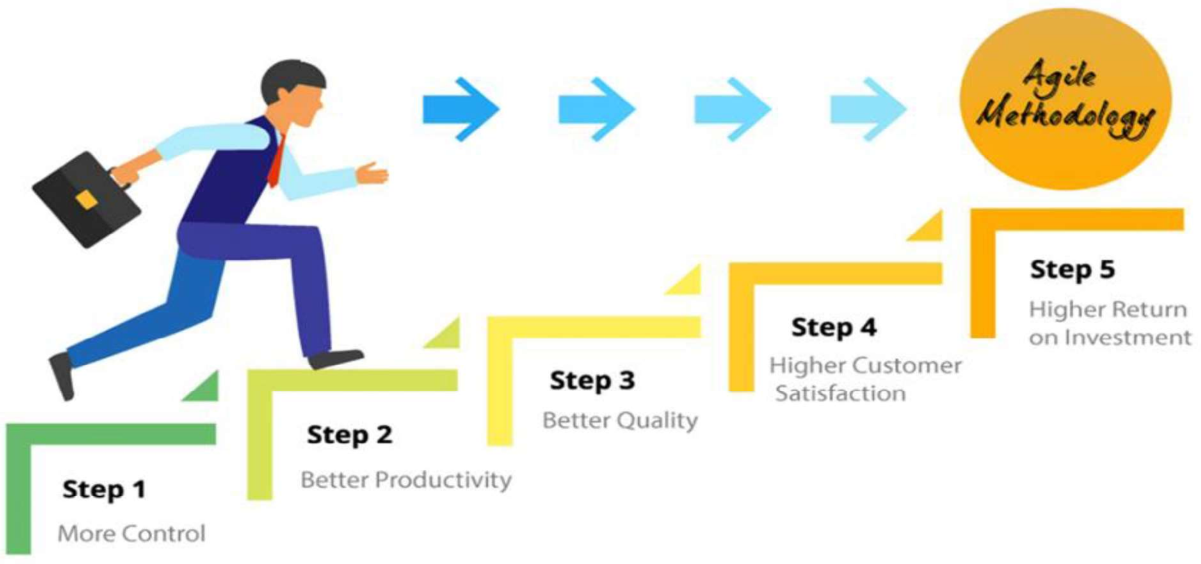
Where to use Agile:

While Agile is still used predominantly in the software development industry, it is also very popular in these industries:

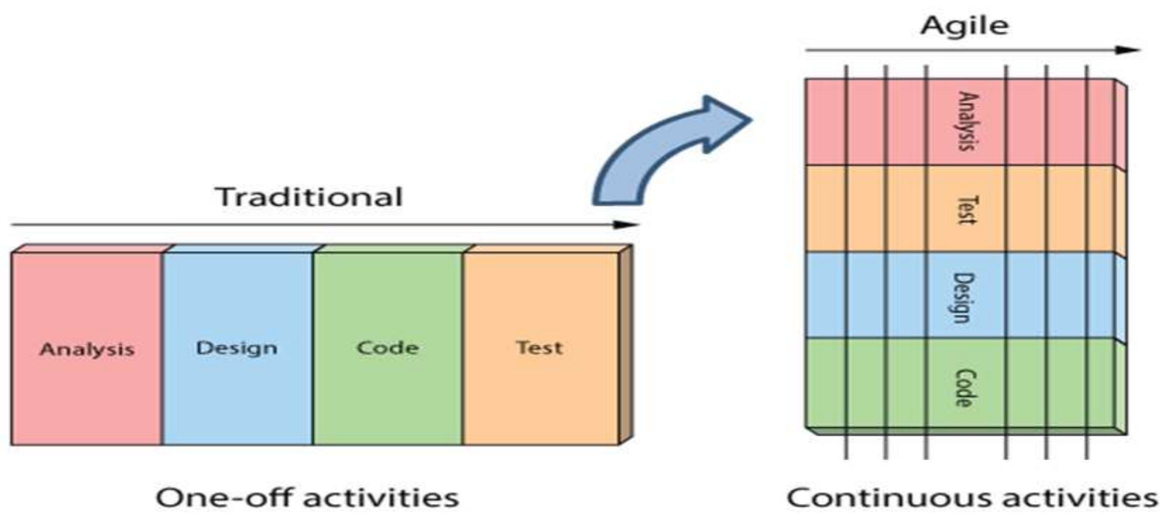
- FMCG
- Finance
- Marketing
- Automobile
- Real Estate
- Manufacturing
- Product development
- Pharmaceuticals
- Healthcare
- Business services
- Logistics
- Travel/leisure

Agile Methodology:

Agile is a software development methodology that helps to deliver the value faster and promotes continuous iteration of development and testing throughout the software development life cycle. It is a step-by-step process to develop our project and make it successful.

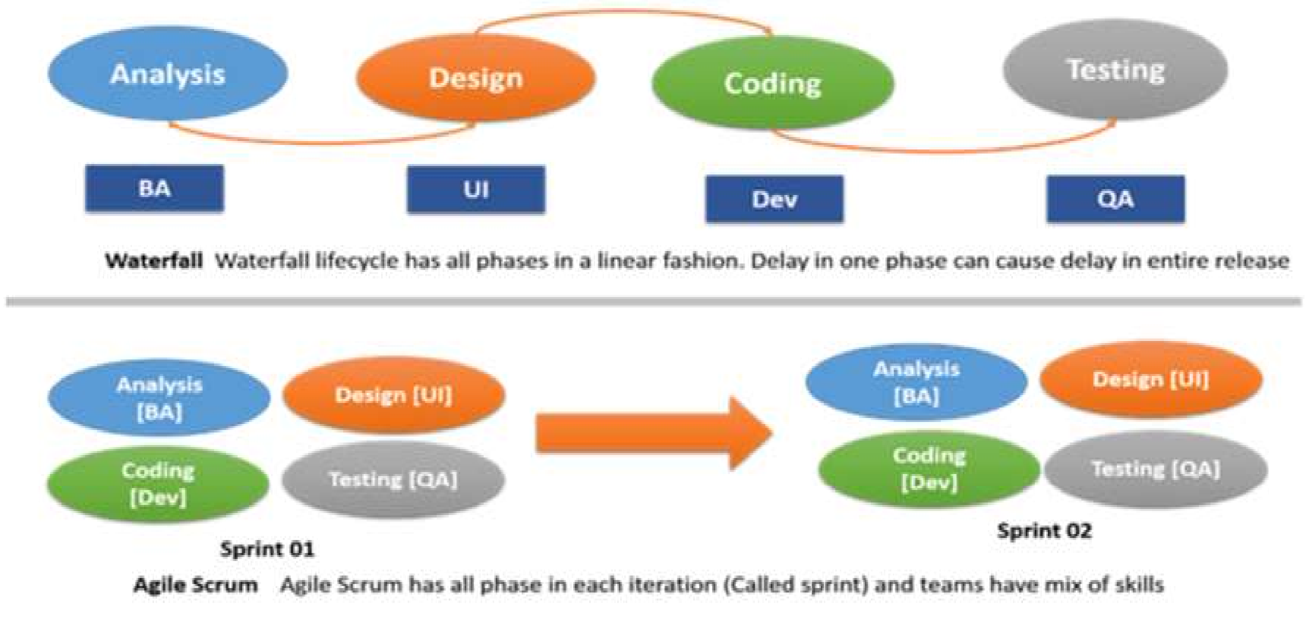


Difference between Agile and Traditional software development approach:



Agile	Traditional approach
focus on the people doing the work	focus on process of doing the work
focus on how people work together	focus on the tools to be used for doing the work
Customer involvement is there in every phase for feedback.	The customer isn’t involved in the process, only the final product is shown to the customer.
Scope is flexible	Scope is fixed
Cost and time are fixed	Cost and time are flexible

Agile VS Waterfall:


Analysis
Design
Coding
Testing
BA
UI
Dev
QA
Waterfall Waterfall lifecycle has all phases in a linear fashion. Delay in one phase can cause delay in entire release
Analysis [BA]
Design [UI]
Coding [Dev]
Testing [QA]
Sprint 01
Analysis [BA]
Design [UI]
Coding [Dev]
Testing [QA]
Sprint 02
Agile Scrum Agile Scrum has all phase in each iteration (Called sprint) and teams have mix of skills

Waterfall vs Agile Scrum

Waterfall challenges:

Traditional Waterfall treats analysis, design, coding, and testing as discrete phases in a software project. This worked OK when the cost of change was high. But now that it's low it hurts us in a couple of ways.

I. **Poor quality:**

First off, when the project starts to run out of time and money, testing is the only phase left. This means good projects are forced to cut testing short and quality suffers.

II. **Poor visibility:**

Secondly, because working software isn't produced until the end of the project, you never really know where you are on a Waterfall project. That last 20% of the project always seems to take 80% of the time.

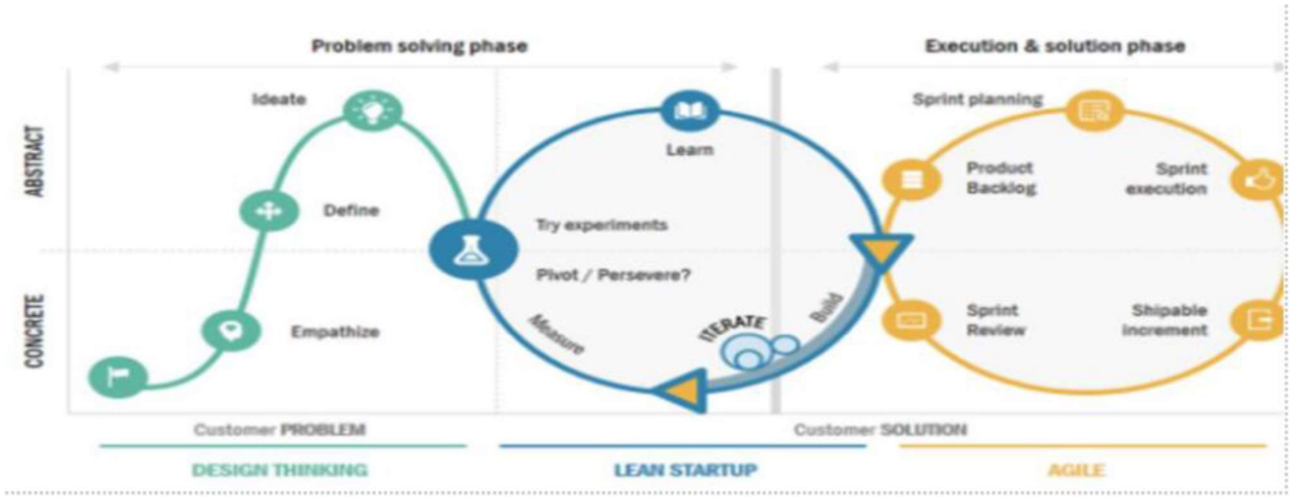
III. **Too risky:**

You've got technical risk because you don't get to test your design or architecture until late in the project. And you've got product risk because you don't even know if you are building the right until it's too late to make any changes.

IV. **Can't handle change**

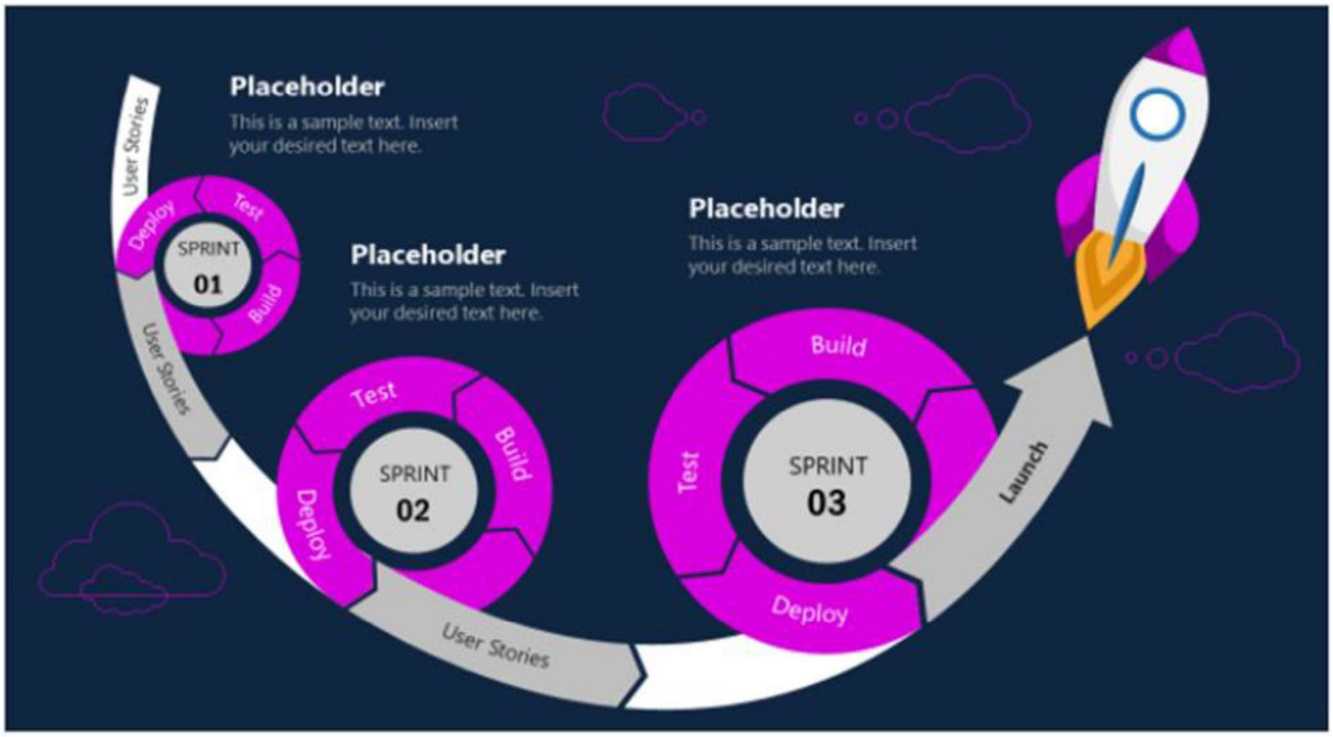
And finally, most importantly, it's just not a great way to handle change.

Design Thinking Lean Startup Agile Diagram:



- Design Thinking: Emphasize – Define – Ideate
- Lean Startup: Measure – (multiple Iterations) - Build – Lean
- Agile: Sprint review – shippable increment – sprint execution – sprint planning – product backlog.

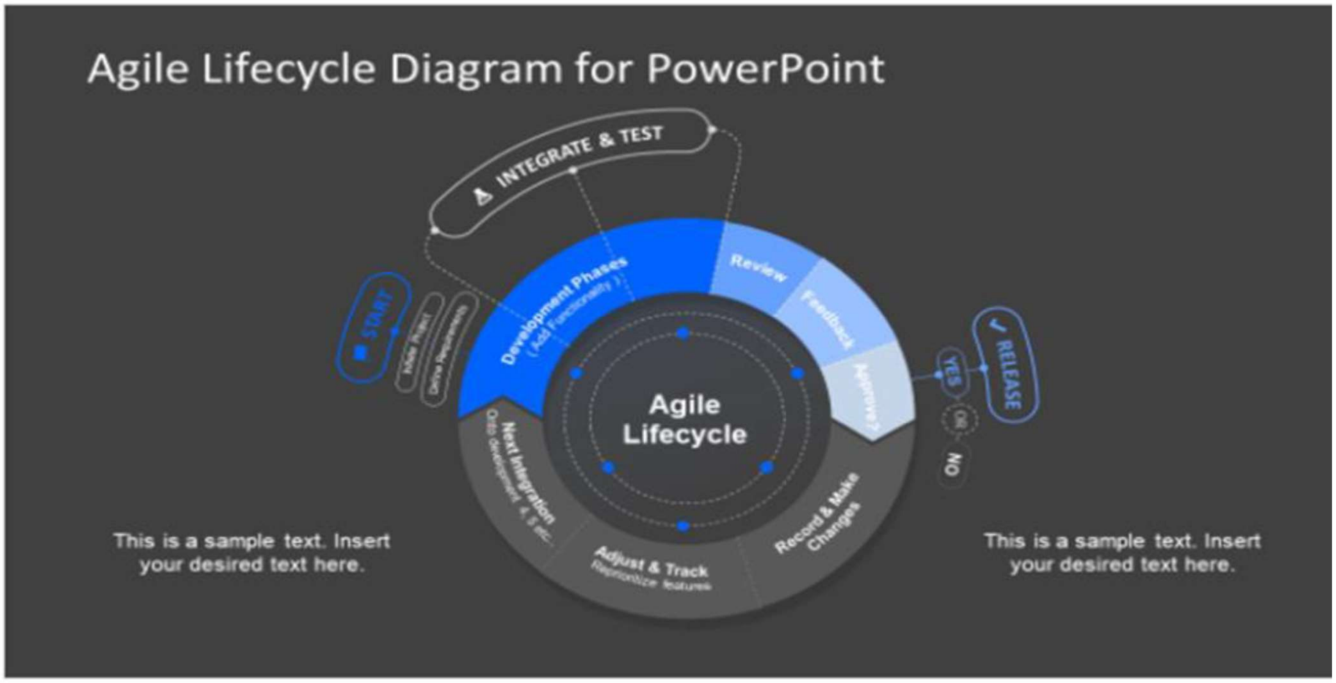
Agile Development Process:


Placeholder
This is a sample text. Insert your desired text here.
User Stories
Deploy
Test
Build
SPRINT 01
Placeholder
This is a sample text. Insert your desired text here.
Test
Build
Deploy
SPRINT 02
Placeholder
This is a sample text. Insert your desired text here.
Build
Test
Deploy
SPRINT 03
User Stories
Launch

Agile Life Cycle:

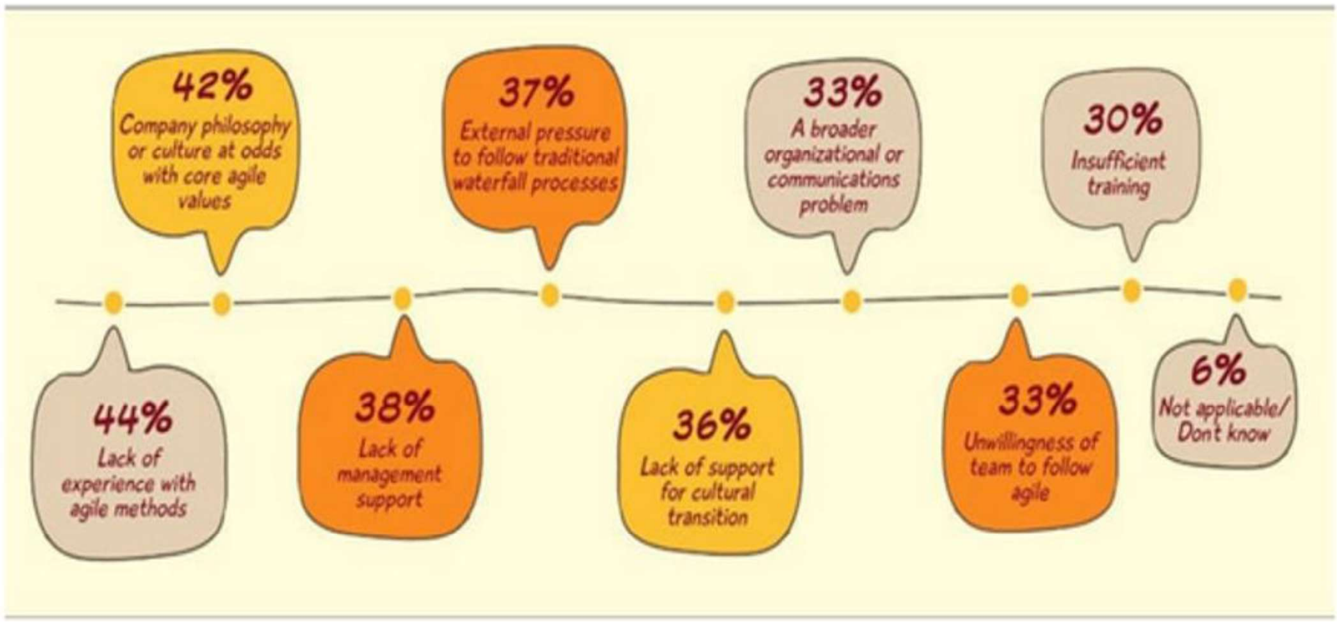
- Seamless display of development process for employees and customers.

- Helps create an illustrative analysis of project procedures.
- A professional summary of Agile lifecycle according to business processes.
- Explainable units for educational purposes.
- Presentation of Agile framework and timeline.

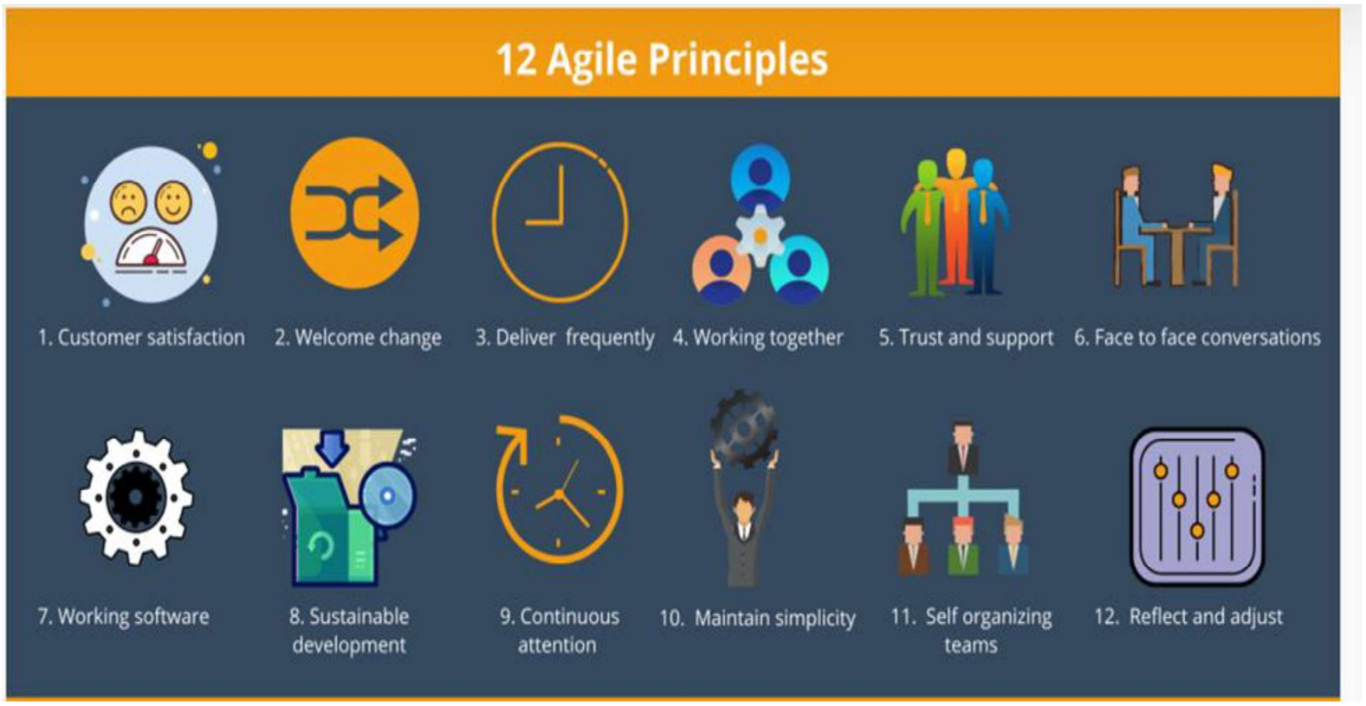


Why Agile Get Failed Sometimes:

☹ Lack of Experience with Agile Methods.
☹ Company Philosophy or Culture at Odds with Core Agile Values.
☹ Lack of Management Support.
☹ External Pressure to Follow Traditional Waterfall Processes.
☹ Lack of Support for Cultural Transition.
☹ A Broader Organizational or Communications Problem.
☹ Unwillingness of Team to Follow Agile.
☹ Insufficient training.



Principles Governing Agile:



1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
   - ✓ It's true that it takes a lot of time and effort to build a product.
   - ✓ But at the end of the day what makes a customer happy is a valuable Working software.
   - ✓ Therefore, this principle emphasizes early and continuous delivery of the software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
   - ✓ Agile believes in empiricism. We become more knowledgeable when we have spent time working on something, in the beginning, we me have a lot of knowledge, but not as much as we have after the commencement of the work.
   - ✓ Similarly, our customers understand their needs even more clearly after seeing the increment provided by us.
   - ✓ Therefore, these principals welcome changing requirements from the customers so that the process remains empirical, and the customer gets what is required to stay in the competition.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
   - ✓ This principle emphasizes the frequent delivery of the product as frequent delivery helps the development process to be more agile.
   - ✓ It mitigates the risks in your release, you get faster feedback from the customer and hence u can bring changes in the product before it is too late.

4. Businesspeople and developers must work together daily throughout the project.

- ✓ This principle helps in keeping the business aspect and the technical aspect of the project on the same page.
- ✓ When businesspeople and developers work together, they gain a common understanding of the direction towards which the project is moving.
- ✓ The development team also gets an end user view from the business side.

5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
   - ✓ This principle emphasizes having motivated team members.
   - ✓ When people are motivated to do their work, they produce results which can never be produced by people who are forced to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
   - ✓ Most of the communication in a project is about project process or project content.
   - ✓ It is crucial for the success of the project that the development team understand the right information correctly.
   - ✓ Written communication is prone to ambiguity whereas face to face conversation provides the ground for prompt clarification and quicker communication.
   - ✓ As a result, product development becomes faster and precise.

7. Working software is the primary measure of progress.
   - ✓ As we know that working software is most valuable for the customer.
   - ✓ The planning of construction of the product has no direct benefits for them until a working software is not given in their hands.
   - ✓ Software is not finished when it is successfully tested and delivered, it is finished when it is tested and accepted by the end user.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
   - ✓ Software development is like running a marathon, you must maintain the speed, but you shouldn’t run so fast that you exhaust yourself.
   - ✓ Similarly, sustainable development happens when you do constant production of software features during a long-lasting period.

9. Continuous attention to technical excellence and good design enhances agility.
   - ✓ Agile focuses on development of software like craftmanship rather than just working on regular tasks.
   - ✓ Continuous attention to technical excellence and good design helps in bringing this craftmanship to the project.

Good Design

10. Simplicity--the art of maximizing the amount of work not done--is essential.
    - ✓ More features make a product more complex.
    - ✓ Complex codes are difficult to test, maintain and develop.
    - ✓ Sometimes some products have features which are so rarely used that their development and maintenance results in a negative ROI.
    - ✓ Therefore, it is better to create a product which does a few things extraordinarily and is completely useful for someone rather than creating software which tries to do everything does nothing extraordinarily and is not useful to anyone.

Simplicity

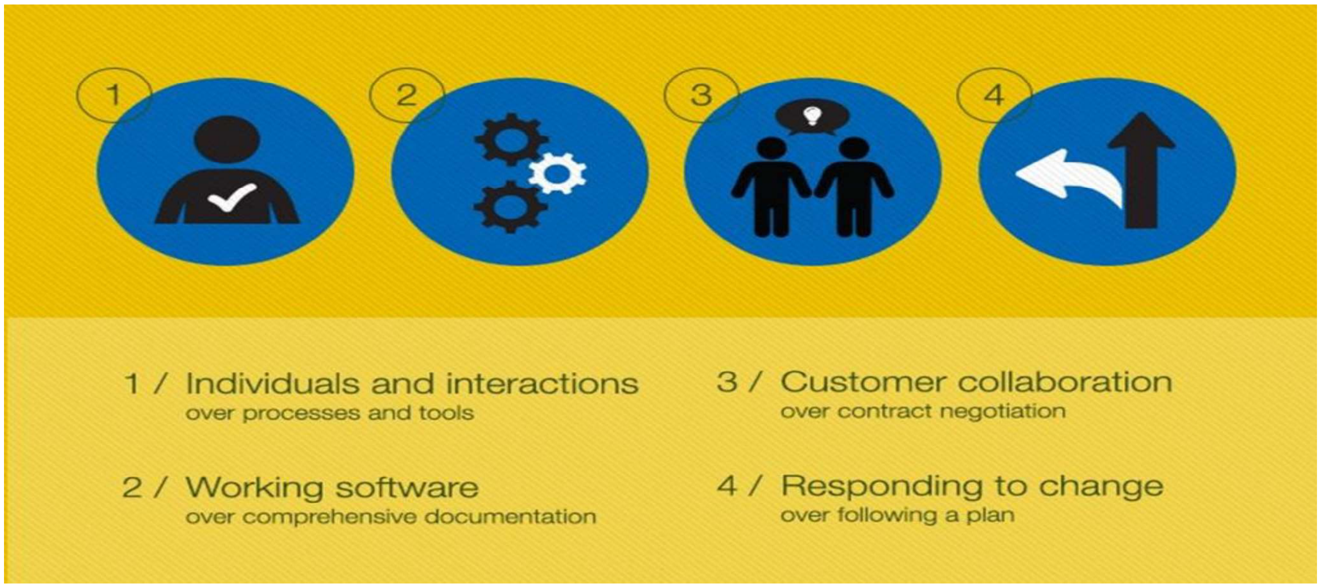
11. The best architectures, requirements, and designs emerge from self-organizing teams.
    - ✓ A self-organizing team is one that does not depend on or wait for others to assign work.
    - ✓ Instead, these teams find their own work and manage the associated responsibilities and timelines.
    - ✓ They take on the responsibility of choosing the most effective and efficient way to complete their work and regularly look for ways to improve through experimentation.
    - ✓ While self-organizing teams don’t require a manager to assign work, set deadlines, and so on, they do require a mentor who ca help grow their skills.
    - ✓ Having self-organizing teams promotes collaboration, teamwork, competency, regular growth, motivation and commitment.
    - ✓ Hence, the best architecture, requirements, and design emerge from such teams.

Self Organization

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
    - ✓ Agile encourages continuous improvement and productivity.
    - ✓ When a team reflects on how to become more effective, and adjusts its behavior accordingly at regular intervals, it brings continuous improvements and increases its productivity.

Reflect and Adjust

Values of Agile:

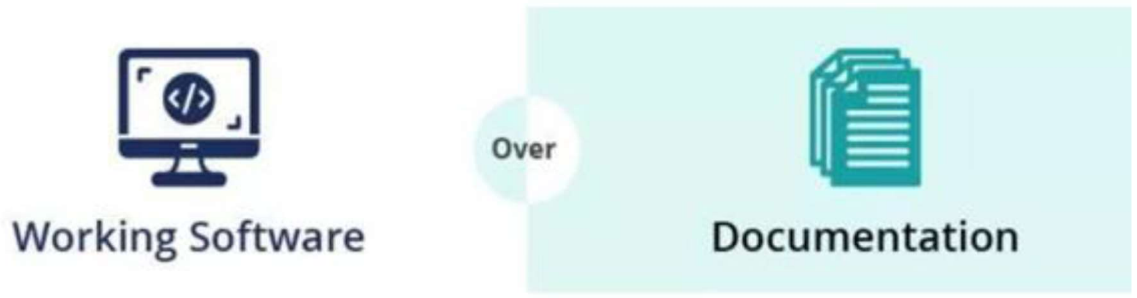

1 / Individuals and interactions over processes and tools
2 / Working software over comprehensive documentation
3 / Customer collaboration over contract negotiation
4 / Responding to change over following a plan

I. Individual and interaction:

Individuals and interactions over processes and tools. Valuing people more highly than processes or tools is easy to understand because it is the people who respond to business needs and drive the development process.

II. Working software:

Here, Agile comes into play and makes things easier for the developers. It breaks down the requirements of the client in the form of documents as user stories and that is exactly what each developer would need to begin working on developing the software.

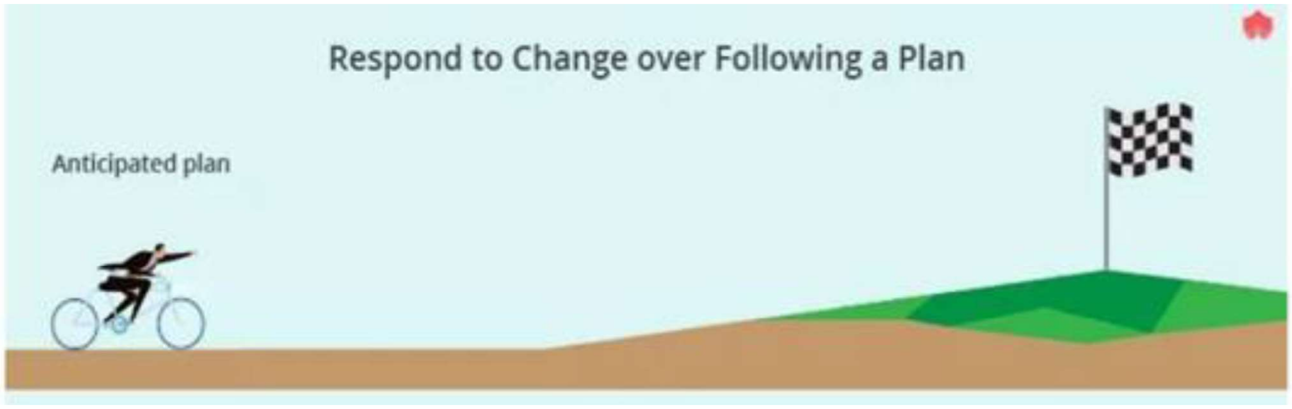


III. Customer collaboration:

Sometimes, legal contracts with customers act as a barrier for you in communicating with your customers. You will need to devise a plan to separate the legal bounding that you have with your customers from the product relationship. Contract negotiations will be there as a part of the deal, but forming a relationship with the customer to facilitate communication will help you interact with the customers with a human touch, failing to do which will not help in developing great software.

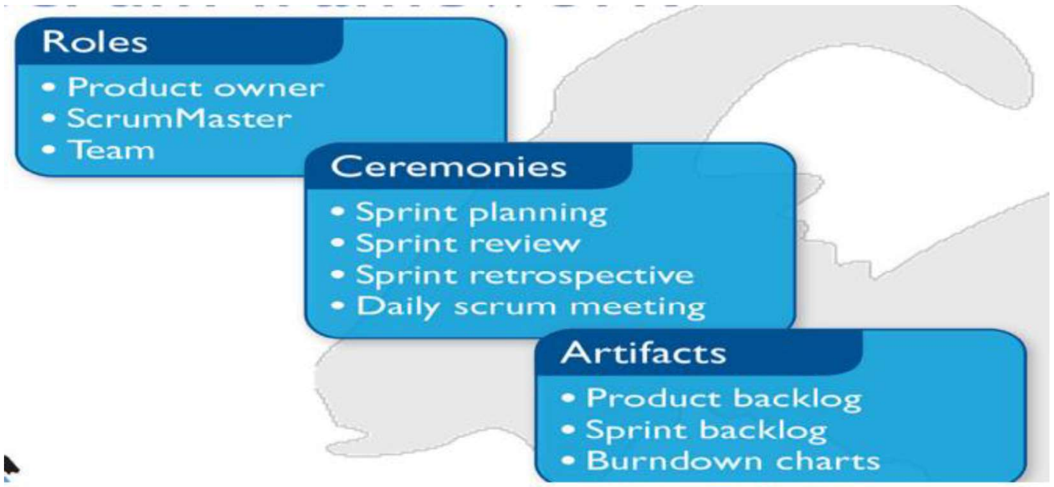
IV. Respond to change:

In order to get this issue sorted out, you need to go back to the first value of Agile, which is communicating across teams to stay updated about the changes for a better and more effective workflow. It is more like an initiative to be taken by the testing team, that is, to communicate with the developers to stay in the loop of changes or a new course of action.



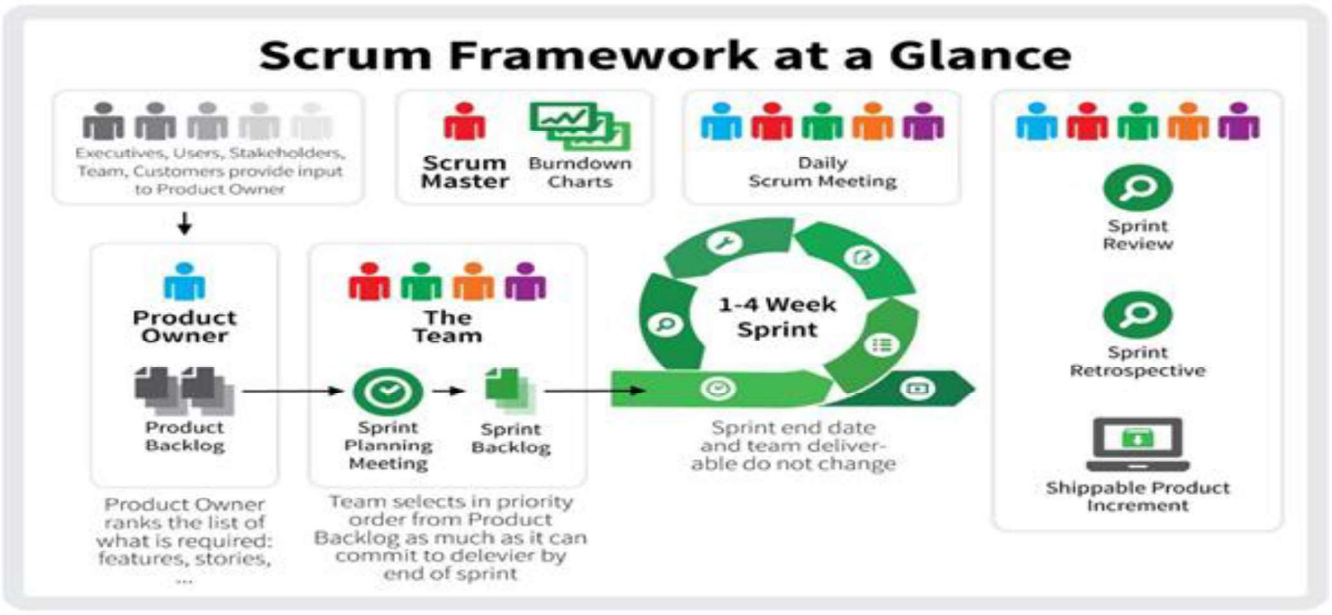
SCRUM FRAMEWORK

Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.



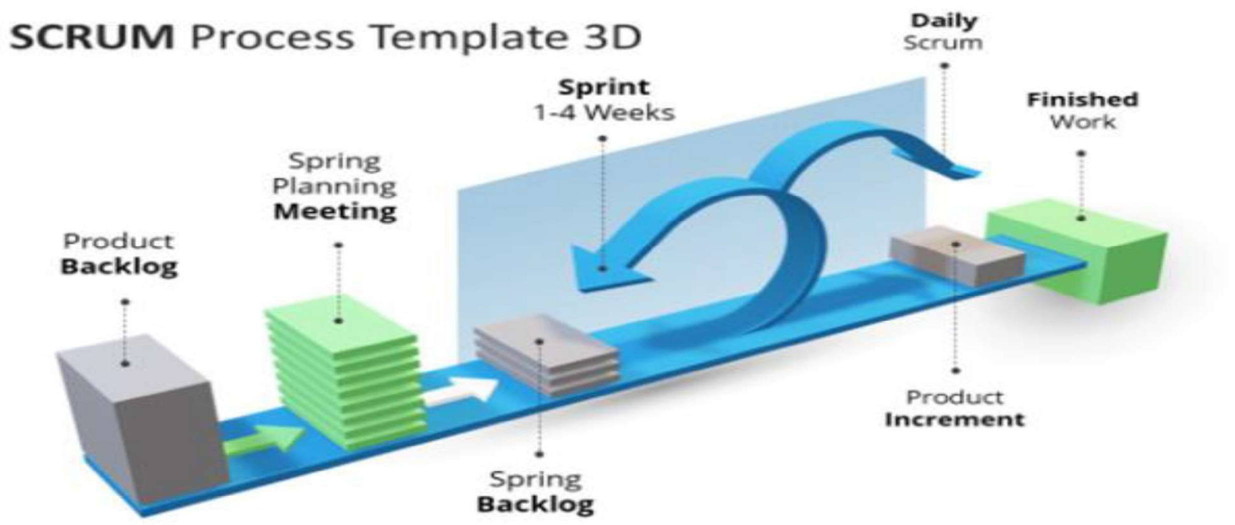
Pillars of Scrum:

- Transparency
- Inspection
- Adoption

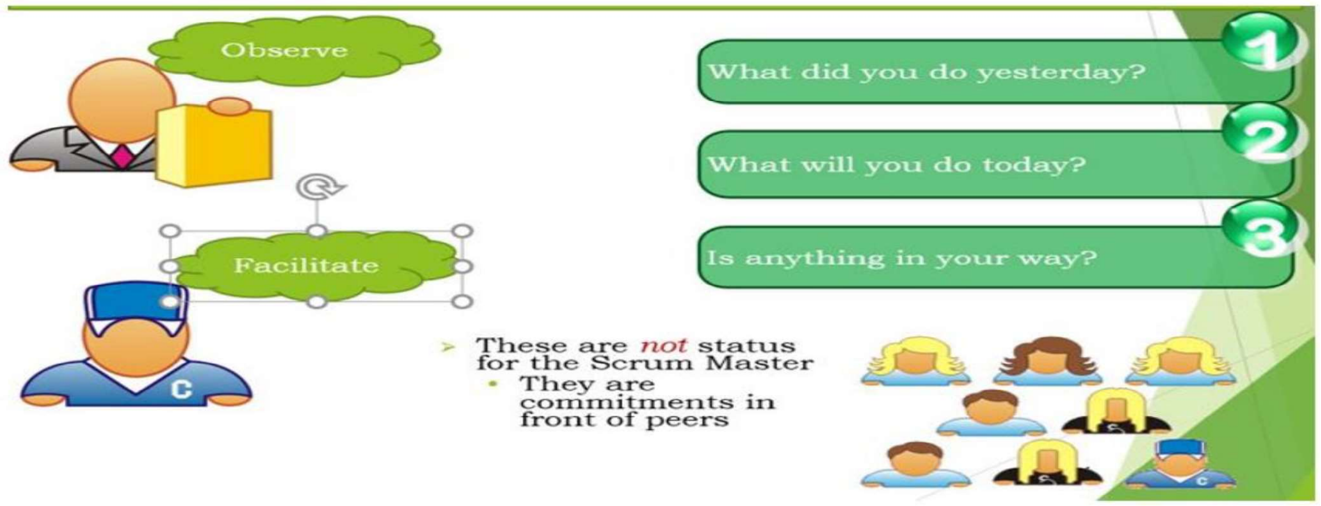

Scrum Framework at a Glance
Executives, Users, Stakeholders, Team, Customers provide input to Product Owner
Scrum Master
Burndown Charts
Daily Scrum Meeting
Product Owner
Product Backlog
The Team
Sprint Planning Meeting
Sprint Backlog
1-4 Week Sprint
Sprint end date and team deliver-able do not change
Sprint Review
Sprint Retrospective
Shippable Product Increment
Product Owner ranks the list of what is required: features, stories, ...
Team selects in priority order from Product Backlog as much as it can commit to delevier by end of sprint

★ The first pillar is **Transparency** – the word itself in terms of scrum means that the processes carried out in the product development should be visible to those who are responsible for the better outcome.
★ Second is **Inspection** – checking should be done timely before the product is presented to the third party to avoid uncertainties and problems. It's not that hard to understand, but the process is vital since the next step depends on it.
★ Third and most important pillar is **Adaption** – after certain processes and inspection if the product qualifies all the requirements, then the same process with some adjustments is applied to other products to minimize the issue.

3D Animated Scrum Process:



- **Product backlog:** A list of things that need to be done within the project.
- **Sprint planning meeting:** A meeting between the scrum master, product owner and scrum team to describe the high priority features.
- **Sprint backlog:** A list of tasks defined by the scrum team to deliver within 1 sprint cycle.
- **Sprint (1-4 weeks):** A timeframe to complete the task.
- **Daily Scrum:** A daily team meeting to review work and identify any problem which may halt the project completion.


Observe
Facilitate
What did you do yesterday?
What will you do today?
Is anything in your way?
1
2
3
These are *not* status for the Scrum Master
They are commitments in front of peers

- **Product Increment:** One increment occurs after completing the items from product backlog in one sprint.
- **Finished Work:** The items marked-as-completed after the final review.

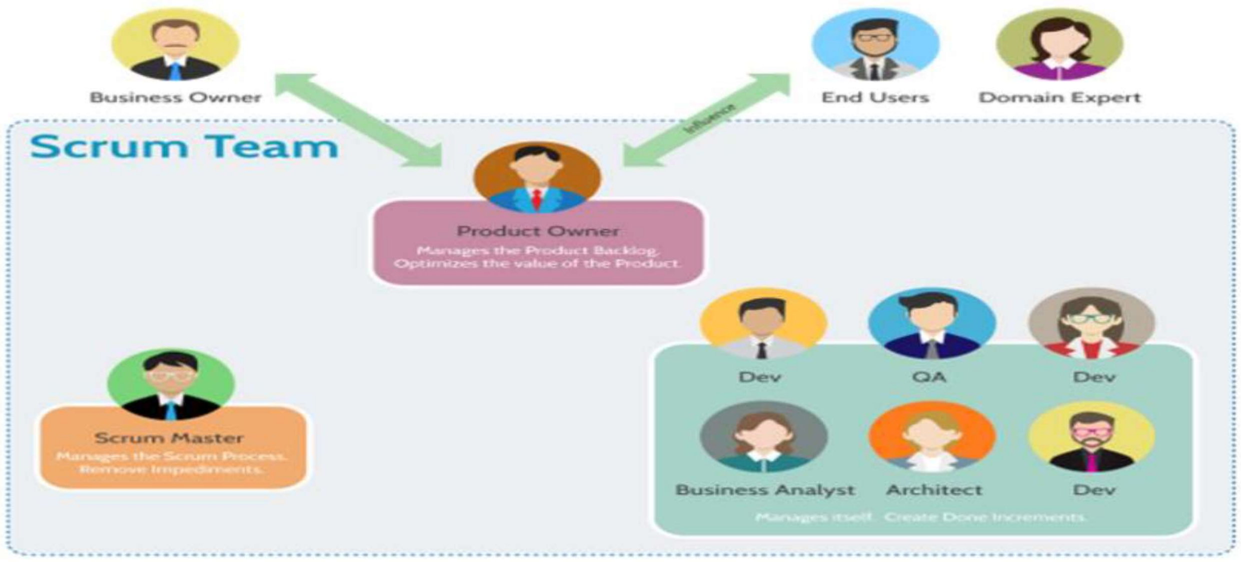
Scrum Values:

These are the values that each team member must indulge into him. We can get to know about the values thoroughly and the first one is **COURAGE**. This is the normal aspect I think that each member of the team should have. Courage to do the work, face the problem & solve it are the main things. Second is **FOCUS**, the focus of each member should be clear, and he/she should be knowledgeable to do so. Third is **COMMITMENT,** which is nothing, but the people should be committed to do work to achieve the final goal. **Respect** is the most important value, and each member should give to one another. The last one is **OPENNESS**, there should be openness between the team members & its stakeholders to avoid misunderstanding. If we follow these values, we are one step closer to solving the problems.

Scrum Ceremonies:

	Sprint Planning	Daily Scrum	Sprint Review	Sprint Retrospective
Purpose	•Create Sprint Backlog •Identify Sprint Goal	•Provide opportunity to discuss •Progress •Daily commitments •Identify and improve impediments	•Signoff/ Approval •Gather feedback	Inspect and Adapt
What	•Discuss User Stories •Story point Estimation •Sprint Capacity Planning	•What was done •What will be done •Blockers	•Demonstrate working software •Create increment	•Good •Bad •Ideas •Actions
Duration	No longer than 4 hours	No longer than 15 min	2hrs recommended	No longer than 1.5 hours
Attendees	Entire Scrum Team	•Scrum Master Development Team •Product Owner (Optional) •Stakeholders (Optional)	•Entire Scrum Team •Stakeholders •Managers	•Scrum Master Development Team •Product Owner (Optional) •Stakeholders (Optional)

Scrum Team:



So basically, who is the scrum team – the product owner, scrum master & development team. Here there is nothing to say much because we get a clear idea from the topic itself. To elaborate more, we can say that product owner is the end user or customer. Scrum master is someone who manages the work progress and decides the grading accordingly. Lastly the development team includes developer, QA, analyst, architecture etc.

Scrum has been used by:

• Microsoft	• Yahoo	• Google
• Electronic Arts	• IBM	• Lockheed Martin
• Philips	• Siemens	• Nokia
• Capital One	• BBC	• Intuit
• Apple	• Nielsen Media	• First American Corelogic
• Qualcomm	• Texas Instruments	• Salesforce.com
• John Deere	• Lexis Nexis	• Sabre
• Salesforce.com	• Time Warner	• Turner Broadcasting
• Oce		

Scrum has been used for:

• Commercial software	• In-house developments
• Contract development	• Fixed-price projects
• Financial applications	• ISO 9001-certified applications
• Embedded systems	• 24x7 systems with 99.999% uptime requirements
• The Joint Strike Fighter	• Video game development
• Approved, life-critical systems	• Satellite-control software
• Websites	• Handheld software
• Mobile phones	• Network switching applications
• ISV applications	• Some of the largest applications in use

Characteristics:

- Self-organizing teams.
- Product progresses in a series of month-long "sprints".
- Requirements are captured as items in a list of "product backlog".
- No specific engineering practices prescribed.
- Uses generative rules to create an agile environment for delivering projects.
- One of the "agile processes".

Scrum Metrics and KPI

What are scrum metrics?

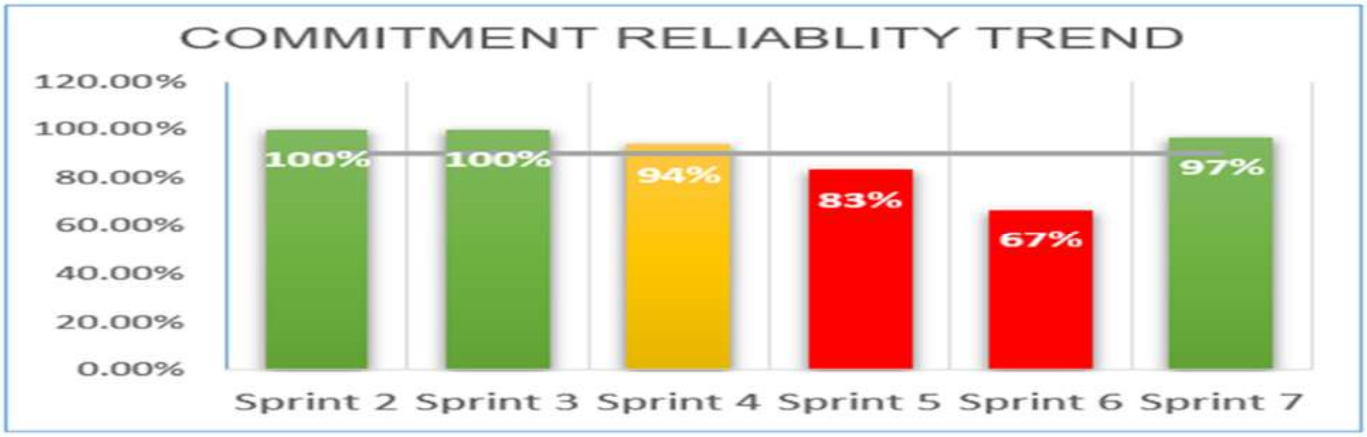
Scrum metrics are specific data points that scrum teams track and use to improve efficiency and effectiveness. When defined, understood, and implemented, scrum metrics can become insights that help guide and improve a team's agile journey. Scrum teams use metrics to inform decision-making and become more efficient in planning and execution. They can also be used to establish a baseline on the status quo and set target goals and improvement plans.

Why do you need scrum metrics?

Scrum metrics can help teams establish benchmarks and guide the direction of the work. For this reason, scrum metrics are helpful for established and new teams alike.

Commitment Reliability:

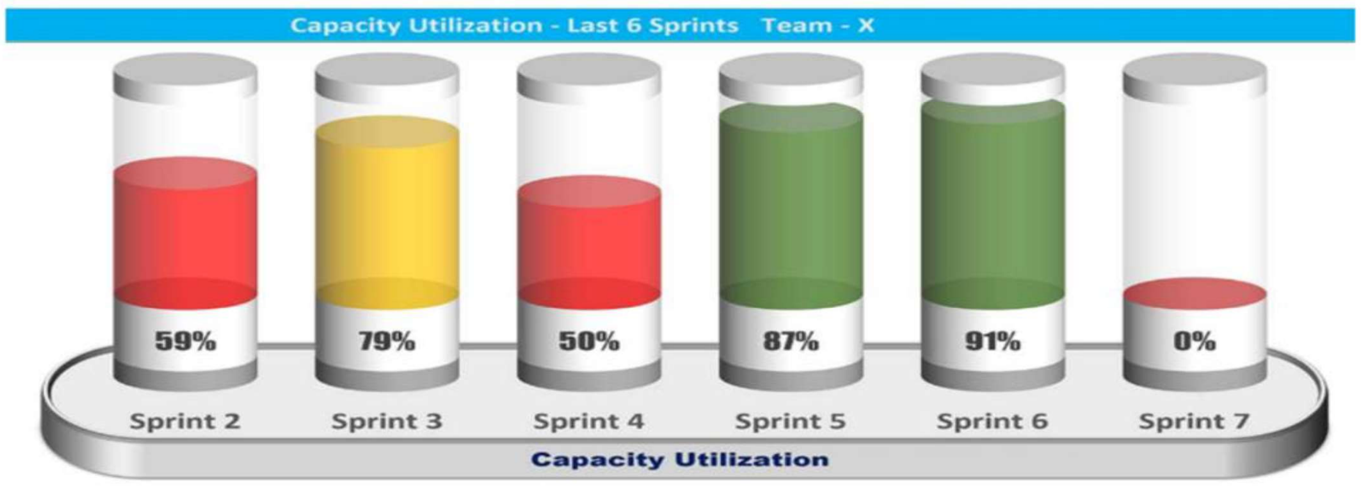
- Shows the percentage of story points completed against its commitment for every sprint.
- This is also referred to as your Say/Do Ratio. Well-functioning organizations build reliable products, and reliable products are the sum of the reliability of the teams building them.



Capacity Utilization:

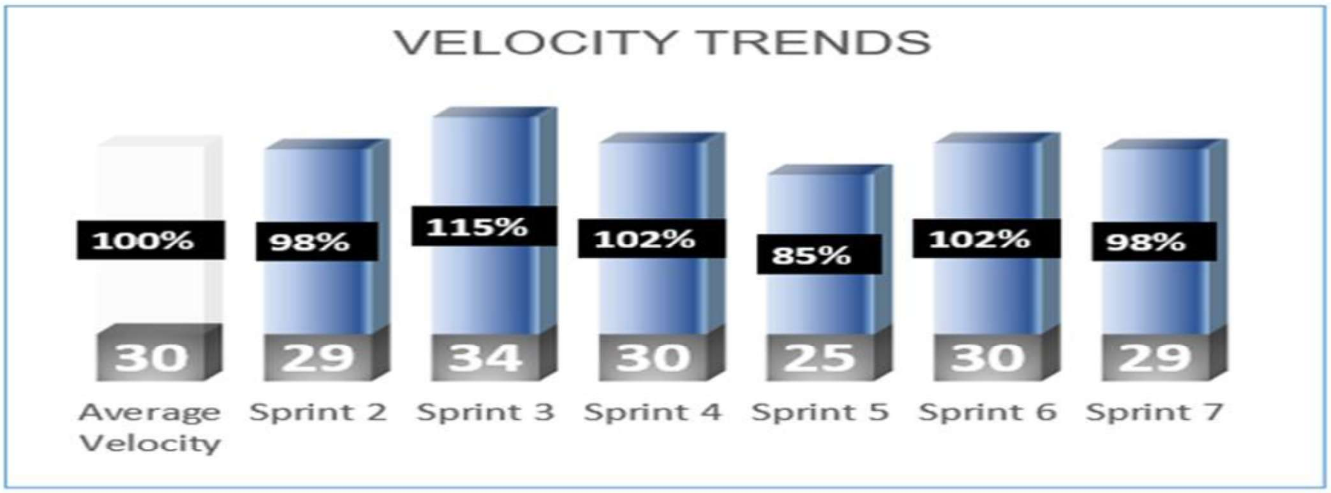
- Represent the team capacity in hours and committed hours for each sprint, measured in effort hours.

- Capacity of an agile team is an important measure that should be used to inform the team how much workload it should undertake during its sprint planning meeting for an upcoming sprint.
- If the capacity and workload are not in balance, then the team members should have an informed conversation with the product owner on how to achieve the balance.



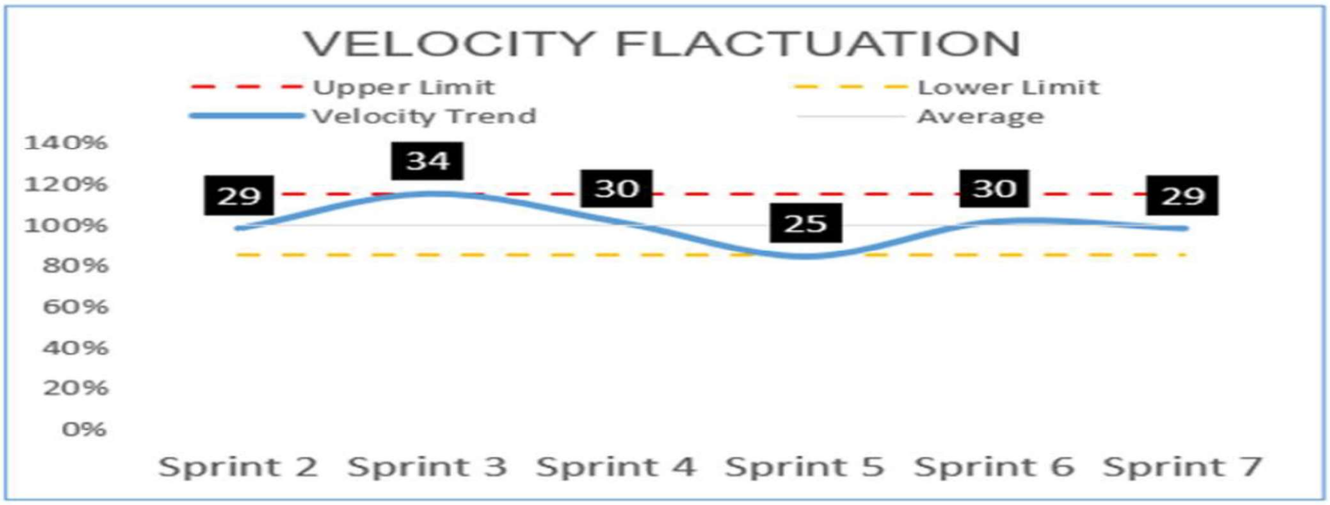
Velocity:

- Represents total number of story points completed each sprint.



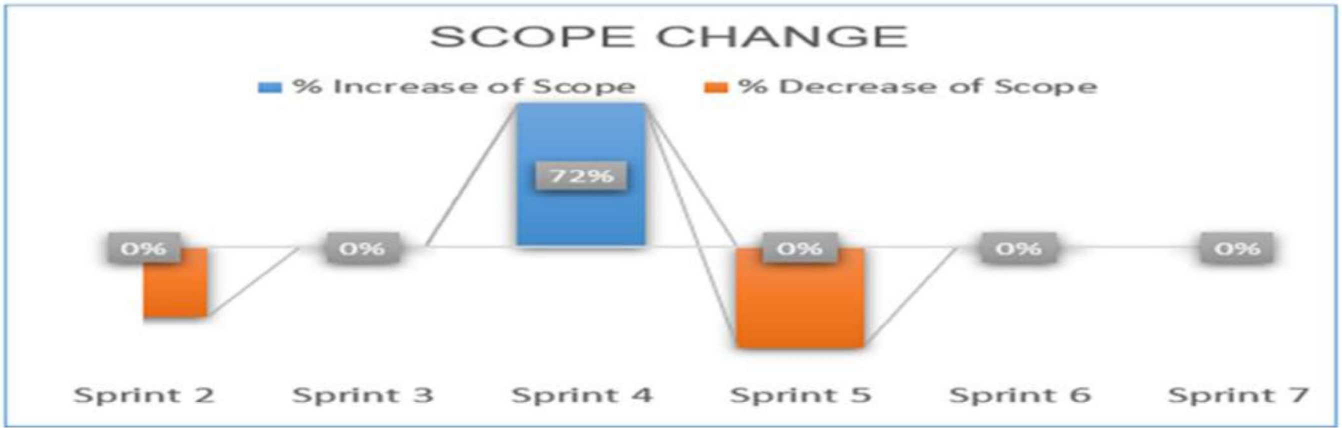
Velocity Fluctuation:

- Represents the velocity variance from average with relation to max and min tolerance.



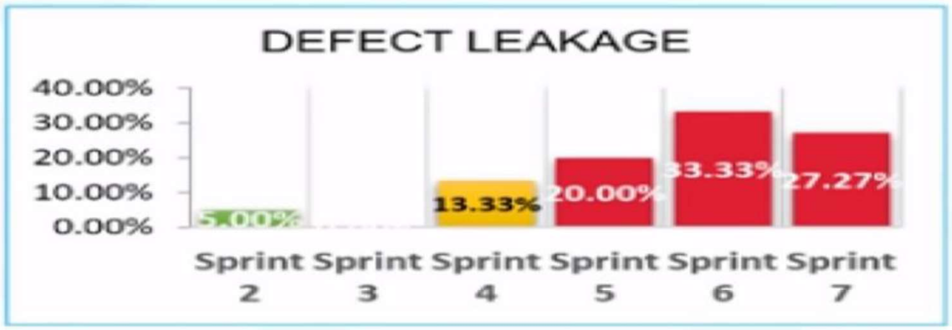
Scope Change:

- Amount of story points added to the sprint or amount of story points removed from the sprint, After Sprint Started in each Sprint.
- Product backlog items added or removed from Sprint scope after sprint started. Usually, scope change doesn't represent good practice but in practical scenario it happens.
- It too reflects the quality of backlog grooming, prioritization, sprint and capacity planning.



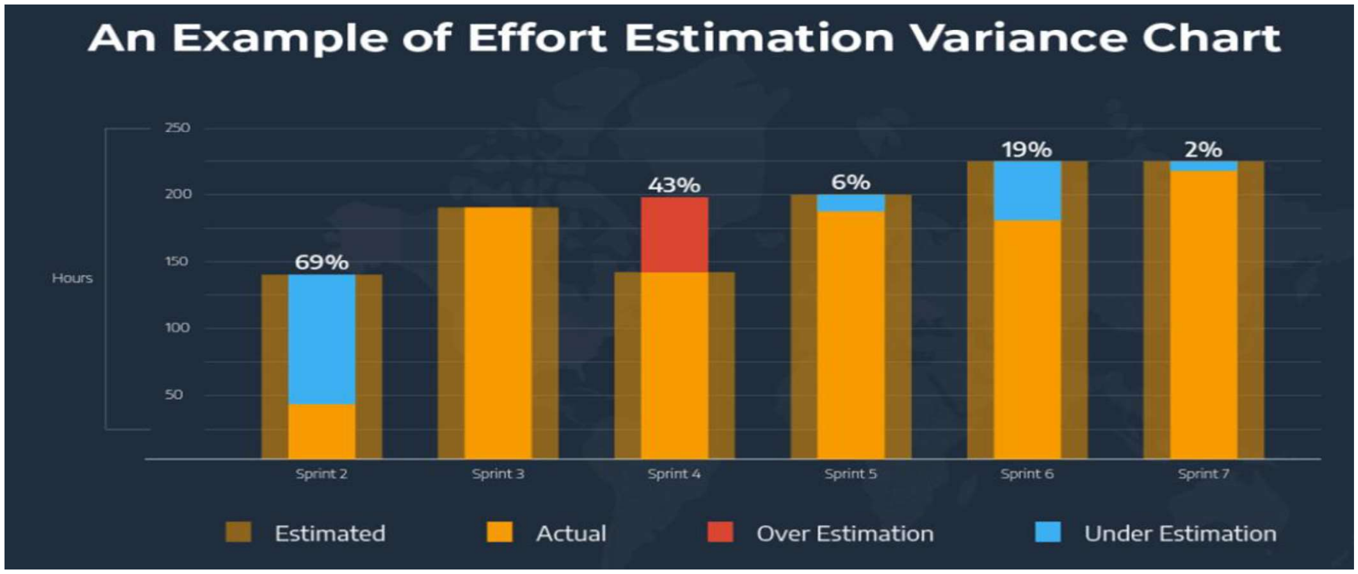
Defect Leakage:

- Number of Defects detected after development, we can calculate it as defects detected by development team Vs Defects detected by Product Owner / UAT.



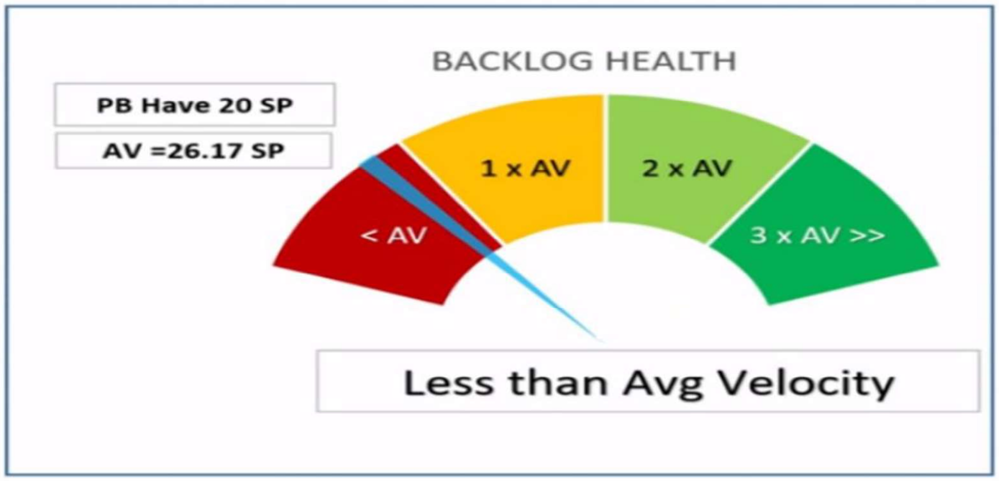
Effort Estimation Variance:

- Effort estimated Vs Actual, shows Overestimation hours and under estimation hours.
- Estimation Variation represents the comparison of the effort that was planned (Estimated Hours) among the actual efforts (Spent Hours) within a sprint.



Backlog Health:

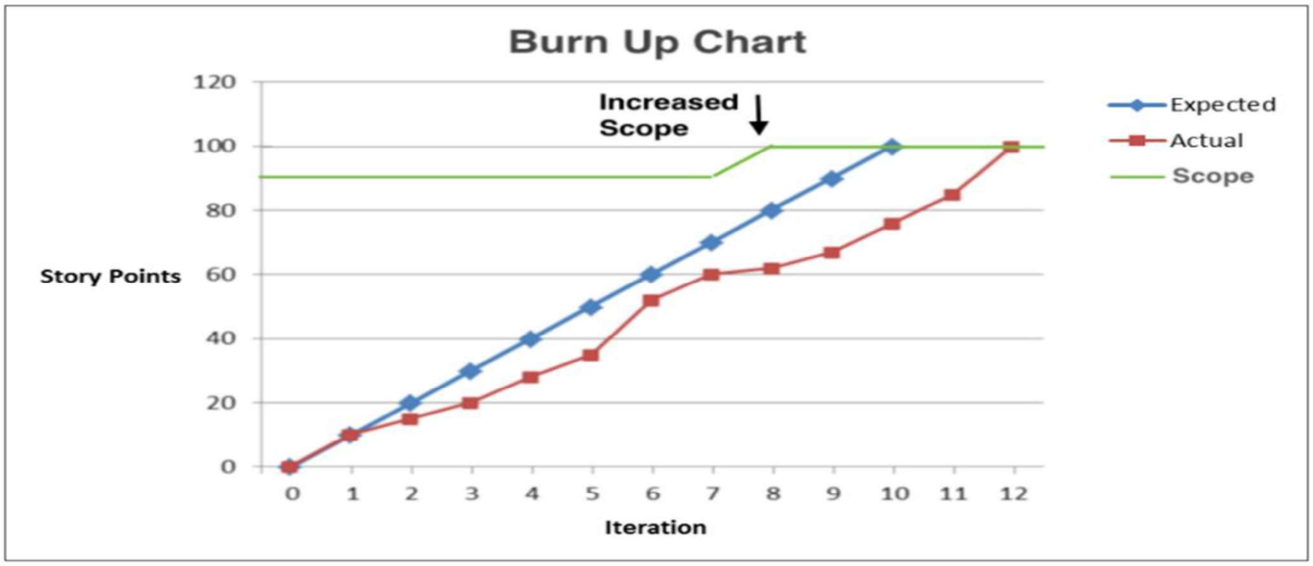
- Shows the health of backlog in as total number of Story Points available in backlog Vs Average Velocity.
- The speedometer shows the current backlog health is below 1 time of Average Velocity, at least 1 time of Average velocity, at least 2 times of average velocity or more than 3 times of average velocity.



Burn Up Chart:

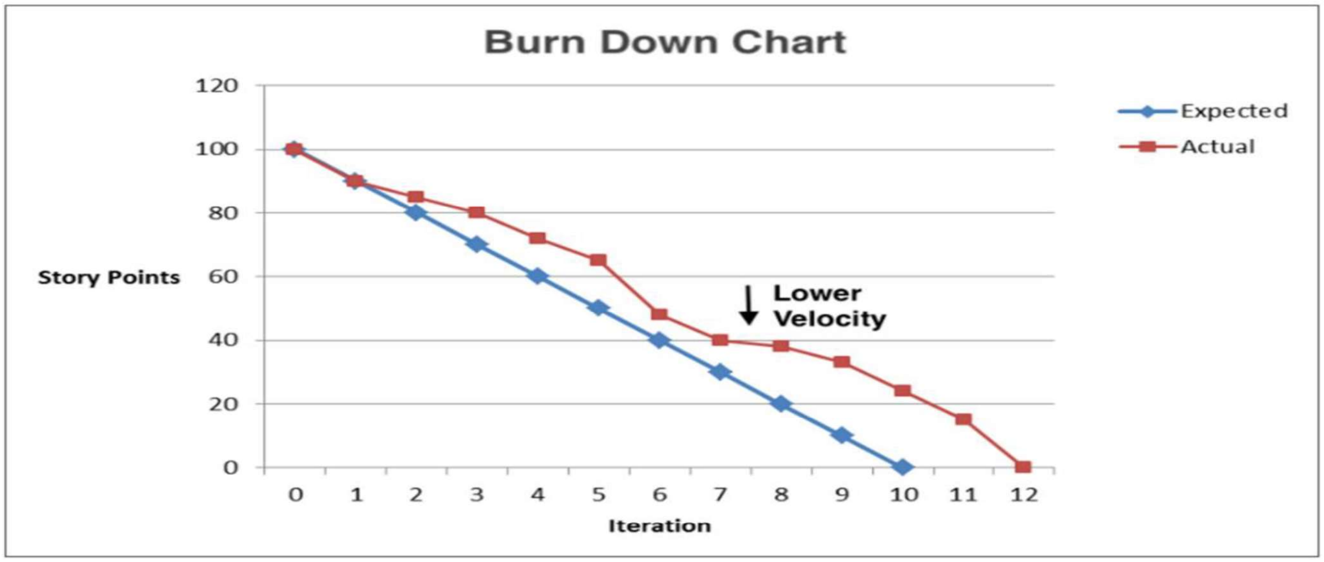
- The burn-up chart shows how much work has been completed, and the total amount of work.
- A Burn Up Chart is a tool used to track how much work has been completed and show the total amount of work for a project.

- The completed work and total work are shown on the vertical axis in whatever units a project team feels works best, i.e., workhours, workdays, story points, or any other work unit.
- The horizontal access displays time, usually in days, weeks, or iterations (sprints).



Burn Down Chart:

- A burn down chart shows how much work is remaining to be done on the project.
- Burn Down Charts are simple and easy for project members and clients to understand. A line representing the remaining project work slowly decreases and approaches zero over time.
- However, this type of chart doesn't show clearly the effects of scope change on a project.
- If a client adds work mid-project the scope change would appear as negative progress by the development team on a Burn Down Chart.



Roles and responsibilities:

I. **Development Team:**

- ✓ They are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality.
- ✓ Development Teams are cross-functional, with all the skills as a team necessary to create a product Increment.
- ✓ Scrum recognizes no titles for Development Team members, regardless of the work being performed by the person.
- ✓ Scrum recognizes no sub-teams in the Development Team, regardless of domains that need to be addressed like testing, architecture, operations or business analysis.

II. **Product Owner:**

- ✓ Expressing Product Backlog items clearly.
- ✓ Ordering the Product Backlog items to best achieve goals and missions.
- ✓ Optimizing the value of the work the Team performs.
- ✓ Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Team will work on further.

III. **Scrum Master:**

- ✓ Ensure the process runs smoothly.
- ✓ Remove obstacles that impact productivity.
- ✓ Organize critical events and meetings.