Contents

[Acronyms 7](#_Toc154664487)

[TIMWOOD 7](#_Toc154664488)

[INVEST 8](#_Toc154664489)

[DEEP 8](#_Toc154664490)

[MoSCoW 8](#_Toc154664491)

[SAMI 9](#_Toc154664492)

[EVM (Earned Value Management) 9](#_Toc154664493)

[PDEL (Plan, Develop, Evaluate, Learn) 9](#_Toc154664494)

[Formulas 9](#_Toc154664495)

[NPV (Net Present Value) 9](#_Toc154664496)

[NPV vs. IRR 9](#_Toc154664497)

[CPI (Cost Performance Index) 10](#_Toc154664498)

[SPI (Schedule Performance Index) 10](#_Toc154664499)

[IRR (Internal Rate of Return) 10](#_Toc154664500)

[ROI (Return on Investment) 10](#_Toc154664501)

[CV (Cost Variance) 10](#_Toc154664502)

[SV (Schedule Variance) 10](#_Toc154664503)

[Process Cycle Efficiency 11](#_Toc154664504)

[Acceptance Test Driven 11](#_Toc154664505)

[Acceptance Criteria 11](#_Toc154664506)

[Adaptive / Situational Leadership Stages 11](#_Toc154664507)

[Agile Framework 12](#_Toc154664508)

[Agile Manifesto 12](#_Toc154664509)

[4 Values 12](#_Toc154664510)

[12 Principles 12](#_Toc154664511)

[Agile Triangle 12](#_Toc154664512)

[Agile Constraints 12](#_Toc154664513)

[Agile Planning Levels 13](#_Toc154664514)

[Agile Tailoring 13](#_Toc154664515)

[Affinity Diagram 13](#_Toc154664516)

[Affinity Estimating 13](#_Toc154664517)

[Artifacts (3 main) 13](#_Toc154664518)

[Asynchronous Continuous Integration: 14](#_Toc154664519)

[Backlog 14](#_Toc154664520)

[Product Backlog 14](#_Toc154664521)

[Backlog Refinement 14](#_Toc154664522)

[Backlog - 3 Key Elements for Creating 15](#_Toc154664523)

[Backlog – 3 Types of Changes 15](#_Toc154664524)

[Risk Adjusted Backlog 15](#_Toc154664525)

[Barely sufficient and Just-in- time documentation 16](#_Toc154664526)

[Business value delivered chart 16](#_Toc154664527)

[Broken Comb 16](#_Toc154664528)

[Burnup Chart 16](#_Toc154664529)

[2 Types of Knowledge 16](#_Toc154664530)

[1. Tacit knowledge 16](#_Toc154664531)

[2. Explicit Knowledge 16](#_Toc154664532)

[3 Pillars of Scrum 17](#_Toc154664533)

[3 Cs (Card, Conversation, & Confirmation) 17](#_Toc154664534)

[3 Levels of Active Listening 17](#_Toc154664535)

[4 parts of Agile Hierarchy 17](#_Toc154664536)

[5 Stages of Team Development (Tuckman Model) 18](#_Toc154664537)

[5 Values of Scrum 18](#_Toc154664538)

[5 Levels of Conflict 18](#_Toc154664539)

[Level 1: Problem to Solve: 18](#_Toc154664540)

[Level 2: Disagreement 18](#_Toc154664541)

[Level 3: Contest 19](#_Toc154664542)

[Level 4: Crusade 19](#_Toc154664543)

[Level 5: World War 19](#_Toc154664544)

[7 Wastes in Agile Project Management 19](#_Toc154664545)

[Chief Programmer and the Domain Expert - FDD 19](#_Toc154664546)

[Conflict Resolution Methods 19](#_Toc154664547)

[Withdraw/Avoid 19](#_Toc154664548)

[Smooth/Accommodate 20](#_Toc154664549)

[Compromise/Reconcile 20](#_Toc154664550)

[Force/Direct 20](#_Toc154664551)

[Collaborate/Problem Solve 20](#_Toc154664552)

[Cockburn 7 Success Factors 20](#_Toc154664553)

[Collaboration Game 21](#_Toc154664554)

[Collective Code Ownership 21](#_Toc154664555)

[Common Cause/Special Cause Variation 21](#_Toc154664556)

[Communication Method Effectiveness 22](#_Toc154664557)

[Continuous Integration 22](#_Toc154664558)

[Continuous Deployment 22](#_Toc154664559)

[Cost of Change 22](#_Toc154664560)

[Cumulative Flow Diagram (CFD) 22](#_Toc154664561)

[Crystal family 23](#_Toc154664562)

[Four levels of criticality 23](#_Toc154664563)

[Five Color (Team Size) 23](#_Toc154664564)

[Cycle Time 23](#_Toc154664565)

[Daily Standup 23](#_Toc154664566)

[Daily Scrum Meeting Rules 24](#_Toc154664567)

[Daily Standup Facilitator 24](#_Toc154664568)

[Defined Process 25](#_Toc154664569)

[Definition of Done 25](#_Toc154664570)

[Definition of Ready 25](#_Toc154664571)

[Double Loop Learning 25](#_Toc154664572)

[Dynamic Scope Contract Type 26](#_Toc154664573)

[Effectiveness Vs. Efficiency 26](#_Toc154664574)

[Embellishment 26](#_Toc154664575)

[Emergent Leadership 27](#_Toc154664576)

[Empirical Process 27](#_Toc154664577)

[Escaped Defects 27](#_Toc154664578)

[ESPV 27](#_Toc154664579)

[Epic 28](#_Toc154664580)

[Explicit Knowledge 28](#_Toc154664581)

[Exploratory Testing 28](#_Toc154664582)

[Fast Failure 29](#_Toc154664583)

[Feature Driven Development 29](#_Toc154664584)

[Fishbowl Window 29](#_Toc154664585)

[Five Whys 29](#_Toc154664586)

[Fixed-Price Increments 29](#_Toc154664587)

[Feeding Buffers 30](#_Toc154664588)

[Flow Master/Manager (Kanban) 30](#_Toc154664589)

[Generalizing Specialist (T-shaped professional) 30](#_Toc154664590)

[Gulf of Evaluation 31](#_Toc154664591)

[Hardening Iteration or Iteration H 31](#_Toc154664592)

[Health Check Questionnaire 31](#_Toc154664593)

[Heuristics 31](#_Toc154664594)

[I-Shaped professional 31](#_Toc154664595)

[Incremental Design 32](#_Toc154664596)

[Intrinsic quality 32](#_Toc154664597)

[Iteration 0 32](#_Toc154664598)

[Iterations 32](#_Toc154664599)

[Iteration Burn-Down Chart 32](#_Toc154664600)

[Kaizen 32](#_Toc154664601)

[Kanban 33](#_Toc154664602)

[Kanban Benefits 33](#_Toc154664603)

[Flexibility 33](#_Toc154664604)

[Visibility 33](#_Toc154664605)

[Continuous Improvement 33](#_Toc154664606)

[Kanban Order of Practice 34](#_Toc154664607)

[Six Practices of Kanban 34](#_Toc154664608)

[Kanban Roles 34](#_Toc154664609)

[The Service Delivery Manager (SDM) 34](#_Toc154664610)

[The Service Request Manager (SRM) 34](#_Toc154664611)

[The Kanban Team 34](#_Toc154664612)

[The Product Owner 35](#_Toc154664613)

[The Stakeholders 35](#_Toc154664614)

[Kano model analysis 35](#_Toc154664615)

[Key Performance Indicators (KPI) 35](#_Toc154664616)

[Levels in the Agile Adoption Framework 36](#_Toc154664617)

[Large Scale Scrum or Less 36](#_Toc154664618)

[Last Responsible Moment 36](#_Toc154664619)

[Lean Portfolio 36](#_Toc154664620)

[Lessons Learned 36](#_Toc154664621)

[Low-tech, high-touch tools 37](#_Toc154664622)

[Mad Sad Glad Retrospective 37](#_Toc154664623)

[Maximizing work not Done 37](#_Toc154664624)

[Mobbing 38](#_Toc154664625)

[MSA – Master Service Agreement 38](#_Toc154664626)

[Multitasking 38](#_Toc154664627)

[MVP 39](#_Toc154664628)

[Osmotic Communication 39](#_Toc154664629)

[Pair Programming 39](#_Toc154664630)

[Participatory Decision Making 40](#_Toc154664631)

[Planning Poker 40](#_Toc154664632)

[Pre-Mortem 40](#_Toc154664633)

[Predictive Project Management Approach 41](#_Toc154664634)

[Project Charter 41](#_Toc154664635)

[Project Tweet 41](#_Toc154664636)

[Product Roadmap 41](#_Toc154664637)

[Primary Goal of Lean 41](#_Toc154664638)

[Project Approaches 42](#_Toc154664639)

[Agile Approach 42](#_Toc154664640)

[Predictive Approach 42](#_Toc154664641)

[Prune the Product Tree 42](#_Toc154664642)

[Ranking Techniques 43](#_Toc154664643)

[Weighted prioritization technique 43](#_Toc154664644)

[Simple ranking technique 43](#_Toc154664645)

[Systems Thinking Approach 43](#_Toc154664646)

[Red Zone 43](#_Toc154664647)

[Red, Green, and Refactor 43](#_Toc154664648)

[Regulatory Compliance – Just Because 43](#_Toc154664649)

[Release burndown 44](#_Toc154664650)

[Remember the Future 44](#_Toc154664651)

[Retrospective Stages 44](#_Toc154664652)

[Risk Based Spike 45](#_Toc154664653)

[Risk Identification Techniques 45](#_Toc154664654)

[Risk Mitigation Strategies 45](#_Toc154664655)

[Risk Reward Contract Relationship 45](#_Toc154664656)

[Risk Register 46](#_Toc154664657)

[Rolling Wave Planning 46](#_Toc154664658)

[Sailboat Retrospective 46](#_Toc154664659)

[Speed boat (Collective Workshop) 47](#_Toc154664660)

[Servant Leadership 47](#_Toc154664661)

[Story Map 47](#_Toc154664662)

[Scrum Master 47](#_Toc154664663)

[Scrum Team Members 48](#_Toc154664664)

[ScrumBan 48](#_Toc154664665)

[Shu-Ha-Ri 48](#_Toc154664666)

[Sashimi 48](#_Toc154664667)

[SMART Goals 48](#_Toc154664668)

[Spike 49](#_Toc154664669)

[Sprint Planning – Acceptance Criteria 49](#_Toc154664670)

[Sprint Review Meeting 49](#_Toc154664671)

[Swarming 50](#_Toc154664672)

[Swarming vs. Mobbing vs. Pairing 50](#_Toc154664673)

[Stakeholder meetings 50](#_Toc154664674)

[Systems Thinking 50](#_Toc154664675)

[Benefits of systems thinking 50](#_Toc154664676)

[Components of systems thinking 51](#_Toc154664677)

[Steps of systems thinking 51](#_Toc154664678)

[Synchronous Continuous Integration Steps 51](#_Toc154664679)

[Task Switching 51](#_Toc154664680)

[Tabaka Self-Assessment Model 51](#_Toc154664681)

[Tacit Knowledge 51](#_Toc154664682)

[Target Date 51](#_Toc154664683)

[Team Augmentation 52](#_Toc154664684)

[TDD - Test First Development 52](#_Toc154664685)

[Test-driven development 52](#_Toc154664686)

[Training Needs 52](#_Toc154664687)

[Trend Analysis 52](#_Toc154664688)

[Triple Constraint in Project Management 53](#_Toc154664689)

[Triple Nickel 53](#_Toc154664690)

[Tuckman Ladder Model 53](#_Toc154664691)

[User Story 54](#_Toc154664692)

[Value Stream Map 54](#_Toc154664693)

[Variance Analysis 54](#_Toc154664694)

[Wideband delphi Technique 54](#_Toc154664695)

[Working Agreement 54](#_Toc154664696)

[Working Agreement/Team Charter 54](#_Toc154664697)

[Working Software 55](#_Toc154664698)

[Workshop 55](#_Toc154664699)

[XP Coach 55](#_Toc154664700)

[XP Core Value 55](#_Toc154664701)

[XP System Metaphor 55](#_Toc154664702)

[XP Team Members 55](#_Toc154664703)

[XP – 4 Basic Activities 55](#_Toc154664704)

# Acronyms

## TIMWOOD

acronym TIMWOOD is used to describe the 7 wastes in Lean manufacturing or production. The letters stand for:

* T: Transport - unnecessary movement of materials or products
* I: Inventory - excess inventory or work-in-progress (WIP)
* M: Motion - unnecessary movement of people or equipment
* W: Waiting - delays in the production process or waiting for materials or information
* O: Overproduction - producing more than what is required by the customer
* O: Overprocessing - using more resources than necessary to produce the desired output
* D: Defects - producing defective products or requiring rework

## INVEST

* I – Independent - The PBI should be self-contained, in a way that there is no inherent dependency on another PBI.
* N – Negotiable - PBIs are not explicit contracts and should leave space for discussion.
* V – Valuable - A PBI must deliver value to the stakeholders.
* E – Estimable - You must always be able to estimate the size of a PBI.
* S – Small - PBIs should not be so big as to become impossible to plan/task/prioritize within a level of accuracy.
* T – Testable - The PBI or its related description must provide the necessary information to make test development possible.

## DEEP

DEEP - Detailed, estimable, emergent, prioritized

* **Detailed Appropriately -** User stories on the product backlog that will be done soon need to be sufficiently well understood to be completed in the coming sprint. Stories that will not be developed for a while should be described with less detail.
* **Estimated** - The product backlog is more than a list of all work to be done; it is also a useful planning tool. Because items further down the backlog are not as well understood (yet), the estimates associated with them will be less precise than estimates given items at the top.
* **Emergent -** A product backlog is not static. It will change over time. As more is learned, user stories on the product backlog will be added, removed, or reprioritized.
* **Prioritized -**The product backlog should be sorted with the most valuable items at the top and the least valuable at the bottom. By always working in priority order, the team is able to maximize the value of the product or system being developed.

## MoSCoW

The **MoSCoW** prioritization technique is one of the most used & popular prioritization techniques where the letters mean:

|  |  |
| --- | --- |
| M: Must Have, | - Must have: These are the requirements that are essential and must be delivered for the success of the project. Without these requirements, the project cannot be considered complete. |
| S: Should Have | - Should have: These are the requirements that are important but not critical for the success of the project. They can be delivered after the must-have requirements. |
| C: Could Have | - Could have: These are the requirements that are desirable but not critical for the success of the project. They can be delivered after the should-have requirements. |
| W: Would like to Have (Won’t have) | - Won't have: These are the requirements that are not necessary for the success of the project and will not be delivered in the current project scope. |

## SAMI

SAMI – Sidky Agile Maturity Index - **Collaborative, Evolutionary, Integrated, Adaptive, Encompassing**

The Sidky agile measurement index (SAMI) is a 5-step value-based roadmap to agility. The SAMI is designed to help guide organizations seeking to become more agile. Each of SAMI's 5 steps (collaborative, evolutionary, integrated, adaptive, and encompassing) aims to instill a new value in teams and organizations.

## EVM (Earned Value Management)

 EVM can’t tell the team what they are providing is valuable to the project. EVM is good for tracking how a team’s outputs compare to a plan or financial target but does not necessarily indicate if what is being provided is valuable. Given Agile’s focus is providing value for the customer, this can make EVM a less desirable metric.

## PDEL (Plan, Develop, Evaluate, Learn)

The equivalent for the “kaizen” in agile is Plan-Develop-Evaluate-Learn cycle.

The PDEL cycle can be a useful framework for Agile teams to follow as they seek to continuously improve their processes and practices. The cycle involves four key stages:

1. Plan: Define what you want to achieve and create a plan for how to achieve it.
2. Develop: Execute your plan and develop your product or service.
3. Evaluate: Assess your progress and determine how well you are meeting your goals.
4. Learn: Reflect on what you have learned and use that knowledge to improve your future efforts.

By following this cycle, Agile teams can identify areas for improvement, test out new ideas, and continuously evolve their processes and practices to deliver greater value and achieve their goals.

# Formulas

## NPV (Net Present Value)

You will need to know the discount rate and the future cash flow plus the initial investment. [The formula for calculating the net present value (NPV) of an individual cash flow is NPV = F / [ (1 + i)^n ]](https://www.bing.com/ck/a?!&&p=76e12e98c8e22d92JmltdHM9MTcwMzExNjgwMCZpZ3VpZD0yYWZjMWVjMi1lZDM4LTYwODctMDJiYy0wZGUwZWM3ODYxNzYmaW5zaWQ9NTg4OA&ptn=3&ver=2&hsh=3&fclid=2afc1ec2-ed38-6087-02bc-0de0ec786176&psq=NPV+formula&u=a1aHR0cHM6Ly9jb3Jwb3JhdGVmaW5hbmNlaW5zdGl0dXRlLmNvbS9yZXNvdXJjZXMvdmFsdWF0aW9uL25wdi1mb3JtdWxhLw&ntb=1). [To calculate NPV, start with the net cash flow (earnings) for a specific time expressed as a dollar amount. Divide that by the product of 1 plus the discount rate or interest rate (i) expressed as a decimal](https://www.bing.com/ck/a?!&&p=3533d2a53ba6ee1cJmltdHM9MTcwMzExNjgwMCZpZ3VpZD0yYWZjMWVjMi1lZDM4LTYwODctMDJiYy0wZGUwZWM3ODYxNzYmaW5zaWQ9NTg5MA&ptn=3&ver=2&hsh=3&fclid=2afc1ec2-ed38-6087-02bc-0de0ec786176&psq=NPV+formula&u=a1aHR0cHM6Ly93d3cuYnVzaW5lc3NpbnNpZGVyLmNvbS9wZXJzb25hbC1maW5hbmNlL25wdg&ntb=1). [NPV can also be calculated with the formula NPV = ⨊ (P/ (1+i)t) – C, where P = Net Period Cash Flow, i = Discount Rate (or rate of return), t = Number of time periods, and C = Initial Investment](https://www.bing.com/ck/a?!&&p=51eb079f68cf20e8JmltdHM9MTcwMzExNjgwMCZpZ3VpZD0yYWZjMWVjMi1lZDM4LTYwODctMDJiYy0wZGUwZWM3ODYxNzYmaW5zaWQ9NTg5Mw&ptn=3&ver=2&hsh=3&fclid=2afc1ec2-ed38-6087-02bc-0de0ec786176&psq=NPV+formula&u=a1aHR0cHM6Ly93d3cud2lraWhvdy5jb20vQ2FsY3VsYXRlLU5QVg&ntb=1), [NPV is used by companies to decide whether to invest in a project](https://www.bing.com/ck/a?!&&p=9bb446dde4ef391bJmltdHM9MTcwMzExNjgwMCZpZ3VpZD0yYWZjMWVjMi1lZDM4LTYwODctMDJiYy0wZGUwZWM3ODYxNzYmaW5zaWQ9NTg5NQ&ptn=3&ver=2&hsh=3&fclid=2afc1ec2-ed38-6087-02bc-0de0ec786176&psq=NPV+formula&u=a1aHR0cHM6Ly93d3cuaW52ZXN0b3BlZGlhLmNvbS9hc2svYW5zd2Vycy8wMjExMTUvd2hhdC1mb3JtdWxhLWNhbGN1bGF0aW5nLW5ldC1wcmVzZW50LXZhbHVlLW5wdi1leGNlbC5hc3A&ntb=1).

Note: you will not have to calculate NPV on the test. You simply need to know that you have to compare the NPV of two project and choose the one with the higher value.

## NPV vs. IRR

IRR is a superior measure and should be used for making decisions in benefit-cost analysis. Project A has the highest IRR value. The internal rate of return is a metric used in financial analysis to estimate the profitability of potential investments. The internal rate of return is a discount rate that makes the net present value (NPV) of all cash flows equal to zero in a discounted cash flow analysis.

## CPI (Cost Performance Index)

EV/AC (Earned Value/Actual Costs)

CPI is Earned Value divided by Actual Cost (CPI=EV / AC). CPI measures the cost performance of a project; is the project budget being spent as planned? In order to calculate CPI you need two pieces of information, the earned value and the actual cost of the project. There are three possible results when calculating this: CPI = 1 is good and means funds are being used as planned; CPI >1 is also good and means the funds are being used more efficiently than planned; and CPI <1 is bad and means the funds are being overspent.

## SPI (Schedule Performance Index)

EV/PV (Earned Value/Planned Value)

SPI is Earned Value divided by Planned Value (SPI=EV / PV). This formula measures the schedule performance of a project, is the project performing as planned? In order to calculate SPI you need two pieces of information, the earned value and the planned value of the project. There are three possible results when using this formula: CPI = 1 is good and shows the project is progressing as planned; CPI >1 is also good and shows the project is progressing at a faster rate than planned; and CPI <1 is bad and shows the project tis progressing at a slower rate than planned.

The schedule performance index (SPI) of 1.14 means the team is working faster than expected.

## IRR (Internal Rate of Return)

IRR (Internal Rate of Return) is most like the concept of interest earned on a savings account. IRR helps measure the value of a project over time, like a savings account builds interest over time. IRR is used as a capital project budgeting metric to determine if an investment should be made. It looks at the present value of the cash flows as compared to the initial investment which results in an IRR value. For example, if as a Project Manager you need to compare two or more projects to determine which one would be the better investment for your organization you can use IRR to do this. If you are given the IRR for three projects; Project A IRR =25%, Project B IRR = 30%, and Project C IRR = 12% you can determine that Project B is the better investment for the organization because it has the largest IRR value.

## ROI (Return on Investment)

ROI is used to evaluate the money gained or lost in relation to the money invested in a project. ROI is also often referred to as gain/loss, profit/loss, or net income/loss. A Project Manager can use the ROI of one or more projects to determine which project is the better investment.

## CV (Cost Variance)

CV is the Earned Value minus the Actual Cost (CV=EV-AC) of a project. This formula measures the cost performance of a project and looks at whether the project is on budget or not. To calculate CV, you need two pieces of information, the earned value and the actual cost of the project. If a CV result is a negative number the project is over budget, which is bad. If a CV result is a positive number the project is under budget, which is good. If CV is zero, then the project is exactly on budget.

## SV (Schedule Variance)

SV is the Earned Value minus the Planned Value (SV=EV-PV) of a project. This formula measures the schedule performance of a project and looks at whether the project is behind schedule or ahead of schedule. To calculate SV, you need two pieces of information, the earned value, and the planned value of the project. If an SV result is a negative number, then the project is behind schedule, which is bad. If an SV result is a positive number, then the project is ahead of schedule, which is good. If SV is zero, then the project is exactly on schedule.

# Process Cycle Efficiency

The formula for finding process cycle efficiency is: Total value-added time / total cycle time. In this question, the value-added time is 2 hours, and the total cycle time is 2 minutes + 8 minutes + 120 minutes = 130 minutes. So the correct answer is 120 / 130 = 92%.

Bottom of Form

# Acceptance Test Driven

In this type of acceptance-test-driven development approach**, the entire team gets together and discusses the acceptance criteria** for the work product.

The main point here is that the entire team's input is taken into consideration, and the whole team works together to finalize or arrive at the acceptance criteria for our product. After this, the team creates the test, which allows the team to write just enough code and automated tests to meet the acceptance criteria. In ATDD, the entire team gets together and discusses the acceptance criteria for a work product. Then the team creates the tests, which allow the team to write just enough code to meet the criteria.

# Acceptance Criteria

A story without acceptance criteria should not be worked. The best choice is to stop working on the item, as it should not have been estimated without clear criteria. Without clear acceptance criteria it will be hard, if not impossible, for the team to know what they need to accomplish. A Scrum Master is not responsible for acceptance criteria. The product owner could add acceptance criteria, but the item should not have been estimated without it. The item should be re-estimated, but work should not necessarily continue, especially if the estimate is different.Bottom of Form

# Adaptive / Situational Leadership Stages

* **Directing** - Directing is the initial or basic level of leadership style. Nearly all new employees need a more directive style of leadership. They are very formative with little or no experience in their new roles. In the formative stage the follower is characterized by low competence and high commitment, the inability to comply, with possible feelings of insecurity.
* **Coaching** - Coaching is for followers who have developed some competence along with an improved commitment. The follower is not totally confident in their abilities but is getting there.
* **Supporting** - Supporting addresses the follower who is now competent at the job but remains somewhat inconsistent and is not yet fully committed. The follower may be uncooperative or performing as little work as possible, despite their competence with the tasks.
* **Delegating** - The ultimate goal is Delegating: to create a follower who feels fully empowered and competent enough to take the ball and run with it, with minimal supervision. The follower is highly competent, highly committed, motivated, and empowered.

# Agile Framework

The life cycle category of project management that uses small frequent deliveries is known as the Agile methodology. This approach to project management is based on the Agile Manifesto, which values individuals and interactions, working software, customer collaboration, and responding to change. The Agile methodology is characterized by a flexible and iterative approach to project management, with a focus on delivering value early and often. The team works closely together, with frequent communication and collaboration between team members and the customer or stakeholders.

# Agile Manifesto

## 4 Values

* Individuals and interactions over processes and tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiation
* Responding to change over following a plan

## 12 Principles

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

# Agile Triangle

Three parameters of the Agile Triangle –**Value, Quality, constraints** The measures of the Agile triangle are valued (to the customer), quality (required to deliver continuous value to the customer), and constraints (scope, schedule, and cost).

# Agile Constraints

Scope, Schedule, Cost

# Agile Planning Levels

**Agile planning should be done at multiple levels - these levels include.**

* **strategic -**At the strategic level, project planning should align with the overall organizational goals and strategic objectives.
* **release -**Plan for the appropriate releases. This will be much broader and will consider the resources, schedule, and risks at a release level.
* **iteration -**Given the dynamic nature of artificial intelligence technologies and the potential for changing requirements, it may be beneficial to adopt an iterative or incremental approach to project planning.
* **daily -**Planning should involve defining the specific tasks, responsibilities, and timelines for individual team members.

In agile teams use rolling wave planning and progressive elaboration to balance predictability of outcomes with ability to exploit opportunities.

# Agile Tailoring

As an PMI-ACP you should tailor and adapt the project process by periodically reviewing and integrating team practices, organizational culture, and delivery goals to ensure team effectiveness within established organizational guidelines and norms.

# Affinity Diagram

Affinity diagram helps the project team organize a large number of ideas into their natural relationships. Such diagrams are the result of a brainstorming session. These are used to generate, organize, and consolidate information related to a product, process, complex issue, or problem.

# Affinity Estimating

**Affinity Estimating** is a technique many Agile teams use to quickly and easily **estimate** a large number of user stories in story points. This is a great technique if you're just starting a project and have a backlog that hasn't been estimated yet, or in preparation for release planning.

Affinity estimates, also known as relative estimation, is a technique used in Agile teams to estimate the size or effort required to complete user stories or tasks. It involves comparing the size or effort of one item to another, rather than assigning absolute values.

Affinity estimating is a technique that involves grouping items into similar categories or collections.

Affinity estimates are useful in Agile teams as they allow for quick and collaborative estimation, encourage team engagement and discussion, and provide a relative measure of effort that can be used for planning and prioritization of user stories or tasks in Agile frameworks such as Scrum or Kanban.

# Artifacts (3 main)

In Scrum, there are three main artifacts that are used to facilitate transparency, inspection, and adaptation in the Agile development processes - product backlog, sprint backlog and increments.

In Scrum, there are three main artifacts that are used to facilitate transparency, inspection, and adaptation in the Agile development process:

1. Product Backlog: This artifact represents the ordered list of all the items (features, enhancements, bug fixes, etc.) that need to be developed or delivered in the product. It is owned and managed by the Product Owner, who is responsible for prioritizing and refining the items based on business value, market needs, and customer feedback. The Product Backlog is dynamic and constantly evolving, and it serves as the single source of truth for the team's work.

2. Sprint Backlog: This artifact represents the set of items from the Product Backlog that are selected and committed by the Development Team to be worked on during a specific Sprint. The Sprint Backlog is created during the Sprint Planning event and is owned and managed by the Development Team. It includes the tasks, user stories, or other work items that the team plans to complete during the Sprint, along with their estimates, dependencies, and progress tracking.

3. Increment: This artifact represents the sum of all the work completed by the Development Team during a Sprint, including all the items that were part of the Sprint Backlog and have been Done according to the Definition of Done (DoD). The Increment is the potentially releasable product increment that is ready to be delivered to customers or stakeholders at the end of each Sprint. It provides transparency on the progress and quality of the work completed by the team and serves as the basis for feedback and inspection in the Sprint Review event.

# Asynchronous Continuous Integration:

you code, test, and fire the integration script, but you move to the next step of coding and do not wait for integration to finish.

# Backlog

Whenever you work on a large Scrum project, ensure that there is one product backlog that contains the work necessary to bring the product to life. Avoid team- or component-specific backlogs that translate product requirements into subsystem or component requirements. So employ one product backlog, extend the grooming horizon, and consider providing team-specific views into the backlog.

## Product Backlog

A prioritized list of work for the development team that is derived from the roadmap and its requirements.

## Backlog Refinement

The right time for the team to conduct the backlog refinement meeting in Scrum is during the sprint, typically after the current sprint review and before the next sprint planning. Therefore, the backlog refinement meeting can be conducted during the middle of the sprint, in this case in the middle of the 2nd sprint.

This meeting helps the development team to refine the product backlog items and prepare them for the upcoming sprints, ensuring that the team has a well-prepared backlog for the next sprint planning.

The backlog refinement meeting, also known as backlog grooming, should involve the product owner, the development team, and the Scrum Master. Other stakeholders may also be invited to attend as needed, but the core group should be the product owner and the development team.

## Backlog - 3 Key Elements for Creating

There are three key elements used by the agile teams for analyzing the product and creating a backlog:

1. People involved, which basically represents the identification of the personas or roles existing in the customer environment
2. Identification of the functions performed by these identified roles or personas
3. Break down functionalities into implementable pieces i.e. user stories. Out of the given options, Identification of the personas or roles existing in the customer environment is the only key element used by the agile teams for analyzing the product and creating a backlog.

## Backlog – 3 Types of Changes

There are **three types of changes** that may need to be made to the backlog, during project execution,

1. New stories maybe added
2. Existing stories may be reprioritized or removed
3. Stories may be sliced into smaller chunks or resized.

**Customer**should add or remove user stories and the development team should slice stories from the product backlog

The customer or product owner is responsible for adding or removing user stories based on changing business needs, customer feedback, and market conditions. The product owner works closely with stakeholders to prioritize the product backlog items based on business value, risks, dependencies, and other factors. The product owner also ensures that the product backlog items are well-defined, testable, and aligned with the product vision and strategy.

The development team is responsible for slicing user stories from the product backlog into smaller, manageable, and testable pieces of work that can be completed within a sprint. The development team collaborates with the product owner to clarify the requirements, assumptions, constraints, and acceptance criteria for each user story.

The development team also estimates the effort and complexity of each user story, based on their technical skills, experience, and knowledge. Finally, the development team commits to delivering the user stories in the sprint, based on their capacity and velocity.

## Risk Adjusted Backlog

A backlog that contains activities relating to managing risk in addition to the usual features associated with delivering value. Agile projects prioritize the backlog based on business value or perceived needs. Need to know the financial return on the whole project to distribute the return across features and come up with the risk-adjusted backlog.

# Barely sufficient and Just-in- time documentation

Documentation that is as minimal as possible to cover basic needs that is done only right before it is needed and not before. Barely sufficient documentation is focused on providing the least amount of information to get work estimated and completed. Just-in-time documentation is produced at the latest time possible without negatively impacting the team.

# Business value delivered chart

The entire enterprise (business, management, and development teams) needs the line of sight to velocity (points/time) dashboard-type view of work management which in other terms is a business value delivered chart.

# Broken Comb

The Broken Comb approach refers to a person with various specializations.

# Burnup Chart

For tracking the team’s progress, agile tends to focus on low-tech, high touch tools. Burn charts which come in 2 flavors, burndown and burnup can be used to show the team’s progress. **Burnup charts containing story points which are plotted on the y-axis and iterations over a time are plotted on x-axis.**

On a burnup chart, the x-axis typically represents time, and the y-axis represents the amount of work completed or the number of story points completed.

The time frame on the x-axis can be divided into days, weeks, sprints, or any other time interval that the team prefers to use.

The y-axis represents the amount of work completed and is measured in units of work, such as story points or tasks completed. The total amount of work to be completed is represented by the top line of the chart, also known as the "scope" or "backlog." The actual work completed is represented by the bottom line of the chart.

Also, a burnup chart can show scope changes if it is updated accordingly.

Typically, Burnup charts track the work that has been completed**. The big advantage of using a burn up chart is that it can show changes in scope, making the impact of those changes visible.**

# 2 Types of Knowledge

1. Tacit knowledge which refers to knowledge that is not easily codified or articulated, and is often based on personal experience, intuition, insights, or skills that are difficult to transfer through explicit means like documentation or written instructions. Tacit knowledge is subjective and context-specific, and is typically acquired through direct involvement in a specific domain or activity over time.
2. Explicit Knowledge which refers to knowledge that is codified, documented, and easily transferable through written or visual means. It is tangible and can be easily shared, stored, and accessed by others.

# 3 Pillars of Scrum

The three pillars of the Scrum methodology are transparency, inspection, and adaptation. These pillars provide the foundation for the Scrum framework and help ensure that the team can deliver high-quality, valuable software in a timely manner. The first pillar of Scrum is transparency, which means that all aspects of the project should be visible to everyone involved. This includes the project goals, progress, and any issues or impediments that may arise. Transparency helps ensure that the team is aligned on the project goals and can make informed decisions based on the project's status.

1. **Transparency** – Giving visibility to the significant aspects of the process to those responsible for the outcome.
2. Inspection – Timely checks on the progress toward a sprint goal to detect undesirable variances.
3. **Adaption** – Adjusting a process as soon as possible to minimize any further deviation of issues.

# 3 Cs (Card, Conversation, & Confirmation)

As the user stories assist the developers to be focused on the value being delivered to the customer or end users, they give the agile practitioners, particularly the developers, a blueprint for successful development. There are a lot of approaches in Agile for capturing user stories. Out of these available approaches, one is “three Cs”. A good user story is supposed to have at least three attributes - card, conversation, and confirmation. A user story is needed to be enough concise for documentation on a 3\*5 index card which is simply known as a card. These cards can be accessed by the team members or the product owner to have a conversation about the functionality or feature being suggested or discussed to make the story as complete and accurate as possible. Eventually, the user story is confirmed with the team using the acceptance criteria to develop and confirm a common understanding of each user story.

The main difference between traditional and agile approaches to capturing the needs is the conversion of the rough needs of the product into the user stories, agile, emphasizes the value being delivered to the customer or the end users. There are a lot of various approaches in Agile for capturing user stories. Out of these available approaches, one is “three Cs”. A good user story is supposed to have at least three attributes card, conversation, and confirmation. Therefore, applying the three Cs of user stories will help the project manager in documenting the product requirements roughly for further discussion and validation with project stakeholders.

# 3 Levels of Active Listening

1. Internal Listening — Listening to your inner voice. Interpret what has been listened to through our own lens and miss the real message.
2. Focused Listening — Listening intently to another person. Empathize with the speaker and understand their message correctly.
3. Global Listening — Listening to others in the context of their entire surroundings. In addition to focused listening, also pick up on subtle physical and environmental indicators.

# 4 parts of Agile Hierarchy

1. **Epic**
2. **Feature**
3. **User Story**
4. **Task**

# 5 Stages of Team Development (Tuckman Model)

1. **Forming**
2. **Storming**
3. **Norming**
4. **Performing**
5. **Adjourning**

# **5 Values of Scrum**

* **Commitment** - Commitment to getting the best outcome and achieving the goals of the Scrum Team in all Scrum events.
* **Focus** - Everyone must focus on the team's efforts and sprint backlog tasks.
* **openness** - Open to other opinions and advice, open to new ideas, open to finding better ways of working.
* **Respect** - Respect of yourself and others.
* **Courage** - The team members must have the audacity to do the right things in the projects by solving complex and unpredictable problems.

# **5 Levels of Conflict**

* **Level 1 (Problem to Solve)** – The language is friendly and mostly constructive. People back up with statements with facts. E.G. “Oh, I see what you are saying now. I still prefer the other approach, but I understand your suggestion.”
* **Level 2 (Disagreement)** – People start to include self-protection. E.G. “I know you think my idea won’t work, but we tried your suggestion last time, and there were a lot of problems.”
* **Level 3 (Contest)** – The language becomes distorted with over-generalizations and magnified positions. E.G. “If only he wasn’t on the team…”, “She always takes over the demo.”
* **Level 4 (Crusade)** – The conflict becomes more ideological and divided. E.G. “They’re just plain stupid”, and “It’s not worth talking to them.”
* **Level 5 (World War)** – The language is altogether combative, or the opposing people do not speak directly to each other. Only talking to those “on their side” and saying things such as, “It is us or them” and “We’ve got to beat them!”

## **Level 1: Problem to Solve**:

In level 1 of agile conflict, the agile team identifies the problem rising in the project. The team discusses and shares the problem among the team members and also along with the agile project leader. This level of conflict is constructive problem-solving

## **Level 2: Disagreement**

In level 2 of agile conflict, the conflict becomes a matter of disagreement among the agile team members. The agile team members begin to distance themselves from each other. Due to various problems, they stop taking the initiative for resolving the conflict. Although this level doesn’t witness an open war all the team members ideally wait for some other to enter and resolve the conflict.

## **Level 3: Contest**

In level 3 of agile conflict, multiple problems left unresolved cause difficult positions inside the agile team. Different groups within the same team are built up and every agile team member stands to support one or the other group. In that case, the conflict becomes like a contest, and more than resolving the issue the focus is more highlighted on winning rather than compromising.

## **Level 4: Crusade**

In level 4 of agile conflict the contest becomes a crusade. At this level, the groups started thinking that the other group will never change and therefore must be decimated. Positions get replaced and the only focus is on protecting one’s belonging group.

## **Level 5: World War**

In level 5 of agile conflict, the conflict becomes a war where no solution seems to be arising. Here, it literally heads to complete destruction by mutual consent. This is the highest difficulty level of agile conflict.

# 7 Wastes in Agile Project Management

The seven wastes in Agile project management are: **overproduction, overprocessing, waiting, excess inventory, unnecessary motion, defects, and unused talent.**

# Chief Programmer and the Domain Expert - FDD

The **chief programmer** determines which features will be designed and built in a two-week iteration. This person also defines the feature priorities and determines who will be involved on the feature team. A design review needs to be completed by the whole team before moving forward.

# Conflict Resolution Methods

In Problem Solving method, people involved in the conflict or having a difference in opinion, they come forward to discuss the problem at hand with a very open mind. They focus on resolving the conflict and finding the best alternative/solution for the team. They discuss by rising above personal emotions with the sole intention to finding what is best for the team. This leads to a win-win kind of an outcome. Here everyone collaborates.

## **Withdraw/Avoid**

In this conflict resolution technique, you avoid the conflict or retreat and allow the conflict to resolve itself. This is for when stakes are low, and the conflict is likely to disappear on its own. Use this technique in the following cases:

* Individuals involved in the conflict are not influential stakeholders.
* The issue does not require a time investment.
* An intense argument has already happened, and the individuals need time to cool off.
* You do not have enough information to pursue other techniques.

## **Smooth/Accommodate**

In this conflict resolution strategy, you find areas of agreement, try to smooth out the situation, and circumvent tough discussions. The smoothing technique gives more consideration to one party than the other. You downplay the seriousness and behave as if the conflict never existed. This technique is helpful in the following cases:

* You don’t have time to deal with it.
* You require a temporary solution to the problem.
* The conflict is minor and involves less influential stakeholders.

## **Compromise/Reconcile**

In this conflict management strategy, you take suggestions from both sides and partially satisfy them. This technique is useful when the stakeholders involved hold equal power. You may use this technique in the following cases:

* All parties involved need to win
* When you have an equal relationship with both parties
* Collaborative and forcing techniques have failed
* When you need a temporary solution

## **Force/Direct**

In this conflict management strategy, you agree with one party’s viewpoint and enforce their wishes. This is a win-lose situation and risks demoralizing the team. You can use this conflict resolution technique in the following cases: When you need a quick solution

* When you know that one party is right
* You do not have time to investigate
* When the conflict is not very important
* When the relationship with stakeholders is not essential

## **Collaborate/Problem Solve**

In a collaborative conflict resolution strategy, you discuss the issue with all parties and agree on a solution while considering multiple viewpoints. You may use this technique in the following cases:

* When incorporating multiple views
* If influential stakeholders are involved
* When a consensus is required
* If you want to distribute responsibility

# Cockburn 7 Success Factors

There are **seven success factors** or modes provided by Cockburn that have a significant contribution to the success of the team. These factors involve:

* learning desire and capability,
* being good at looking and analyzing around,
* feeling pride at work,
* being malleable,
* the desire to contribute,
* taking initiative,
* and being a good citizen.

He also described some factors for failure like committing mistakes, focusing on invention rather than research, preference to failure conservatively, being inconsistent, and being a creature of habit. In the given scenario, the focus of the team was to invent something new rather than learning and researching the existing approaches and that is the most likely reason for the failure of the team in the recognition of the issue.

# Collaboration Game

The best approach to reach consensus is to use a collaboration game, as it will be the most likely to bring the group to consensus. Not only can this help get to a solution but will help the team practice their collaboration skills.

Collaboration games are a valuable tool in Agile project management as they promote teamwork, communication, and collaboration among team members. Using a collaboration game will most likely bring the group to a consensus. Using collaboration games in Agile project management can help to create a positive and productive team environment, foster creativity and innovation, and ultimately lead to better project outcomes.

# Collective Code Ownership

Knowledge sharing is a key component of agile methods, and the practice of collective code ownership means that any developer can maintain any portion of the system.

Depending on any single person is always a high risk. Consider below questions for example.

* What happens if this 1 person leaves the organization?
* What if he is met with an accident and cannot work for the next 3-months?
* This 1 person is not showing the right attitude and doesn't listen to any team members or even to his or her manager?

All these above questions clearly highlight the risky nature of over-dependencies on an individual. This is the reason why an agile practitioner should always encourage collective code ownership.

# Common Cause/Special Cause Variation

* Common cause variation: Also known as random or systemic variation, common cause variation is inherent to a process and is considered normal. It arises from the natural variability of the process, which is typically caused by a combination of multiple factors acting together in a stable and predictable manner.
* Special cause variation: Also known as assignable or non-systemic variation, special cause variation is caused by specific identifiable factors that are not part of the normal process variability. Special cause variation is typically sporadic and unpredictable, and it can result in data points that fall outside the control limits in a control chart. Special cause variation may be caused by factors such as equipment malfunction, human error, material defects, or other unexpected events that disrupt the normal process flow.

The key difference between common cause and special cause variation is that common cause variation is inherent to the process and is expected, while special cause variation is caused by specific identifiable factors that are not part of the normal process variability and require investigation and corrective action. In process improvement efforts, it is important to distinguish between common cause and special cause variations to effectively identify and address the root causes of process performance issues.

# Communication Method Effectiveness

Good communication is highest in **information density and interactivity**. Information density is how much useful information can be conveyed in the communication and interactivity is how much the communication allows for a back-and-forth between the parties.

# Continuous Integration

Continuous integration (CI) is a software development practice that involves merging code changes from multiple developers into a shared repository regularly. The main purpose of CI is to minimize code conflicts and merge issues during the development process. By integrating changes frequently, developers can detect and resolve conflicts early, ensuring that the codebase remains stable and functional. CI often involves automated build and testing processes to validate the integration and identify any issues early on.

# Continuous Deployment

A strategy in software development where code changes to an application are automatically released into the production environment.

# Cost of Change

Increases as the project moves closer to a production release.

# Cumulative Flow Diagram (CFD)

Cumulative Flow Diagrams chart the number of tasks in each stage of your workflow at any given period along the process timeline.

A shallow area on a cumulative flow diagram indicates that the area is likely a bottleneck. A CFD is a tool commonly used in Agile project management to visualize the flow of work throughout the software development process. It provides a snapshot of the work in progress by showing how many items are in each stage of the workflow, from the backlog to completion. The horizontal axis shows the timeline, while the vertical axis shows the number of items in each stage of the workflow. A CFD is created by collecting data from the Kanban board or other tracking tools and plotting the data on a graph. The diagram can help teams identify bottlenecks and areas where work is getting stuck, and track progress towards project goals. By analyzing the CFD over time, the team can make data-driven decisions to optimize their workflow and improve their delivery process.

# Crystal family

The methodology chosen for each project depends on the impact that requirement and development defects will have combined with the size of the team.

The Crystal family of methodologies consists of a set of lightweight Agile development frameworks that are designed to be flexible and adaptable to a wide range of project types and team sizes. The core differences between the different members of the Crystal family (Crystal Clear, Crystal Yellow, Crystal Red, and Crystal Maroon) **are based on the level of risk, uncertainty, and criticality of the project, as well as the level of communication, collaboration, and documentation required.**

## Four levels of criticality

defined, based on what might be lost because of a failure in the produced system

* Comfort (C)
* Discretionary Money (D)
* Essential Money (E)
* Life (L)

## Five Color (Team Size)

* Crystal clear - smallest team size of 1-6
* Crystal Yellow - Small team size of 7-20
* Crystal Orange - Team size of 21 to 40
* Crystal Red. - Team size of 41 to 80
* Crystal Maroon - Team size of 81 to 200

# Cycle Time

Cycle time is **how long a project takes from starting the work to completion**—when the project is ready for delivery.

Cycle time in a user story is focused just on the coding and testing of a user story. The other stages are part of lead time, but not cycle time. Cycle time includes only the specific completion of an assignment, not the definition or deployment of the assignment.

Both lead and cycle time are valid metrics & can be used to express duration - lead time measures how long something takes to go through the entire process whereas Cycle time is a subset of lead time that measures how long something takes to go through part of the process.

Hence There is nothing wrong with the data shared by the team lead, as lead time is always greater than cycle time.

# Daily Standup

The Daily Scrum is a short, time-boxed daily meeting where the development team comes together to synchronize their work and plan for the day. During the Daily Scrum, team members typically answer the following three questions:

1. What did I do yesterday?
2. What am I going to do today?
3. Are there any impediments or issues that are blocking my progress?

The third question explicitly provides an opportunity for team members to report any issues or impediments they are facing that are hindering their progress towards the sprint goal.

This can include technical challenges, dependencies on other teams, resource constraints, or any other obstacles that are impacting the team's ability to deliver the sprint backlog items.

The Scrum Master, who is the servant-leader and facilitator of the Scrum process, can then help to address and resolve these impediments or escalate them to the appropriate stakeholders for resolution.

It's important for team members to be transparent and share any obstacles they are facing during the Daily Scrum, so that the team can collaboratively work towards resolving them and maintaining a high level of productivity throughout the sprint.

Daily stand up is not a status meeting but instead in this meeting the development team discuss about any of the impediments or problems which they are facing daily standup is the ideal meeting to report the problems which team should use to report problems.

# Daily Scrum Meeting Rules

* Same time and place
* Meeting should be held daily (hence the name)
* Time-boxed to 15 minutes
* All team members attend
* Only the Scrum Team may attend

# Daily Standup Facilitator

In Scrum, any member of the development team can facilitate the daily stand-up or daily Scrum. The Scrum Master is not the only person who can take on this responsibility. The team is considered self-organizing and cross-functional, and any team member can facilitate the daily stand-up based on the needs and dynamics of the team.

However, it is also important to note that the Scrum Master still has a responsibility to ensure that the team understands the purpose and benefits of the daily stand-up, and to coach the team on best practices if needed. The Scrum Master can provide support, guidance, and facilitation skills as needed, and work with the team to continuously improve the effectiveness of the daily stand-up.

In some cases, a development team member who has good facilitation skills and is comfortable with leading discussions may take on the role of facilitating the daily stand-up. This could be someone who has experience in leading meetings, has good communication skills, and is familiar with the purpose and format of the daily stand-up.

The key is to ensure that the daily stand-up is conducted effectively, with all team members actively participating, and that it serves its intended purpose of promoting communication, coordination, and alignment among team members. If a development team member takes on the role of facilitating the daily stand-up, it is important for them to keep the meeting focused, time-boxed, and inclusive, allowing all team members to share updates, discuss impediments, and plan their work for the day.

This point is not only important for the exam but also for the real-world team scrum implementation. There is general perception in most of the scrum project teams that it is only the scrum master who must facilitate the daily standup and others scrum ceremonies.

But it is clearly not the case and anyone from the development team with adequate facilitation skills can lead the stand up.

# Defined Process

A defined process is an approach used when the materials are known and understood, and the approach to the solution is known.

# Definition of Done

The best method is to ensure that the team regularly discusses the definition of done (DOD), so everyone is aligned with the expectations of what done means. The DoD is a shared agreement among the Agile team, including developers, testers, and other stakeholders, on the criteria that must be met before a user story or feature is considered completed. The DoD sets out the minimum requirements that a user story or feature must meet to be considered done and ready for delivery. The DoD helps ensure the team is on the same page and everyone knows what to expect from a completed user story or feature. It also helps prevent misunderstandings and ensures that the team is delivering quality work that meets the acceptance criteria.

# Definition of Ready

The "Definition of Ready" is a concept used in Agile development to ensure that a user story or backlog item is adequately prepared and ready for the development team to start working on it. It is a set of criteria or conditions that must be met before a user story is considered ready for development.

The Definition of Ready serves as a quality gate, ensuring that user stories are well-prepared and meet the team's standards before they are pulled into a sprint or assigned to the development team for implementation. It helps to prevent issues or delays during development by ensuring that the user stories are sufficiently refined, understood, and ready for the team to start working on them.

Definition of ready is **checklist for user centric requirement that has all the information which team needs to be able to begin work** on it and definition of done is **checklist of all the criteria required to be met so that the deliverable can be considered ready for customer’s use**.

# Double Loop Learning

Double loop learning is a concept that refers to a process of learning that involves not only addressing immediate problems or issues (single loop learning), but also reflecting on and **challenging the underlying assumptions, values, and mental models** that drive behavior and decision-making (double loop learning).

In double loop learning, individuals or teams not only correct errors or problems in their actions (single loop learning), but also question and challenge the underlying assumptions, beliefs, and norms that inform their actions.

This deeper level of reflection and inquiry allows for a more comprehensive and transformative learning process, as it involves questioning and potentially changing the fundamental ways of thinking and operating, rather than just fixing surface-level issues.

Double loop learning is often associated with adaptive learning, where individuals or teams can continuously improve and innovate by challenging and updating their mental models and assumptions. It requires a willingness to question existing practices, challenge assumptions, and be open to new perspectives and feedback.

Double loop learning can lead to more effective and sustainable changes in behavior, as it addresses the root causes of problems and promotes a deeper understanding of the underlying dynamics at play.

# Dynamic Scope Contract Type

In Agile project management, a dynamic scope option contract technique allows for flexibility in the project scope based on the feedback and changing requirements of the customer. This means that the scope of work can be modified throughout the project, without impacting the overall timeline or budget. Under this contract technique, the customer and the Agile project team collaborate to identify the high-priority features and functionality that need to be included in the project, and then work to deliver those features in an iterative manner. As the project progresses, the customer has the ability to adjust the scope of work based on their evolving needs and priorities. The dynamic scope option contract technique is a popular choice in Agile project management, as it aligns with the Agile values of customer collaboration, flexibility, and responsiveness. It enables the customer and the Agile project team to work together to achieve the project goals to best meet the customer's needs.An early cancellation option is not applicable to the scenario description. In Agile project management, an early cancellation option contract technique is a contract that allows the customer to cancel the project early without incurring a significant financial penalty. This type of contract technique provides the customer with flexibility and reduces the financial risk associated with the project.

# **Effectiveness Vs. Efficiency**

Techniques of leadership focus on **effectiveness**; efficiency is a focus of management techniques. The Agile methodology values leadership over management.

# Embellishment

Embellishment is something that teams add on that produces little value. An embellishment in a user story refers to additional details or information that goes beyond the basic requirements or functionality of the story. As a general guideline, embellishments should be limited to information that is essential to understanding the story and should be kept as concise as possible. This helps to ensure that the user story remains focused and actionable.

# Emergent Leadership

A team member who takes the initiative to try a new approach or process once they get approval from the team is an emergent leader. Emergent leadership refers to a form of leadership that arises spontaneously in a group or organization, rather than being appointed or designated by formal authority. It is characterized by individuals who possess the ability to influence and motivate others towards a common goal, even without a formal leadership role.

# Empirical Process

An empirical process is one where learning happens throughout the project, and the path is uncertain.

# Escaped Defects

Escaped defects are defects that make it through all checkpoints to production. Because its remediation will require a large amount of rework, testing, retesting, discovering any dependent code, and then repeating the testing process, escaped defects are the costliest type of defect.

# ESPV

The ESVP was designed to engage and focus on those involved in the retrospective. At the same time, it gained insight into the participants’ attitudes towards the agile retrospective meeting itself.

While it’s still a popular way to kick off a retrospective, ESVPs are now used more widely. They can be icebreakers used to warm up any type of meeting or collaborative session. They can also be used at any point during a session as a temperature check.

ESVP stands for ‘Explorers, Shoppers, Vacationers and Prisoners’. Participants are invited to indicate the persona with whom they most relate at that point in time.

An ESVP focuses on four areas:

1. **Explorer**– Will dive in and discover new things.
2. **Shopper** – Will see what can be procured.
3. **Vacationer**– Will relax and switch off.
4. **Prisoner**– Does not want to be there.

1. Explorer: Team members who take on the role of "Explorer" are enthusiastic and actively seek out new ideas and improvements. They are open to experimentation and are willing to take risks to drive positive change.

2. Shopper: "Shoppers" are team members who are more cautious and observant. They prefer to gather information and learn from the experiences of others before committing to any changes. They are more reserved and may take a wait-and-see approach before actively engaging in improvement initiatives.

3. Vacationer: "Vacationers" are content with the current state of the project or sprint and may not actively participate in improvement discussions. They may be complacent or disinterested in the retrospective process and may require motivation or encouragement to actively engage in discussions.

4. Prisoner: "Prisoners" are team members who may have negative attitudes or feel stuck in their current situation. They may express frustration or resistance to change and may require special attention to understand and address their concerns.

# Epic

An epic is a high-level, strategic initiative that requires more analysis and planning before being broken down into smaller user stories. Epics are used to capture large features or ideas that cannot be completed in a single iteration. The team needs to create a detailed plan for completing the epic, breaking it down into smaller, more manageable user stories. The user stories can then be prioritized and completed in iterations.

# Explicit Knowledge

Explicit knowledge, also known **as expressive knowledge**, is any information that you can document, store, and share with others. You can easily transfer this knowledge to an organization's customers or other employees. This type of knowledge typically serves as a resource for others.

# Exploratory Testing

Exploratory Testing is used to verify functionality in addition to functionally focused testing. Exploratory testing should be used in addition to functional testing. Exploratory testing is a type of software testing approach where the tester actively and dynamically designs and executes tests while exploring the system under test, without relying on pre-written test cases. In an Agile environment, exploratory testing is an effective way to test software in short iterations and constantly changing requirements. It allows the testers to quickly identify and report issues as they arise and work collaboratively with the development team to resolve them in real-time. Exploratory testing is often paired with test automation to provide a well-rounded testing approach that can keep up with the pace of Agile development.

Exploratory testing is a type of testing that typically **requires minimum documentation**. Exploratory testing is a dynamic and flexible approach to testing where testers actively explore the application under test (AUT) while simultaneously designing and executing test cases based on their domain knowledge, experience, and intuition.

Unlike scripted testing, which follows predefined test cases and detailed documentation, exploratory testing relies on the tester's ability to adapt and improvise based on their understanding of the system, its requirements, and the specific areas they want to focus on. This approach allows testers to identify defects, uncover hidden issues, and gain a deeper understanding of the AUT in real-time.

While exploratory testing may involve some documentation, such as capturing defects or logging issues found during testing, it generally requires less formal documentation compared to scripted testing, as the focus is on the tester's skills, creativity, and intuition rather than following predefined test cases or scripts.

This makes exploratory testing well-suited for Agile and iterative development environments, where flexibility, adaptability, and rapid feedback are highly valued, and minimum documentation is preferred to keep the testing process efficient and effective.

Compared to acceptance, integration or scripted automating testing, exploratory testing relies on the tester’s autonomy, skill, and creativity in trying to discover issues and unexpected behavior.

Hence it perhaps needs a least documentation as testers will be using their autonomy, skill, and creativity in trying to discover issues and unexpected behavior.

# Fast Failure

While it can be frustrating and demoralizing, sometimes canceling a project that is not creating value is the best option as those resources can be reallocated to other projects.  Agile preaches failing fast if a project is going to fail. It is much better to cancel a project that will be a failure than to complete it if there is not sufficient value in the project.

# Feature Driven Development

The most distinguishing feature of the FDD approach to software development is having a single owner for each section of code. The FDD approach is focused on having a single owner for each section of code.

# Fishbowl Window

The Fishbowl Window technique involves creating a long-lived video conferencing link to enable remote teams to work together on design and development efforts and exchange information in real-time.

# Five Whys

The Five Whys is a problem-solving technique that originated from the Lean manufacturing philosophy and is often used in agile methodologies as well.

It involves asking "why" multiple times, typically five times or more, to identify the root cause of a problem or issue. The goal is to dig deep into the underlying causes of a problem to address it at its source and prevent recurrence.

In an agile context, the Five Whys can be used as a powerful tool for root cause analysis and continuous improvement. As the name suggests - Five Whys is a technique in which the Participants analyze the underlying cause of a problem by asking “Why?” five times to move beyond their automatic answers and identify the actual root cause of the issue

Agile coaches can use the Five Whys as a coaching tool to facilitate problem-solving and critical thinking among team members. By guiding the team through the process of asking "why" multiple times, coaches can help the team develop a deeper understanding of the problem and arrive at more effective solutions.

# Fixed-Price Increments

Fixed price increments in Agile project management refer to a project pricing model where the project scope is divided into smaller increments with a fixed price assigned to each increment. In this approach, the Agile team works with the customer to identify the scope of the project, and then break it down into smaller increments, each of which has a fixed price. The scope is typically defined in terms of user stories or features, and each increment represents a specific set of functionalities. Fixed price increments provide both the customer and the Agile team with a clear understanding of the project scope and cost and allows for greater flexibility and adaptability in project delivery. As each increment is completed, the customer can choose to continue with the project or stop without incurring any further costs. This approach helps ensure the project stays on track and delivers value to the customer at each stage. Emphasize value delivered, graduated time and material, and dynamic scope option are also contracting techniques but do not relate to the given description.

# Feeding Buffers

It is a buffer that is inserted before the first activity on the Critical Chain. This is to ensure that any tasks feeding into the Critical Chain may not delay the Critical Chain. When a delay does occur in the feeding activities, the feeding buffer is consumed so that Critical Chain is not affected.

# Flow Master/Manager (Kanban)

Kanban is one of the popular Lean workflow management methods for defining, managing, and improving services that deliver knowledge work. The word Kanban is a Japanese word, meaning “visual board” or a “sign”. It helps the project team to visualize their work, maximize efficiency by keeping the work in progress or WIP under control.

In scrum, we have well-defined roles like product owner, scrum master and scrum team. The roles in Kanban however will not be the same as Scrum.

Kanban is a flow-based approach and many teams who are new to Kanban need some hand-holding, and a bit of practical advice, especially if they don’t have access to an agile coach.

Getting high-quality training to learn the Kanban is essential, but not always possible and may require an additional cost.

In these situations, the project team may need someone to step them up and help the team get to a more mature level. This person or the role is the Flow Manager.

This is a scrum master equivalent role in the context of continuous or a flow based approach such as Kanban.

# Generalizing Specialist (T-shaped professional)

Required to master several areas within the team’s delivery work but are not expected to master all. This allows individual team members to explore additional areas of work, while also smoothing the overall workload as more individuals can tackle any given challenge.

A generalizing specialist, also known as a T-shaped professional, refers to an individual who has a broad skill set and expertise in multiple areas, while also possessing deep knowledge or specialization in one or more specific areas. This person can contribute across different disciplines and is not limited to a single role or skill set. A generalizing specialist is typically adaptable, collaborative, and able to work on a variety of tasks or projects as needed.

When team size is reduced, the Agile team leader can encourage the team members to become generalizing specialists in order to reduce team size and bottlenecks, and to create a high performing cross-functional team.

# Gulf of Evaluation

 The Gulf of Evaluation, in the context of Agile project management, refers to the cognitive effort or gap that users experience when they need to interpret and make sense of the displayed information or interface elements. It represents the difficulty users face in understanding and evaluating the system's state, feedback, and actions.

# Hardening Iteration or Iteration H

In Agile methodologies, such as Scrum, the concept of "hardening iteration" refers to a dedicated time-boxed iteration or sprint focused on stabilizing, improving, and preparing the product for release. This iteration typically occurs after the main development iterations and before the product is released to production.

The purpose of a hardening iteration is to address any remaining defects or issues, perform final testing and quality assurance, optimize performance, enhance security, and ensure the product is ready for release. It provides a buffer period for addressing technical debt, resolving any outstanding bugs, and fine-tuning the product before it is deployed to production.

During the hardening iteration, the development team may work closely with the quality assurance (QA) team, operations team, and other stakeholders to address any identified issues, conduct additional testing, and make necessary improvements. The focus is on achieving a stable and high-quality product that is ready for deployment to the end-users or customers.

It's worth noting that not all Agile teams use a hardening iteration, as some teams prefer to address issues and perform final testing within the regular development iterations. However, for teams that do use a hardening iteration, it serves as a dedicated time-boxed period for addressing any remaining technical issues and ensuring the product is ready for release to production.

# Health Check Questionnaire

Health check questionnaire can be used to find issues in a process and determine how well a team is adhering to Agile methods. The Team Health Check is a specialized self- assessment tool for agile project teams who are looking to improve the way they work together to meet project outcomes. Changes are tracked over time so that the team can see progression or identify new areas for improvement.

# Heuristics

Heuristics are cognitive shortcuts or mental rules of thumb that humans use to simplify decision-making and problem-solving in complex situations - not something team is trying to do in the scenario.

# I-Shaped professional

An I-shaped specialist refers to an individual who has deep expertise in one specific area or domain but may have limited or no skills in other areas. This person may excel in a particular domain but may be less flexible or adaptable in other areas outside their specialization.

# Incremental Design

Incremental design is to split the design problem into small problems and work on these problems one by one while making improvements in the solutions by getting feedback from the customer.

# Intrinsic quality

Intrinsic quality is required to deliver continuous value to the customer, it’s an internal quality of the product which is not visible to the end user but is needed to make the product adaptable for future needs. It consists of all built-in qualities possessed by the product. That includes suitability, durability, or reliability, along with uniformity and maintainability. We can measure such a type of quality quantitatively thanks to test coverage, escaped defects, bugs per line of code, and so on. So **“Intrinsic quality is needed to build a releasable product, in the absence of intrinsic quality users will not be able to get the intended value from the product”.**

# **Iteration 0**

Iteration 0, also known as the "Inception" or "Initiation" phase, is the initial phase of a project or product development cycle in Agile methodologies. It is a preparatory phase where the team conducts initial planning, sets up the development environment, establishes team roles and responsibilities, defines the project vision and goals, identifies stakeholders, and conducts initial research and analysis.

# **Iterations**

In Agile project management, iterations are typically used to review the work completed during a Sprint and make changes to the project plan based on feedback from stakeholders. This allows the team to adapt to changes quickly and ensure that the final product meets the needs of the stakeholders. Iterations are a part of a Sprint, not a replacement for Sprints. Sprints and iterations are not the same thing, and iterations are not used for monthly progress reviews.

# **Iteration Burn-Down Chart**

The Iteration Burndown Chart is a graphical representation and projection of the remaining unfinished work in that Iteration.

# Kaizen

The best example of the Kaizen principle's small incremental improvements method is shown in the team's retrospective meetings; retrospective meetings are focused on improvement.

Kaizen focuses on the below aspects,

1. Team - the people who are doing the work & not just the leadership
2. Frequent changes, against infrequent changes
3. Small & incremental changes against large changes,

Kaizen is all about making small, incremental improvements - often to reduce or better yet eliminate waste or to eliminate overly hard work or to improve on the existing state of the system.

It is an ongoing effort to improve processes, tools, and techniques in small steps over time. Kaizen involves everyone in the team, from the individual team members to the management, and focuses on making small changes that lead to significant improvements.

Using the kaizen approach, the team works together to create a plan for implementing desired change during the next sprint. They identify specific tasks, assign responsibilities, and set timelines for completing the work. Throughout the sprint, the team continuously monitors their progress and adjusts as needed.

At the end of the sprint, the team reviews the results of their kaizen approach and the improvements they made to their process. They analyze the impact of their changes and make further adjustments as necessary, continuing the cycle of continuous improvement.

# Kanban

Kanban methodology is a popular Agile framework that emphasizes visualizing work, optimizing workflow, and limiting work-in-progress (WIP) to improve efficiency and delivery times. One of the key features of Kanban is the concept of WIP limits. WIP limits help teams focus on completing work already in progress before starting new work, thereby reducing the amount of multitasking, and improving efficiency. Teams using Kanban typically set explicit limits on the number of items that can be in each stage of the workflow at any given time, based on their capacity and workload. Another feature of Kanban is the concept of "pull" rather than "push." Work is pulled through the workflow as capacity becomes available, rather than being pushed through by the team. In other words, a team member will only pull an item into their workflow stage when they can work on it, rather than being assigned a new task when they are already at capacity. This helps to ensure that work is not overloaded on any one team member or stage of the workflow, and that items move through the workflow at a steady pace.

## Kanban Benefits

### Flexibility

Kanban is extremely flexible. It starts with what you do now and helps you optimize workflow over time. There is no rigid set of rules to follow. Teams can modify Kanban boards and processes to suit their needs allowing for evolutionary change rather than an abrupt overhaul of how work gets done.

### Visibility

Kanban also provides excellent visibility into the current state of work and potential issues. The visual boards make it easy to see where tasks are in the process and spot any constraints or bottlenecks. This real-time information allows teams to make on-the-fly adjustments to improve flow and deliveries.

### Continuous Improvement

Finally, Kanban encourages continuous improvement through monitoring of [flow and cycle times](https://deeprojectmanager.com/kanban-metrics/). Teams. meet regularly in front of the Kanban board to discuss what’s working, what’s not, and make tweaks to improve. This collaboration leads to faster and more efficient delivery over time.

## Kanban Order of Practice

The order of the practices in Kanban: **visualize workflow, limit WIP, manage flow, make process policies explicit, implement feedback loops, improve collaboratively, and evolve experimentally**.

## Six Practices of Kanban

The five core practices of Kanban are Define and Visualize the workflow, Limit WIP, Measure and Manage Flow, Make Process Policies Explicit, and Use Models to Suggest Improvement.

1. Visualize workflow
2. Limit work in progress
3. Manage flow
4. Make process policies explicit
5. Implement feedback loops
6. Continuously improve.

## Kanban Roles

### The Service Delivery Manager (SDM)

The Service Delivery Manager oversees the overall process and ensures that the team is functioning effectively. They’re responsible for:

* Implementing and maintaining the Kanban system
* Ensuring that work in progress is limited and that the team is focused on the most important tasks.
* Coaching the team in Kanban principles and practices.
* Facilitating communication between team members and stakeholders.

### The Service Request Manager (SRM)

The Service Request Manager plays an integral role in the Kanban system and aims to optimize the flow and improve the overall throughput of the system. They’re responsible for:

* Analyzing incoming service requests from customers and determining the priority and complexity of the requests
* Reviewing the requests, clarifying any vague areas with the customers, and then deciding which Kanban board the request cards should be placed.
* Working with the teams to make sure work in progress limits are not exceeded.
* Monitoring the flow of requests through the Kanban boards and ensuring service level agreements are met.
* Working to smooth workflow and removing any blockers.

### The Kanban Team

The Kanban team is a cross-functional group of individuals who collaborate to complete work items. They’re responsible for:

* Selecting and pulling work from the backlog
* Completing tasks in a timely manner
* Communicating with other team members to ensure smooth workflow
* Actively participating in continuous improvement efforts

### The Product Owner

The [Product Owner](https://deeprojectmanager.com/business-analyst-vs-product-owner/) is responsible for defining the work items and prioritizing them in the backlog. Their main responsibilities include:

* [Collaborating with stakeholders](https://deeprojectmanager.com/manage-stakeholder-engagement-effectively/) to gather requirements
* Creating and refining work items, such as user stories or tasks
* [Prioritizing the backlog](https://deeprojectmanager.com/backlog-grooming-vs-sprint-planning/) based on business value, dependencies, and risk
* Ensuring that the team has a clear understanding of the work items and their respective priorities

### The Stakeholders

Stakeholders are individuals who have an interest in the project outcome. They include clients, users, management, and other teams. Stakeholders are responsible for:

* Providing input and feedback on work items
* Participating in reviews and demonstrations
* Supporting the team by [providing resources](https://deeprojectmanager.com/resource-breakdown-structure/), information, and decisions as needed

# **Kano model analysis**

The Kano model is a prioritization model. It helps you prioritize features or scope during the Definition Stage of your project, or for creating a Product Roadmap in an Agile project. It differs from MoSCoW as it prioritizes features according to how much they are likely to satisfy and delight customers. Weigh this against implementation costs.

1. Dissatisfiers – basic attributes the customer expects to be present. These “must-be” attributes or hygiene factors are taken for granted and will trigger dissatisfaction If unfulfilled.
2. Satisfiers – attributes that generate satisfaction when fulfilled and dissatisfaction when unfulfilled. Improvement in performance on these attributes produces higher level of satisfaction.
3. Delighters – exciting features that customers do not expect but are delighted when the find them.
4. Indifferent – attributes which offer requirements that customers do not care about or are no longer interested in.
5. Reversed – attributes where the reduction in functionality increase satisfaction. For example, several customers feel overwhelmed by the extra features in some devices. When extra features are removed, it simplifies the device and satisfaction improves.

# Key Performance Indicators (KPI)

KPIs (key performance indicators) are the most reliable metrics to use. KPIs give you the progress rate, the remaining work to be done, the likely date of completion, and the likely costs remaining.

# Levels in the Agile Adoption Framework

If your team is at level 1 in the Agile adoption framework, the team should focus on communication and collaboration (C). The other options require a higher level of maturity.

* Level 5: Encompassing Establishing a vibrant environment to sustain agility
* Level 4: Adaptive Responding to change through multiple levels of feedback
* Level 3: Effective Developing quality, working software in an efficient an effective manner
* Level 2: Evolutionary Delivering software early and continuously
* Level 1: Collaborative Enhancing communication and collaboration

# Large Scale Scrum or Less

Remember, the PMI Agile practice guide includes the description of agile frameworks such as LeSS - which can provide a framework for scaling scrum to multiple teams who work together on a single product.

As mentioned in the above scenario, the company is planning to utilize their global talent pool which is spread across multiple countries, time zone etc.  and yet want these multiple teams to work on a common project/product.

Large Scale scrum or LeSS is the correct choice in above scenario because this agile framework/approach provides the necessary practices required for systematic coordination between the multiple development teams.

With LeSS, there are two frameworks to support diverse needs of scaling of a project, as follows,

* LeSS - is smaller scale framework supporting up to 8 teams
* LeSS Huge - as the name suggests, it can support up to 1000 team one product

# Last Responsible Moment

Always Remember the principle of the last responsible moment; in agile we do things at the last responsible moment. The stories should have details and acceptance criteria attached before they get developed not before they get estimated.

# **Lean Portfolio**

A set of strategic business targets captured in initiatives. It is a collection of Value Streams for a specific business domain in an Enterprise.

# Lessons Learned

In an iteration based agile delivery model team meets, After the end of the iteration in a meeting called retrospective and in this meet the entire team along with all the stakeholders can discuss the new learnings from the concluded iteration.

Doing the lessons learned at the end of the project is not very useful because team will not be able to apply the learning into the current project. Also doing the lessons learned at the end of certain project milestones is not recommended because these project milestones can be wide apart, and team will miss the chance of of learning in between them. Every iteration will be an opportunity to learn and apply the new learnings so doing the the lessons learnt at the end of first few iterations is not desirable.

# Low-tech, high-touch tools

Agile teams prefer to use low-tech, high-touch tools. These tools are simple, such as cards and charts, and therefore easy for all team stakeholders to manipulate by moving the cards, reordering the lists etc.

* Low-tech tools are often simpler and more straightforward, which can make them easier to understand and use. They may have fewer features and settings, reducing complexity and potential distractions, and allowing the team to focus on the essentials of project planning and tracking. This simplicity can also help the team avoid unnecessary complications or delays in their agile project management process.
* promote communication and collaboration. High-touch tools, which rely on human interaction, can foster better collaboration and communication among team members. They may encourage face-to-face interactions, discussions, and brainstorming sessions, which can lead to more effective problem-solving, decision-making, and alignment among team members. This can be particularly beneficial in construction projects where effective communication and collaboration among team members are critical for project success.
* One of the other reasons to use low-tech, high-touch tools is they also help avoid a tool-related perception of data accuracy.

# Mad Sad Glad Retrospective

The mad sad glad retrospective frames discussion around the emotional journey of by your team during the previous sprint, and is a great way to identify opportunities to improve team morale and job satisfaction. The retrospective asks participants to share what made them feel frustrated or annoyed (mad), disappointed (sad) and what made them feel happy or proud (glad).

This mad sad glad style retrospective encourages your team to be more emotionally-aware to help build a positive team dynamic, and provides an opportunity to reflect on issues and opportunities from a different perspective. It is important you remember the [retrospective prime directive](https://www.teamretro.com/retrospective-prime-directive/) and focus on events, behaviours and processes – not on assigning blame or ‘guilting’ individuals.

* Mad - List the things that are driving you crazy. What is stopping you from performing at your best?
* Sad - What are some of the things that have disappointed you or that you wished could be improved?
* Glad - What makes you happy when you think about this project? What are the elements that you enjoy the most?

# Maximizing work not Done

"The art of maximizing the work not done" is a phrase from the Agile Manifesto, which emphasizes the value of simplicity and efficiency in product development.

It means that while it is important to deliver value to the customer, it is equally important to avoid unnecessary work and complexity. This can be achieved by prioritizing work based on its value to the customer, eliminating work that does not add value, and simplifying processes and systems wherever possible.

By focusing on maximizing the work not done, teams can increase their efficiency, reduce waste, and deliver value to customers more quickly and effectively. In the above scenario, based on the complexity of the upcoming product, the team should focus on "simplicity" - the art of maximizing the work not done.

The team should prioritize work based on its value to the customer and avoid unnecessary work and complexity wherever possible. This approach will help the team to reduce waste, increase efficiency, and deliver value to the customer more quickly and effectively. The team can also consider creating a detailed architecture, hiring subject matter experts, and hiring highly motivated individuals with technical excellence to support the development process, but the primary focus should be on simplicity and avoiding unnecessary work.

# Mobbing

The team can use mobbing as a collective problem-solving technique to help the two programmers.

In mobbing, the whole team comes together to work on a single problem or task, and they collaborate and share their knowledge and ideas to find a solution. This can be helpful in situations where a problem is complex or difficult to solve, and the team needs to leverage the collective intelligence and expertise of all team members to find a solution.

By working together on a problem, the team can often reach a solution more quickly and effectively than if each person worked individually.

With multiple sets of eyes looking at the same code, it's more likely that defects will be caught and corrected before they become a problem.

# MSA – Master Service Agreement

The situation mentioned in the above scenario can be quite common when the project has requirements that require both predictive style project management and adaptive style project management. It is quite natural in this case to utilize the master service agreement, or MSA, for the overall engagement between the two parties involved in the work.

For adaptive work where the changes are supposed to happen more frequently, they can be placed in an appendix or a supplement. This allows changes to occur on the adaptive scope without impacting the other elements of contract engagement.

When there are elements of both predictive and adaptive work, the team can develop a master services agreement, or MSA, for overall engagement and place adaptive work in an appendix or supplement. This allows changes to occur on the adaptive scope without impacting the overall contract.

# Multitasking

Multitasking can be bad for Agile teams for several reasons:

1. Decreased productivity: Constantly switching between tasks can lead to a decrease in productivity as it takes time to refocus and get back into the flow of a task.
2. Reduced quality: Multitasking can lead to mistakes and oversights, which can affect the quality of the work being done.
3. Increased stress: Trying to juggle multiple tasks and deadlines can lead to increased stress and burnout among team members.
4. Inefficient use of resources: Multitasking can lead to inefficient use of resources, such as time and energy, as work is spread across multiple tasks.

In Agile, the focus is on completing tasks one at a time and delivering high-quality work in each iteration. Multitasking can hinder this process by creating distractions and making it harder for the team to prioritize tasks and manage their time effectively.

Multitasking slows down the throughput of the whole team, as teams waste time switching tasks and hence multitasking is considered undesirable.

# MVP

MVP (minimum variable product) refers to the functionality that is complete enough to be useful to the users or the market, yet still small enough that it does not represent the entire project.

The primary benefit of an MVP approach is as a product team, you can gain understanding about your customers’ interest in the product without fully developing the product with all its features, which obviously takes long duration and is more costly.

The sooner the project team can find out whether the product will appeal to customers or not, the less effort and expense will be spent on a product that will not succeed in the market. The final, complete set of features in a product can only be designed and developed after considering feedback from the product's initial users.

The 3 key points about a MVP that still needs to be kept in mind are below,

* The MVP should have enough value that people are willing to use it or buy it initially.
* The MVP demonstrates enough future benefit to retain early adopters.
* The MVP provides a feedback loop to guide future development.

You should Plan for small releasable increments by organizing requirements into minimally marketable features/minimally viable products to allow for the early recognition and delivery of value.

# Osmotic Communication

Osmotic communication means that information flows into the background hearing of members of the team, so that they pick up relevant information by osmosis. This is normally accomplished by seating the project team members in the same room.

# Pair Programming

XP utilizes pair programming: when one developer codes and the second developer checks the code is the correct answer. This does require more resources as two individuals are working on the code at the same time; however, it does help improve code quality and team knowledge and two individuals are supporting each other.

This approach finds defects earlies in the process resulting in the greatest cost savings.

Pair programming should be done throughout the day and allows the team to notice when something is not right. This allows multiple individuals to review each other's work and suggest improvements.

# Participatory Decision Making

Participatory decision-making means engaging the project stakeholders and the team members in the decision-making process.

# Planning Poker

Planning poker is useful to remove or minimize bias.

Planning Poker involves team members estimating work items or user stories using a deck of cards, with each card representing a relative size or effort level. Estimation is done anonymously, meaning team members do not know who estimated what. This helps minimize bias that can arise from social or hierarchical influences, as team members are free to provide their independent estimates without being influenced by others.

After each team member provides their estimate using the Planning Poker cards, the team discusses the estimates and any discrepancies. This open discussion allows team members to share their perspectives, reasoning, and insights, which can help identify and minimize biases that may have influenced the estimates. The team then works towards achieving consensus on the estimated effort or size of the work item, which encourages collective decision-making and minimizes individual biases.

During planning poker, the facilitator will count to three, and then everyone will lay down their cards at the same time, showing their estimates of the effort required to develop that user story. They All Turn over then cards simultaneously to ensure that the initial estimates are not influenced or biased by other members of the group.

# Pre-Mortem

A pre-mortem focuses on identifying possible causes of failure and adjustments to the plan before a project even begins. Project premortem is a technique used to identify and mitigate potential project risks before they occur, the key is to conduct the premortem early in the project planning phase so that potential risks can be identified and addressed before they become actual problems.

While project premortem can be applied to any type of project, it is particularly suited for long-term projects where there is a higher degree of uncertainty and risk. These types of projects typically involve multiple stages, and it can be difficult to anticipate all the potential problems and risks that may arise throughout the course of the project.

By conducting a premortem, the team can identify and address these risks early on, which can help prevent problems and ensure a successful outcome. Pre-mortems are especially valuable on long-running projects that are likely to experience more change than short projects, simply because they are exposed to a longer horizon of risk.

# Predictive Project Management Approach

A predictive project with clear procedures and defined work processes. The fact that the company uses new technology does not state that they do not know how to assemble the product, especially if it's an existing model (A113). Definable work means the project manager can use the traditional predictive approach.

# Project Charter

A project charter is a formal document that establishes the authority and responsibilities of a project, and provides a high-level overview of the project's purpose, objectives, scope, stakeholders, and key deliverables. It serves as a reference document that outlines the project's overall direction and provides guidance to the team and stakeholders throughout the project lifecycle.

A project charter is typically created at the initiation phase of a project, and it serves as a guiding document that helps ensure a common understanding of the project's goals, scope, and expectations among team members and stakeholders.

It provides a foundation for effective project planning and execution and serves as a reference document that can be used throughout the project to ensure alignment and adherence to the project's objectives and scope.

# Project Tweet

This exercise requires stakeholders to describe the goal of the project in 140 characters or less. The intent of this activity is **not to create** an all-encompassing description, but to gauge stakeholders’ high-level understanding of the project and their priorities.

# Product Roadmap

In a typical Agile project, the product owner is responsible for producing the product roadmap. The product roadmap is a high-level visual representation of the product vision and the sequence of deliverables that will be created to achieve that vision. It outlines the major features and functionalities that will be developed over time, and the estimated timeline for their delivery. The product roadmap is an essential tool for communicating the product vision to the development team, stakeholders, and customers. It provides a clear understanding of the project's priorities and direction and helps to ensure that everyone is working towards a common goal.

# Primary Goal of Lean

The primary goal of Lean is to optimize the whole with speed and sustainability. This can be summarized as “fast-flexible-flow,” which is the fundamental phrase used in Womack & Jones. This emphasizes the importance of optimizing the entire value stream, from start to finish, to achieve the fastest possible delivery of high-quality products while also ensuring sustainability over the long term.

# Project Approaches

## Agile Approach

Projects that have high requirement complexity and high technical complexity often require a flexible and adaptive approach that allows for frequent iterations and adjustments. Agile methodologies, such as Scrum and Kanban, are designed to handle such projects and provide a framework for managing complex requirements and technical challenges.

Although Agile projects have complex technology and requirements, they do not have technology or requirements that are near chaotic. Agile projects are known for their flexibility and adaptability, which makes them suitable for handling complex and uncertain projects.

## Predictive Approach

Projects with low technical complexity and low requirement complexity may not require the same level of flexibility and adaptability as more complex projects. Similarly, highly uncertain technology and highly uncertain requirements may require a more predictive approach that emphasizes planning and risk management.

# Prune the Product Tree

The Product Tree is a dynamic framework that embodies strategic product planning through a biological metaphor.  It allows [product managers to take their work to the next level](https://fibery.io/blog/how-to-become-a-great-product-manager/) by allowing them to organize features, tasks, and milestones into one cohesive, growth-oriented roadmap. The Product Tree is made of several parts:

* **The trunk: your product’s backbone.** Imagine the trunk of your product tree as the sturdy core features that are currently active in your product. These features form the backbone of your product, they are non-negotiable and intrinsic to the product’s identity.
* **The branches: visualizing priority.** The branches are a representation of your product’s feature branches. Their thickness can vary to symbolize the level of importance – the bigger the branch, the higher the priority.
* **The leaves: feature forecasting.** The leaves are the key to the product tree. They represent the individual features or tasks that, when combined, compose the entirety of your product’s offerings. Each leaf is an idea, an enhancement, or a proposed change that could potentially be implemented into the product.
* **The roots: the supporting technology.** Lastly, the roots depict the technological groundwork and infrastructure that underpin your product. As with any real tree, as the canopy expands, so too does the need for a robust root system.

In the context of your product, this means that you must continuously consider and bolster the technological framework to support your growing feature set.

# Ranking Techniques

## **Weighted prioritization technique**

Weighted scoring prioritization uses numerical scoring to rank your strategic initiatives against benefit and cost categories. It is helpful for product teams looking for objective prioritization techniques that factor in multiple layers of data.

## **Simple ranking technique**

It is the simplest method of job evaluation that involves ranking each job relative to all other jobs, usually based on overall difficulty. This method is considered to be a simple form of job evaluation compared to other methods. Here the output doesn’t have categories like must-have, should-have, could-have, and won’t-have.

As the Development Team works, it keeps the Sprint Goal in mind. In order to satisfy the Iteration Goal, it implements functionality and technology. If the work turns out to be different from what the Development Team expected, they collaborate with the Product Owner to negotiate the scope of the Sprint Backlog within the Sprint, after discussion removes, or changes the selected backlog items, and starts the sprint by monitoring progress. So **“Discard the code from the iteration review and put the story back in the backlog for future prioritization and implementation in a successive iteration”.**

## Systems Thinking Approach

In summary, the most successful agile projects in the systems thinking approach are those that have complex technology and complex requirements, as they require a flexible and adaptive approach to manage the inherent complexities of the project.The systems thinking approach involves understanding the entire system and its interconnections, rather than just focusing on individual components. This approach requires a deep understanding of the technical and requirement complexities of the project.

# Red Zone

In this scenario, Raquel’s team is working in the Red Zone, which goes against Agile principles. Working in the Red Zone refers to tense, defensive environments where individuals do not support each other and constantly bicker. This is contrasted to working in the Green Zone, which is a collaborative, supportive environment.

# Red, Green, and Refactor

Red, Green, and Refactor are the three phases of Test-Driven Development and this is the sequence that gets followed while writing code. When followed, this order of steps helps ensure that you have tests for the code you are writing, and you are writing only the code that you must test for.

 It’s defined as the process of writing a test that initially fails (“Red”), adding code until the test passes (“Green”), then refactoring the code (“Clean”) so that it is streamlined as possible.

# Regulatory Compliance – Just Because

**Regulatory compliance is one instance where “just because” comes into play** when it comes to an agile approach to documentation.

Since regulatory compliance mandates the development of a safety compliance document for the safety camera, the team would need to create documentation that fulfills this requirement. However, in keeping with agile principles, the team would aim to create the minimum amount of documentation necessary to meet the regulatory compliance requirements, without adding unnecessary documentation that does not provide value or hinder the team's ability to deliver working software.

The team would likely create documentation that is relevant, concise, and targeted specifically to meeting the regulatory compliance requirements, without going overboard with excessive documentation that may not be necessary for the development process or the needs of the stakeholders. This approach aligns with agile principles of valuing working software over comprehensive documentation and emphasizes creating just enough documentation to meet the immediate needs of the project.

As per the industry standard set by the regulatory body sometimes it is best to develop the necessary documentation to satisfy regulatory compliance.

# Release burndown

Release burndown chart In general, a burndown chart trends downward until it reaches zero. In practice, some events can reverse the downward trend of your burndown chart. For example, development work frequently uncovers a greater scope for a user story than was initially estimated, or user stories are added to the backlog but should not go below the x-axis.

Burn-down Chart shows work remaining in the sprint or release or Project. A Burn-down chart is a simple graph used to track a team's progress and help estimate how much time is required to complete the Project. The tasks remaining in the Project are usually plotted on the y-axis, while time is plotted on the x-axis. **So this is abnormal and indicates poor data quality being plotted on the burndown chart. The bar graph can touch the X-axis, but not go below it.**

# **Remember the Future**

This technique encourages teams to visualize what success looks like and think through how they can get there.

One such game that the team can use to set vision and requirements-elicitation is a game called **Remember the future** - This is a facilitated workshop exercise that engages project stakeholders in imagining the release or iteration is now complete and then gets them to describe what has occurred for it to be successful.

# Retrospective Stages

Retrospectives tend to occur after each iteration; however, nothing prevents a team from choosing to only have them every release. Different stages of the retrospective are Setting the Stage, Gather Data, Generating Insights, Decide What to Do, and Wrapping up.

* **Set the stage**- The setting should be such that regardless of what we discover, we understand that everyone did the best job he or she could.
* **Gather data**- This is often done by looking back and identifying what went well and what did not.
* **Generate insights**- In this phase, teams typically identify why things happened and what should be done more, done less, and tried out.
* **Decide what to do**- This includes deciding on specific, meaningful, agreed, and realistic actions that will be done in the next Sprint.
* **Wrap up the retrospective -**Wrap up the retrospective.

# Risk Based Spike

Risk-based spike is not a risk identification technique in Agile development, but rather a risk mitigation technique. A risk-based spike is a time-boxed effort aimed at addressing and mitigating specific risks or uncertainties that may arise during a project. It involves a focused exploration or investigation to gather information, conduct experiments, or prototype solutions to reduce the identified risks.

Risk-based spike is **not a risk identification technique** but is often used on software projects to test unfamiliar or new technologies early in the project before we proceed too far with development.

* It allows the team to gather necessary information, validate assumptions, and make informed decisions to manage risks effectively and ensure the successful completion of the project.
* It involves conducting research, analysis, or experimentation to gather information, validate assumptions, or explore potential solutions related to identified risks. The goal of a risk-based spike is to reduce uncertainty, increase knowledge, and make informed decisions to minimize risks and improve project outcomes.

# Risk Identification Techniques

* Risk Workshop
* Sailboat
* Brainstorming

# **Risk Mitigation Strategies**

Risk response Avoid – Change plans to circumvent the problem. So here terminating the project is to avoid the risk.

* **Avoid—**seeking to eliminate uncertainty
* **Transfer—**passing ownership and/or liability to a third party
* **Mitigate—**reducing the probability and/or severity of the risk below a threshold of acceptability
* **Accept—**recognizing residual risks and devising responses to control and monitor them

# Risk Reward Contract Relationship

Focus on pursuing a shared risk-reward relationship would be a better approach in this scenario as it would align the interests of both parties towards the successful delivery of the project.

This can help to establish a collaborative relationship between the organization and the vendor, which can lead to better communication, coordination, and ultimately, better project outcomes.

It is important to have a clear understanding of the risks and rewards involved and to negotiate terms that are fair and equitable for both parties.

# Risk Register

When teams have identified threats and issues, they should maintain a prioritized list that they keep visible and constantly monitor. The reason for this is to encourage the team to act on the issues (rather than ignore them), and to make sure that each issue has an owner and that the team keeps track of the status of each issue. So below are the valid steps to be performed after identification of threats and issues,

1. Assign a relative priority to each of the issues, risks, and problems.
2. Assign owners to each of the problems and risks and keep track of the status.
3. Encourage action on specific issues that were raised.

# Rolling Wave Planning

Rolling wave planning is **iterative, and “detailed” planning is only done for near term work** as more information is available for such work.

Rolling wave planning is the most suitable planning approach for a project with unclear requirements, where the details of the project become clearer as the project progresses. This approach allows the team to plan and execute work in smaller, more manageable phases or waves, where more detailed planning is done for the immediate future and less detailed planning is done for the farther future.

This approach helps to balance the delivery pressure while not compromising on the technical details and adapting to the unknown and changing requirements.

Even though the requirements are unclear, it is still important to have some level of planning to ensure that the team is working towards a common goal. Lack of planning can lead to chaos, rework, and missed deadlines.

# Sailboat Retrospective

The sailboat retrospective is a retrospective technique where you and your agile team members will envision the last sprint as a sailboat. It’s a visual way for your team to identify what pushed the project forward, as well as what held it back.

Sailboat is not a requirements prioritization scheme.

The Sailboat retrospective is a visual facilitation technique used in agile and Scrum teams for reflecting on a project or sprint and identifying areas of improvement. It is often used during the retrospective meeting, which is a dedicated time for the team to reflect on their performance, discuss what went well, what didn't, and identify ways to improve.

The Sailboat technique can also be used to identify project risks in addition to its use in retrospectives.

The team can collaboratively discuss and identify potential risks, their severity, and possible mitigations or actions during a risk identification workshop using the Sailboat technique. It provides a visual representation that helps the team to identify and prioritize risks, as well as plan actions to mitigate them and increase the chances of project success.

# Speed boat (Collective Workshop)

The exercise takes the metaphor of a boat and gets players to think about what will cause them problems, or else help them move forward with a project. The speed boat is much more than just a game; it will enable **collective intelligence** to be brought into play thanks to the following visuals:

* **The boat**: this is the central element. It symbolizes the team, regardless of the subject of the discussion (create a new product, achieve a successful transformation project, design another organization, to name just a few).
* **The island**: this represents the objectives to be achieved (market release of an innovative product, establishment of a new organization, improvement in a process, and more).
* **The wind**: this shows the strengths of the team. Just as the wind blows into the sails of the boat, the assets identified by the group will enable it to move forward.
* **The anchors**: these represent the brakes, the elements which will slow the team down.
* **The reef**: this element isn't used in all speed boat workshops. When it is used, the reef represents the obstacles that might appear on the boat's (meaning the team) path. Once established, all of these elements enable four main themes to emerge, recalling the **SWOT matrix**: Strengths, Weaknesses, Opportunities and Threats.

The speed boat therefore constitutes a vital [**collaborative working tool**](https://klaxoon.com/visual-platform) at any point in the life of a project. Simple to carry out, participatory and adaptable, it is an agile working method which enables the whole team to move forward. As the current carries them and as time moves on, against winds and tides!

# Servant Leadership

A leadership philosophy emphasizes the importance of prioritizing the needs of team members and empowering them to make decisions. In an Agile project management context, servant leadership can help to promote communication, collaboration, and teamwork. It is not a top-down management approach that emphasizes control and direction.

# Story Map

Map out the major product features and the user stories within them ordered by value.

# Scrum Master

Subhash is working as a scrum master and in this role, there are few responsibilities that as a scrum master should be performing in a scrum team like,

* removing the impediments or obstacles, if any, faced by the scrum team.
* help facilitate any meetings with external teams or other members whose help is needed in the project.
* educate the team on scrum and make sure the scrum principles and practices are honored and followed by the team.
* help the team understand product vision.

The team velocity usually varies the most in the first few iterations and then begins to stabilize. While it may seem logical to predict ever-increasing velocity as the team gains experience, velocity does typically plateau. One Reason for this is that, as the product gets bigger, there is more to maintain, refactor, and possibly support if early versions of the product have been deployed.

So beyond a certain point, the team velocity does not always increase rather it becomes stable. So as a scrum master, Subhash should not worry about a stable team velocity, and he should not try to push the team to increase the team velocity. In general, knowledge work projects tend to increase in complexity as the work is being done.

# Scrum Team Members

Product Owner, Scrum Master, Development Team

# ScrumBan

A Hybrid Agile methodology combining Scrum and Kanban. ScrumBan is often used by teams that want to improve their workflow, minimize waste, and increase productivity. It allows teams to use Scrum's framework for planning, review, and retrospective, while using Kanban's pull-based approach for managing the flow of work.

# Shu-Ha-Ri

Shu can either mean "to protect" or "to obey.” Ha is another term with an appropriate double meaning: Ri, too, has a dual meaning, the second part of which is "to set free”. So Following the rule, breaking the rule, and being the rule respectively is the correct order.

* In the Shu stage, the learner follows the rules and learns the basic techniques and principles of the methodology or practice.
* In the Ha stage, the learner starts to break the rules and experiment with the techniques and principles, adapting them to their own needs and style.
* In the Ri stage, the learner has internalized the principles and techniques to such an extent that they can go beyond the rules and create their own style and approach.

# Sashimi

Similar to the way every other slice tastes. Scrum uses the sashimi technique to require that every slice of functionality created by the developers be complete.

# SMART Goals

One Agile method used to turn action items into goals is to use the SMART method: **specific, measurable, attainable, relevant, and timely**. After the goals are clearly understood, each issue is discussed and turned into a SMART goal. If necessary, adjustments are made, and the fix is executed where appropriate.

# Spike

Spike is usually a timeboxed effort, used by the agile team and the purpose of the spike is to explore an approach, investigate an issue, or reduce a project risk. Teams can run “spike” at any time but mostly spikes are carried out at the beginning of the project before the development effort begins.

If the spike’s outcome is a failure, team can either try another approach or discontinue the work since it is no more feasible and thus saves wasting valuable time & resources and thus can help project avoid or minimize the sunk cost.

# Sprint Planning – Acceptance Criteria

In Agile Scrum methodology, the acceptance criteria for user stories are typically developed during the Sprint Planning event or ceremony.

Sprint Planning is a collaborative meeting that occurs at the beginning of each sprint, where the Scrum team comes together to plan the work to be done in the upcoming sprint. During this event, the Product Owner and the Development Team work together to discuss and clarify the user stories that are selected for the sprint backlog.

As part of Sprint Planning, the Product Owner presents the user stories to the Development Team, and the team discusses and asks questions to gain a shared understanding of the requirements.

The acceptance criteria, which define the conditions that must be met for a user story to be considered complete and done, are typically developed during these discussions. The Development Team and the Product Owner collaborate to define clear, measurable, and testable acceptance criteria that provide a shared understanding of what needs to be delivered for each user story.

Along with the acceptance criteria, the team also develops acceptance tests during the sprint planning meeting for the planned user stories.

This helps ensure that the user stories are well-defined, and the team has a clear understanding of the expectations for completing the work in the sprint.

# Sprint Review Meeting

The sprint or iteration review meeting is an important event that occurs at the end of each sprint or an iteration. During this meeting, the project  team demonstrates the work they have completed during the sprint or an iteration to stakeholders, customers, and other interested parties.

The purpose of the sprint/iteration review meeting is to review and inspect the working software or product increment developed during the sprint or an iteration, and to obtain feedback from stakeholders.

This feedback is used to inform and adjust the product backlog and the future sprint or iteration plan. The sprint or iteration review meeting also provides an opportunity for the team to celebrate their accomplishments and discuss any issues or challenges they faced during the sprint or iteration.

The sprint or iteration review meeting typically includes the following elements:

1. Demonstration of the working software or product increment
2. Discussion of the progress made during the sprint or iteration
3. Feedback from stakeholders on the work completed and the product backlog
4. Discussion of any changes or adjustments to be made to the product backlog or sprint or iteration plan
5. Celebration of the team's accomplishments

The sprint or iteration review meeting is an important opportunity for the team to receive feedback and adjust their approach as necessary, ensuring that they are delivering value to their stakeholders and customers.

# Swarming

**Agile swarming** takes place when multiple team members with available time and appropriate skill sets all direct their attention to work together on one feature or user story, i.e., they swarm the task until it is complete.

# Swarming vs. Mobbing vs. Pairing

[Swarming and mobbing are agile methodologies that aim to reduce waste in the software development cycle](https://www.bing.com/ck/a?!&&p=a2ab101f4b93e885JmltdHM9MTcwMzYzNTIwMCZpZ3VpZD0yYWZjMWVjMi1lZDM4LTYwODctMDJiYy0wZGUwZWM3ODYxNzYmaW5zaWQ9NTY0MQ&ptn=3&ver=2&hsh=3&fclid=2afc1ec2-ed38-6087-02bc-0de0ec786176&psq=what+is+the+difference+between+swarming+and+mobbing+in+agile&u=a1aHR0cHM6Ly9wbS1wb3dlcmNvbnN1bHRpbmcuY29tL2Jsb2cvc3BlZWQtdXAtc29mdHdhcmUtZGV2ZWxvcG1lbnQtd2l0aC1zd2FybWluZy1hbmQtZnJhY3RhbHMv&ntb=1). Swarming has a WIP (work in progress) limit of 1, where the team collaborates to get the one item to done. [Mobbing has a WIP limit of 1 for an entire team with one keyboard. Pairing is two people and one keyboard, often with a WIP limit of 1. A WIP limit of one means the team or pair works on just one story/feature at a time](https://www.bing.com/ck/a?!&&p=7a0721900abc70e5JmltdHM9MTcwMzYzNTIwMCZpZ3VpZD0yYWZjMWVjMi1lZDM4LTYwODctMDJiYy0wZGUwZWM3ODYxNzYmaW5zaWQ9NTY0Mw&ptn=3&ver=2&hsh=3&fclid=2afc1ec2-ed38-6087-02bc-0de0ec786176&psq=what+is+the+difference+between+swarming+and+mobbing+in+agile&u=a1aHR0cHM6Ly93d3cuanJvdGhtYW4uY29tL21wZC9wcm9qZWN0LW1hbmFnZW1lbnQvMjAxNi8wNy9wYWlyaW5nLXN3YXJtaW5nLWFuZC1tb2JiaW5nLw&ntb=1).

# Stakeholder meetings

Having these meetings on a regular cadence ensures your stakeholders are in constant contact and have regular opportunities to speak with you about changes or concerns.

# Systems Thinking

Systems thinking is a way of making sense of the complexity of the world by looking at it in terms of wholes and relationships rather than by splitting it down into its parts. Used as a way of exploring and developing effective action in complex contexts, enabling systems change.

The most successful Agile projects in the systems thinking approach can be classified as: Having moderately complex technology and moderately complex requirements. The most successful are neither high nor low, but somewhere in the middle of the complexity standards.

## Benefits of systems thinking

* Explore new opportunities
* Minimize impact of mistakes
* Make Realistic plans
* Repair Broken designs
* Maximizing the outcomes achieved
* See the bigger picture
* Greater clarity

## Components of systems thinking

* Interconnectedness
* Synthesis
* Emergence
* Feedback loops
* Causality
* Systems mapping

## Steps of systems thinking

1. Observe the system
2. Identify a problem to solve
3. Understand the problem
4. Identify interventions
5. Select solutions

# Synchronous Continuous Integration Steps

we do code, test, and integrate, and we wait to integrate to finish before we do the next code.

# Task Switching

Being assigned to different projects is an example of task switching, which creates waste. Waste #6: Task Switching: Team members moving from one task to another without completing the first task properly. Following are some of the reasons for Task Switching. Interruptions on the ongoing tasks. Lack of ground level analysis on the tasks required for the stories. Shared team working on more than one project at a time. Lack of proper coordination between product owner and development team

Bottom of Form

# Tabaka Self-Assessment Model

Self-assessment is always part of the **inspect and adapt cycle**. Self-assessment allows the team to examine how it is performing and adjust as the project continues. It is not focused on development, analysis, or testing.

# Tacit Knowledge

Tacit knowledge is knowledge that is learned through experience and not from an external source (such as a class or book).

# Target Date

It is ok for a customer to have a target release date, given that the scope can vary. Once the date is agreed upon, the project team can then provide feedback on what they feel can be ready by then. The other choices are not accurate responses. A customer can choose a target release date, and it can be done before all iterations have been planned.

# Team Augmentation

Team augmentation is a contracting technique in Agile project management that involves adding new team members to an existing Agile team in order to increase capacity or expertise. This approach allows for greater flexibility in staffing and can help to address specific project needs or challenges.

Bottom of Form

# TDD - Test First Development

Test Driven Development (TDD) is a software development approach in which test cases are developed to specify and validate what the code will do. In simple terms, test cases for each functionality are created and tested first and if the test fails then the new code is written to pass the test and make the code simple and bug-free.

# Test-driven development

refers to a style of programming in which three activities are tightly interwoven: coding, testing (in the form of writing unit tests), and design (in the form of refactoring).

# Training Needs

Who should decide training needs for the agile team? Agile teams are made up of “motivated” individuals and hence it is expected that the team best knows what technical/professional training they need.

In an organization that has fully embraced the agile culture, the responsibility of deciding training needs should lie with the teams themselves. As agile is a team-based approach, each team should have the autonomy to identify and prioritize their training needs based on their project requirements, their strengths, and areas of improvement. However, the agile PMO can provide guidance and support to the teams in identifying the appropriate training needs, selecting the training programs, and monitoring the effectiveness of the training programs.

# Trend Analysis

The agile tool that can potentially help the accounting department team detect potential issues before they occur is "trend analysis." Trend analysis involves monitoring and analyzing historical data and patterns to identify trends and patterns that can provide insights into potential issues or risks.

By analyzing historical data, the team can identify patterns or trends that may indicate potential issues in the future. For example, if the team notices a recurring pattern of delayed deliverables or missed deadlines in previous projects, it may signal a potential risk of similar issues occurring in the current project. Similarly, if there is a consistent trend of increasing defects or bugs in previous releases, it may indicate potential quality issues in the current project.

Trend analysis can help the team proactively identify potential issues, risks, or trends that may impact the project's success and take preventive measures to address them early on. It allows the team to take corrective actions and make informed decisions to mitigate risks and ensure smooth project execution.

# Triple Constraint in Project Management

The Triple Constraints of Project Management is essentially a model which represents the constraints that a project is typically 'bound' to. According to the Triple Constraints there are 3 constraints and hence the name Triple Constraints. And those are

* scope - this represents what the project needs to develop
* cost - this represent the financial constraints of a project
* time - the project's schedule, within which the project is supposed to be completed.

For a project manager it is very important to understand that if one of the triple constraints is changed or the adjustment of one constraint will also lead to a change in the rest of the two constraints, in most of the cases.

The way these Triple Constraints of Project Management are handled in traditional and agile projects is very different. Unlike the traditional triangle of constraints, agile projects attempt to fix time and cost and adjust scope to achieve the highest-priority, best-quality product possible within the fixed constraints.

Hence agile projects attempt to fix time and cost and adjust scope to achieve the highest-priority, best-quality product possible within the fixed constraints. is the correct choice.

# Triple Nickel

Triple nickels is a brainstorming technique used in retrospectives that is based on five minutes of brainstorming and discussion.

# Tuckman Ladder Model

Tuckman’s Ladder Model is a tool for understanding the five stages of team development: **forming, storming, norming, performing, and adjourning**. The Tuckman Ladder Model describes the predictable stages that teams move through and explains how you can use them to improve your functioning as a group.

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Stage** | **Leadership Style** | **Team Member Behavior** | **Leader Behavior** |
| Forming | Directing | Low competence High commitment | High directive Low supportive |
| Storming | Coaching | Low/Some competence Low commitment | High directive High supportive |
| Norming | Supporting | Moderate/High competence Variable commitment | Low directive High supportive |
| Performing | Delegating | High competence High commitment | Low directive Low supportive |

# User Story

User story is one of the most commonly used in scrum to represent the requirements, but it is**not the scrum artifact** and any other representation of a business requirement can be used instead of user story too.

# Value Stream Map

A **value stream map** is a visual tool that displays all critical steps in a specific process and quantifies easily the time and volume taken at each stage. **Value stream maps** show the flow of both materials and information as they progress through the process. The purpose of value-stream mapping is to identify and remove or reduce "waste" in value streams, thereby increasing the efficiency of a given value stream. Waste removal is intended to increase productivity by creating leaner operations which in turn make waste and quality problems easier to identify.

# Variance Analysis

Variance analysis is a technique used in project management and financial management to compare the actual performance of a project or task against its planned or budgeted performance, and to identify and analyze any variances or differences between the two.

Variance is the measure of how far apart things are, how much they vary from each other - in the above scenario, the team is measuring the planned vs. actual - which is an example of Variance analysis

# Wideband delphi Technique

This technique involves soliciting feedback and ideas anonymously then sharing them with the group for discussion.

# Working Agreement

Working Agreements are detailed guidelines that are established by the team to define how they will work together to achieve their goals. **They are more specific than a Team Charter and Ground Rules**and cover topics like meeting schedules, decision-making processes, and communication protocols. Working Agreements are created collaboratively by the team members during the early stages of the project and are revisited regularly to ensure they are still effective.

# Working Agreement/Team Charter

As an Agile Lead, you can help facilitate the development of a team coordination and social contract artifact for the newly formed team working on nuclear reactor component development. This artifact can be in the form of a Team Charter or a Working Agreement.

Both artifacts serve as written agreements or guidelines that outline how team members will interact with each other to foster a collaborative and productive work environment.

Team charter is the document developed by the project development team that clearly outlines how the team members should interact with each other during the project execution. It acts like a social contract for the team.

Team Charter serves as reference document that can help the team establish a shared understanding of how they will work together, set expectations, and promote a collaborative and productive team environment.

It can be created collaboratively by the team members during a team kickoff or planning session and should be reviewed and updated periodically as needed to ensure they remain relevant and effective throughout the duration of the nuclear reactor component development project.

# Working Software

Working software is the primary measure of progress as per the Agile Manifesto principles. Working software is the primary measure of progress. In most cases, working software provides value to the end user, has limited bugs, and is high performance. If we can provide additional value to the end user after each sprint, then we're showing strong signs of progress.

# Workshop

A meeting to define activities and contributions.

# XP Coach

Responsible for following up on issues raised in stand-up meetings, facilitating meetings, and helping to communicate the project vision, goals, and backlog items to the delivery team.

# XP Core Value

In Extreme Programming (XP), respect is one of the core values that guide the team's behavior and shapes the way they work together. The value of respect is based on the belief that everyone on the team has valuable contributions to make and deserves to be treated with dignity and consideration. By embracing the value of respect, XP teams foster a positive and supportive working environment that encourages creativity, innovation, and continuous improvement. This helps build trust, strengthen relationships, and ultimately deliver better quality software.

# XP System Metaphor

An Extreme Programming system metaphor is a practice that is used by XP developers to replace the standard project architecture used in traditional software development methodologies.

# XP Team Members

Coach, Tracker, Customer, Programmer, Tester, Developer, Doomsayer, Gold Owner, manager, XP tracker

# XP – 4 Basic Activities

Coding, Testing, Listening, and designing