PMI-ACP Quick Reference Guide

Domain I. **Agile Principles and Mindset** (9 tasks) - 16% ~19 questions among 120 Exam questions
Explore, embrace, and apply agile principles and mindset within the context of the project team and organization.
Agile Manifesto Values and Principles, Agile Framework and Terminology & Agile Methods and Approaches.

Domain II. Value-Driven Delivery (4 sub-domains, 14 tasks) - 20% ~24 questions among 120 Exam questions
Deliver valuable results by producing high-value increments for review, early and often, based on stakeholder priorities. Have the stakeholders provide feedback on these increments, and use this feedback to prioritize and improve future increments.
ROI, NPV, IRR, Chartering, Relative Prioritization, Customer-Valued Prioritization, Risk-Adjusted Backlog, Agile EVM, Task / Kanban Boards & WIP Limits, Cumulative Flow Diagrams, Value Stream Mapping, Requirement Reviews, KANO Analysis & MoSCoW, Prioritization, Value-Based Decomposition, and Prioritization, Incremental Delivery, Value-Based Analysis, Business Case Development, Agile Project Accounting Principles, Agile Contracting, Agile Project Accounting Principles.

Domain III. Stakeholder Engagement (3 sub-domains, 9 tasks) - 17% ~20 questions among 120 Exam questions
Engage current and future interested parties by building a trusting environment that aligns their needs and expectations and balances their requests with an understanding of the cost/effort involved.
Promote participation and collaboration throughout the project life cycle and provide the tools for effective and informed decision-making.
Personas, User Stories / Backlogs + Story Maps, Wireframes, Information Radiators, Burn Down / Up Charts, Agile Modeling, Conflict Resolution + Negotiation, Workshops, Two-way Communications, Social Media-based Communication, Accessing and Incorporating Community and Stakeholder Values, Stakeholder Management, Communication Management, Facilitation Methods, Knowledge Sharing/ Written Communication.

Domain IV. Team Performance (3 sub-domains, 9 tasks) - 16% ~19 questions among 120 Exam questions
Create an environment of trust, learning, collaboration, and conflict resolution that promotes team self-organization enhances relationships among team members *and cultivates a culture of high performance*.
Servant Leadership + Adaptive Leadership, Co-located Teams / Distributed Teams, Team Space, Agile Tooling + Daily Stand-ups, Emotional Intelligence, Learning Cycle, Productivity, Leadership, Building Agile Teams, Team Motivation, Physical and Virtual Co-location, Global, Cultural, and Team Diversity, Training, Coaching, and Team Diversity, Participatory Decision Models.

Domain V. Adaptive Planning (3 sub-domains, 10 tasks) - 12% ~14 questions among 120 Exam questions
Produce and maintain an evolving plan, from initiation to closure, based on goals, values, risks, constraints, stakeholder feedback, and review findings.
Product Roadmap, Minimally Marketable Feature (MMF) / Minimal Viable Product (MVP), Iteration and Release Planning + Progressive Elaboration, Wide Band Delphi and Planning Poker + Affinity

Estimating + Relative Sizing / Story Points, Time-boxing, Process Tailoring, Velocity + Ideal Time + Throughput, Backlog Grooming / Refinement, Definition of Done, Principles of Systems Thinking, Prioritization, Incremental Delivery, Agile Discovery, Agile Sizing and Estimation, Value-Based Analysis and Decomposition, Agile Project Chartering

Domain VI. Problem Detection and Resolution (5 tasks) - 10% ~12 questions among 120 Exam questions
Continuously identify problems, impediments, and risks; prioritize and resolve in a timely manner; monitor and communicate the problem resolution status, and implement process improvements to prevent them from occurring again.
Continuous Integration + Frequent Verification and Validation, Variance and Trend Analysis, Test-Driven Development / Test First Development
Acceptance Test-Driven Development, Risk-Based Spike, Risk-Adjusted Backlog, Architectural Spikes, Risk Burn Down Graphs, Cycle Time, Escaped Defects, Approved Iterations, The Five WHYs, Control Limits, pre-mortem (rule setting, failure analysis), Fishbone Diagram Analysis, Testing (including exploratory and usability), Problem-Solving, Process Analysis, Regulatory Compliance, Managing with Agile KPIs

Domain VII. Continuous Improvement (Product, Process, People) (6 tasks) - 9% ~ 11 questions among 120 Exam questions
Continuously improve the quality, effectiveness, and value of the product, the process, and the team.
Retrospectives, Product-Feedback Loop, Process Tailoring/Hybrid Models, Value Stream Mapping, Continuous Integration, Kaizen, Developmental Mastery Models, Self-assessment Tools and Techniques, Continuous Improvement, Agile Hybrid Models, PMI's Code of Ethics, and Professional Conduct.

Tech Agilist practice exam on UDEMY

CSM - https://www.udemy.com/course/certified-scrum-master-csm-practice-exams-v/?referralCode=338E2561C3C12C8FE96F
PSM I - https://www.udemy.com/course/scrum-master-certification-practice-exams/?referralCode=358827C432F98F5F42C2
PSM II - https://www.udemy.com/course/professional-scrum-master-ii-psm-ii-practice-exams/?referralCode=026579CD2EB531CB43C6
PSPO I - https://www.udemy.com/course/product-owner-certification-practice-exams/?referralCode=FE7E6BDBA18CFF6E6A72
PSPO II - https://www.udemy.com/course/professional-scrum-product-owner-ii-pspo-ii-practice-exams/?referralCode=538A23D1E564838678AD
PSD - https://www.udemy.com/course/scrum-developer-certification-practice-exams/?referralCode=4EF272249331815DDF85
SPS - https://www.udemy.com/course/scaled-professional-scrum-sps-certification-practice-exams/?referralCode=F41C05C8F07F62E0501C
PMI-ACP - https://www.udemy.com/course/pmi-agile-certified-practitioner-pmi-acp-practice-exams-m/?referralCode=583ACDE0C9A5FEC0E38E
PMP - https://www.udemy.com/course/pmp-certification-practice-exams-pmi-pmp-pmbok7-pmbok6/?referralCode=A03B351C8C0B7D472040

Domain I. **Agile Principles and Mindset** (9 tasks) - 16% ~19 questions among 120 Exam questions

Explore, embrace, and apply agile principles and mindset within the context of the project team and organization.

Agile Manifesto Values and Principles, Agile Framework and Terminology & Agile Methods and Approaches.

Agile Manifesto

- Individuals and interactions over Processes and tools.
- Working software over Comprehensive documentation.
- Customer collaboration over Contract negotiation.
- Responding to change over Following a plan.

Agile Principles: Customer Satisfaction, Welcome Changes, Frequent Delivery, Collocated Teams, Motivated Individuals, Face to Face Contact, Working Software, Constant/Sustainable Pace, Continuous Attention, Simplicity, Self-Organization & Regular Reflection.

Declaration of Interdependence (DOI)

- We increase return on investment by making continuous flow of value our focus.
- We deliver reliable results by engaging customers in frequent interactions and shared ownership.
- We expect uncertainty and manage it through iterations, anticipation, and adaptation.
- We unleash creativity and innovation by recognizing that individuals are the ultimate source of value and creating an environment where they can make a difference.
- We boost performance through group accountability for results and shared responsibility for team effectiveness.
- We improve effectiveness and reliability through situationally specific strategies, processes, and practices.

Waterfall vs Agile project management: Former is plan-driven and the latter is value driven.

Empirical Process:

- **Transparency:** This encompasses not just the process itself but all communications.
- **Inspection:** Frequent inspection and the utilization of frequent reviews of the product service or results is essential.
- **Inspection:** Frequent inspection and the utilization of frequent reviews of the product service or results is essential.

The Five Values of Scrum: Commitment, Focus, Openness, Respect & Courage.

Scrum Roles: Product Owner (PO), Scrum Master & Developers.

Scrum Ceremonies: Sprint Planning, Daily Scrum Meeting & Sprint Review Sprint Retrospective

Scrum Artifacts: Product Backlog, Sprint Backlog, Definition of Done, Product Increment & Burndown Charts.

Agile Methodologies: Scrum, XP (eXtreme Programming), Kanban, LSD (Lean Software Development), Crystal Family, FDD (Feature Driven Development), ASD (Adaptive Software Development), DSDM (Dynamic Systems Development Method.

Core Values in Extreme Programming: Communication, Simplicity, Feedback, Courage & Respect

Lean 5S Tool for Improvement

- **Sort (Seiri)** – remove unnecessary materials from the workplace.
- **Set in order (Seiton)** – arrange all items in a proper sequence, which facilitates a smooth flow.
- **Shine (Seiso)** – periodically inspect the workplace and keep it clean.
- **Standardize (Seiketsu)** – follow standard practices and processes at the workplace.
- **Sustain (Shitsuke)** – maintain order, discipline, and good working conditions.

Principles of Lean Thinking

- Eliminate Waste
- Amplify Learning
- Decide as Late as Possible
- Deliver as Fast as Possible
- Empower the Team
- Build Integrity In
- Optimize the Whole

Principles in DSDM

- Focus on the business need
- Deliver on time
- Collaborate
- Never compromise quality
- Build incrementally from firm foundations
- Develop iteratively
- Communicate continuously and clearly
- Demonstrate control

Feature-Driven Development (FDD)

- Develop overall model
- Build feature list
- Plan by feature
- Design by feature
- Build by feature

Domain II. Value-Driven Delivery (4 sub-domains, 14 tasks) - 20% ~24 questions among 120 Exam questions

Deliver valuable results by producing high-value increments for review, early and often, based on stakeholder priorities. Have the stakeholders provide feedback on these increments, and use this feedback to prioritize and improve future increments.

ROI, NPV, IRR, Chartering, Relative Prioritization, Customer-Valued Prioritization, Risk-Adjusted Backlog, Agile EVM, Task / Kanban Boards & WIP Limits, Cumulative Flow Diagrams, Value Stream Mapping, Requirement Reviews, KANO Analysis & MoSCoW, Prioritization, Value-Based Decomposition, and Prioritization, Incremental Delivery, Value-Based Analysis, Business Case Development, Agile Project Accounting Principles, Agile Contracting, Agile Project Accounting Principles.

Economic models

- Present Value (PV) = FV / $(1+r)^n$ where FV is Future Value, r is interest rate & n is the number of time periods.
- Net Present Value (NPV) = - Initial investment + Sum of each ear in today's terms, a positive NPV means the project is profitable.
- Return on Investment (ROI) or Benefit Cost Ration (BCR) = Net Benefit of the project/Total cost, a positive ROI means the project is profitable.
- Internal Rate of Return (IRR) = The discount rate at which the project inflows and outflows are equal. The higher the positive IRR, the more profitable the project.

Agile Charters: W5H

- What is the scope of the project?
- Why is the project being undertaken?
- Who will be impacted because of the project?
- When will the project start and end?
- Where will the project occur or deliver?
- How will the project be executed?

Little's law, WIP = Lead time x Throughput. Lead Time – Average time an item spends in the system, Throughput – Average rate at which work departs or is completed.

Reduce Cycle Time

- Reduce variability in the rate of arrival.
- Reduce variability in the rate of processing.
- Limiting work in progress.
- Removing blockers, and waiting times.
- Investing in smart engineering tools and practices.
- Investing in a cross-functional team.

CFD to visualize the relationship between work in progress and lead time to identify bottlenecks.

Value stream map = Supplier—> Inputs—> Processing—> Outputs—> Customer.

Value-based prioritization technique

- **Numerical Assignment:** Prioritize business requirements in order of value is to rank them in priorities of 1, 2, 3, or High, Medium, Low, and so on.
- **Analytical Hierarchical Process (AHP):** A structured technique used for complex decision-making in a group environment.
- **100-point or Cumulative Voting Method:** It is like an opinion poll for determining the priority of items in a group environment.
- **MoSCoW:** Must have, Should have, Could have, Won't have
- **Kano Analysis Model:** Using the model, the product owner and the team are able to prioritize a product feature in one of the five categories mentioned below and expend resources to deliver them - Basic / Must-be, Attractive/Delighters, One-dimensional/Performance, Neutral/Indifferent, Reverse
- **Wieger's method:** It is a quantitative analytical approach to prioritization13 that makes a weighted computation of value, cost, and risk associated with a requirement.

Minimally Marketable Features (MMF): The minimal functionality set (a group of user stories or a package of features) that can deliver values (e.g. useful) to the customers / end-users a distinct and deliverable feature of the system that provides significant values to the customer (can be sold/used immediately) chosen for implementation after value-based prioritization can reap the return on investment instantly

Minimal Viable Product (MVP): The minimal product (with just essential features and no more) that allows can be shipped to early adopters to see and learn from the feedback instantly. The concept is somewhat similar to Minimally Marketable Feature (MMF) in which MVP is the first shippable product with the first set of MMF.

DEEP attributes of the product backlog:

- D - Detailed appropriately
- E – Estimable
- E - Emergent
- P - Prioritized

Agile Metrics and KPIs:

- Planned versus Actual Velocity
- Release Burndown charts
- Burnup charts
- Combined Burnup and Burndown Charts
- Iteration Burndown Charts
- Parking Lot Chart
- Kanban board / Task Board
- Earned Value Management (EVM)
    - **Schedule Performance Index (SPI)** = Earned Value (e.g., in story points) / Planned Value (in story points). An SPI less than 1 signifies that the project is behind schedule whereas a value greater than 1 means the project is ahead of schedule.

- **Cost Performance Index (CPI)** = Earned Value (e.g., in the equivalent value of story points completed) / Actual costs incurred (money spent till date). A CPI less than 1 signifies that the project is over budget schedule whereas a value greater than 1 means the project is under budget.

Domain III. Stakeholder Engagement (3 sub-domains, 9 tasks) - 17% ~20 questions among 120 Exam questions

Engage current and future interested parties by building a trusting environment that aligns their needs and expectations and balances their requests with an understanding of the cost/effort involved. Promote participation and collaboration throughout the project life cycle and provide the tools for effective and informed decision-making.

Personas, User Stories / Backlogs + Story Maps, Wireframes, Information Radiators, Burn Down / Up Charts, Agile Modeling, Conflict Resolution + Negotiation, Workshops, Two-way Communications, Social Media-based Communication, Accessing and Incorporating Community and Stakeholder Values, Stakeholder Management, Communication Management, Facilitation Methods, Knowledge Sharing/ Written Communication.

Definition and identification of stakeholders: Stakeholders of a project are individuals, groups, or organizations that are affected or perceived to be affected either positively or negatively by a decision, activity, or outcome of a project.

Classification of stakeholders on basis of Engagement Levels :

- **Unaware** – Stakeholders who are unaware of the project and potential impacts.
- **Resistant** – Aware of the project and resistant to change.
- **Neutral** – Neither supportive nor resistant.
- **Supportive** – Would like to see the change happen and has a positive outlook toward it.
- **Leading** – Actively engaged in ensuring the project is a success.

Managing Stakeholders: Managing Communication, Managing Vendors, Managing Distributed Teams.

Group Decision-Making Techniques

- Styles of Group Decision-Making: Command, Consultative, Consensus.
- Methods of Reaching a Decision: Unanimity, Majority, Plurality & Dictatorship.
- Thumbs Up/Down/Sideways.
- Fist-of-Five Voting.

Definition of done (DoD): Done means the feature is 100% complete according to pre-agreed conditions (e.g. including all the way from analysis, design, and coding to user acceptance testing and delivery & documentation) and ready for production (shippable).

- **Done for a feature:** feature/backlog item completed.
- **Done for a sprint:** work for a sprint completed.
- **Done for a release:** features are shippable.

Information Radiators: Agile teams follow a tradition of communicating and visualizing progress very transparently using information radiators. They put up a variety of artifacts and metrics on prominent public spaces like a large whiteboard or a wall in the corridors or hallways so that they can be effortlessly viewed by anyone walking past them.

Interpersonal Skills for Managing Stakeholders

- Emotional Intelligence

- Collaboration
- Motivating
- Active Listening
  - **Level I** – Internal Listening (thinking about how things will affect me).
  - **Level II** – Focused Listening (trying to understand what the speaker is really trying to say).
  - **Level III** – Global Listening (keep track of not only what has been said but also the different signs and gestures the speaker employs to convey the full message).
- Negotiation

Steps for Negotiation

- Determine the power, influence, legitimacy, history, and background of the party that you are going to negotiate with.
- Determine what stance you would need to adopt and the minimum that you would settle for.
- Do a SWOT (Strength, Weakness, Opportunity, and Threat) analysis of the negotiating parties that will help to devise a strategy.
- Be objective and separate people from the problem (the negotiation topic).
- Anticipate issues and reactions from negotiating parties to help in arguments and counterarguments.
- Arm yourselves with data and facts to justify your stance (and weaken the one from the other side).
- Determine which of the negotiation tactics is most applicable to generate a win-win outcome or the most favorable one.
- Remain professional at all times and be diligent in your argument. But save some tactics up your sleeve for a last resort.
- Envision the worst-case result if a favorable decision does not arrive at the end of the negotiation. Have contingency plans in place.

Conflict Resolution Techniques:

- **Problem-Solving / confronting** – solving problems by examining alternatives.
- **Compromise** – seeking common ground that brings in some degree of satisfaction for all parties.
- **Smoothing/accommodating** – emphasizing areas of agreement rather than differences.
- **Collaborating** – considering multiple views and perspectives, ultimately leading to consensus and commitment.
- **Withdrawing/avoiding** – retreating from a conflicting situation.
- **Forcing** – pushing one's view over others, resulting in win-lose outcomes.

Project Charter:

- **Vision**: the purpose of the Agile project – answering the "why" of the project.
- **Mission**: describes what will be achieved or done – answering the "what" of the project.
- **Success Criteria:** describe how the project will be considered a success or reach an end.

Domain IV. Team Performance (3 sub-domains, 9 tasks) - 16% ~19 questions among 120 Exam questions

Create an environment of trust, learning, collaboration, and conflict resolution that promotes team self-organization enhances relationships among team members, *and cultivates a culture of high performance*.

Servant Leadership + Adaptive Leadership, Co-located Teams / Distributed Teams, Team Space, Agile Tooling + Daily Stand-ups, Emotional Intelligence, Learning Cycle, Productivity, Leadership, Building Agile Teams, Team Motivation, Physical and Virtual Co-location, Global, Cultural, and Team Diversity, Training, Coaching, and Team Diversity, Participatory Decision Models.

Team Formation Stages: Bruce Tuckman's theory of team building

- **Forming** – This is the stage where the team members come together, mostly focused on themselves, their roles, responsibilities, and goals rather than that of the team.
- **Storming** – This is the stage where the team members begin to work on the projects, bring about their individualities, voice their opinions, and at times, attempt to dominate.
- **Norming** – During this stage, the supervisor could intervene, set ground rules and emphasize what the professional behavior should be like.
- **Performing** – This is the highest ('nirvana') stage where teams are motivated to achieve the goals of the project.
- **Mourning or Adjourning** – This is the stage where the project is over and team members get released.

Shu-Ha-Ri Model:

- The starting point is Shu, which means learn & follow the rule.
- The next level is Ha, which means detach & break the rule.
- The final level of mastery is Ri, which means transcend & be the rule.

Dreyfus Model:

- **Novice** – at this skill level, the team members have a very shallow understanding and need very close supervision.
- **Advanced beginner** – at this skill level, the understanding of the steps involved is such that they can apply the same steps in a similar context.
- **Competent** – at this skill level the team members have attained a good understanding of the steps involved and are able to complete their tasks properly without supervision.
- **Proficient** – at this skill level, team members have gathered sufficient practice, experience, and a deep understanding of the subject.
- **Expert** – this is the highest skill level. With lots of knowledge amassed, the team members achieve excellence and go beyond existing interpretations, rules, and guidelines.

Situational Leadership Model:

- **Telling style** – During stage 1, the agile leader is working with a team having low competence and less experience.

- **Selling style** – During stage 2, the team members are still a novice, inexperienced, and unable to take responsibility, but they are enthusiastic and willing to work.
- **Participating style** – During stage 3, the team members are experienced and have the required skills, but lack the confidence to get the job done all on their own.
- **Delegating style** – During the highest stage 4, the team members are mature enough, empowered, and capable of handling a task confidently without the intervention of the leader.

High-Performing Teams

- Takes accountability and ownership.
- Demonstrates commitment consistently.
- Cross-functional.
- Generates positivity in the team environment.
- Adaptive to change.
- Trust-worthy.
- Self-organized.
- Open, clear and transparent communication.
- Focused on the customer's needs.
- Empowered to make decisions as individuals or through group consensus.
- Has an eye for removing non-value-added activities.
- Values diversity and builds strong team bonds.
- Works at a pace that is consistent and sustainable.
- Regularly inspects and adapts processes and practices.
- Takes pride in achievements.

Information radiator – a communication tool to physically displays key information about the current project status to the Agile team/stakeholders in the work area in the most visible and efficient manner, e.g. Kanban boards.

Emotional Intelligence:

- Self-awareness
- Emotional resilience
- Motivation
- Interpersonal sensitivity
- Influence
- Intuitiveness
- Conscientiousness

Communication among Team: Release planning meetings, Iteration planning meetings, Daily stand-up meetings, Iteration reviews & Retrospectives meetings.

Osmotic Communication: The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. Team members in a co-located space can overhear conversations/discussions of other members. The team members will be able to extract useful parts from the conversations or join in if necessary.

Systems Thinking: While planning for value-driven delivery, the holistic picture helps team members to think of the system as a whole, its complex assembly of closely integrated components, and how each component interfaces with each other as one unit.

BART Analysis of Team: It is a tool used to identify problems and the effectiveness of processes in agile teams. The acronym BART stands for boundary, authority, role, and task.

Contract Types in Agile Projects:

- Fixed Price, but with Provision for Change in Scope in Future Iterations.
- Contract with Premature Closure Clause.
- Fixed Fee and Not-to-Exceed Clauses.
- Fixed Price per Story Point.
- Multi-Stage Contracts.
- Target Cost Contract.
- Contract Extension and Payment Based on Delivery and Acceptance.

Functions of an Agile PMO: Agile transformation, Agile adoption and training, Agile coaching, Agile governance and center of excellence, Resource management, Vendor management, Reporting, Audits and compliance & Continuous improvement.

Domain V. Adaptive Planning (3 sub-domains, 10 tasks) - 12% ~14 questions among 120 Exam questions

Produce and maintain an evolving plan, from initiation to closure, based on goals, values, risks, constraints, stakeholder feedback, and review findings.

Product Roadmap, Minimally Marketable Feature (MMF) / Minimal Viable Product (MVP), Iteration and Release Planning + Progressive Elaboration, Wide Band Delphi and Planning Poker + Affinity Estimating + Relative Sizing / Story Points, Time-boxing, Process Tailoring, Velocity + Ideal Time + Throughput, Backlog Grooming / Refinement, Definition of Done, Principles of Systems Thinking, Prioritization, Incremental Delivery, Agile Discovery, Agile Sizing, and Estimation, Value-Based Analysis and Decomposition, Agile Project Chartering

Return on investment (ROI): ROI is being calculated using the formula of benefit to cost.

- ROI = (Gain - Cost) / Cost
- Discounted ROI considers the future value of the money (benefits and costs).
    - Discounted ROI = NPV of Benefits / NPV of Costs
- ROI is also calculated by multiplying the velocity against the margin of the project. ROI = Velocity × Margin

Earned value technique (EVT):

- Planned story points of the release = Sum of story points planned for the iterations.
- Completed story points of the release = Sum of story points completed of the iterations.
- Earned value of the release = Sum of the earned value of the iterations.

Kano analysis: Kano analysis prioritizes customer requirements as customer satisfaction.

- **Must Be**: Whatever the quality characteristic is, it must be present, such that if it is not, the customer will go elsewhere.
- **Performance**: The better we are at meeting these needs, the happier the customer is.
- **Delighter**: Those qualities that the customer was not expecting, but received as a bonus.

Important formula

- Schedule Variance (SV) = Earned value (EV) – Planned value (PV)
- Cost Variance (CV) = Earned value (EV) – Actual cost (AC)
- Schedule Performance Index (SPI) = Earned value (EV) / planned value (PV)
- Cost Performance Index (CPI) = Earned value (EV) / actual cost (AC)
- Lead Time = Work In Progress (units) / Average Completion Rate (units per time period)
- Work In Progress = Lead Time* Average Completion Rate (units)
- Average Completion Rate = Work In Progress / Lead Time

MoSCoW Approach:

- MUST have this requirement to meet the business needs.
- SHOULD have this requirement if possible, but the project's success does not rely on it.
- COULD have this requirement if it does not affect the business needs of the project.

- WON'T have this requirement, and the stakeholders have agreed that it will not be implemented in a release but may be considered for the future.

Adaptive planning/rolling wave planning/progressive elaboration: A multi-step, intermittent process of planning which is based on the philosophy that the plan can be detailed out only as more details are revealed with time and with the changing scenario in the project.

Core Agile project management phases:

- Envisioning
- Speculating
- Exploring
- Adapting
- Closing

Agile Planning Stages

- **Product Vision** –A document created by the product owner describing what the product is, who will and why use it, and how the product supports company strategy. [revised once a year].
- **Product Roadmap** – A document created by the product owner describing the high-level product requirements and the timeframes for deliverables, providing a visual overview of all the planned releases and major components. [revised twice a year].
- **Release Plan** –A document created by the product owner describing the high-level timeline for product releases (features with higher values are given higher priority in the releases). [revised twice / four times a year].
- **Personas** – a tool used in requirement collection and testing in which realistic depictions of likely users for the product are created, these users can be real or fictitious.
- **Extreme Persona** – extreme persona is a persona taken to the extreme (with extreme characters/ requirements) in order to identify user stories that would be missed otherwise.
- **Wireframes** – sketch graphical presentations of how the requirements are fulfilled (usually as interface designs), can act as a kind of fast requirement documentation.

Sprint Plan / Iteration Plan – A document created by the product owner, scrum master, and development team describing sprint goals, tasks, and requirements and how those tasks will be completed [revised once a month/iteration].

- Iteration ZERO is for carrying out tasks before the actual development work begins for technical and architectural setup (e.g. spikes) and gathering initial requirements into the backlog, etc.
- Iteration H represents hardening iteration which is the time used to test and prepare the launch software.
- Iteration Planning vs Release Planning
  - Iteration planning involves schedule development at a lower/more detailed level about tasks and time.
  - Release planning involves schedule development at a high level about features and iterations.

Daily Stand-up / Daily Scrum – [daily for 15 minutes] a planning meeting to be attended by the project team and stakeholders (as observers only).

Sprint Review – A meeting scheduled at the end of each sprint for demonstration of working product/ increment/deliverable to stakeholders for feedback and/or acceptance [monthly, at least an hour, for scrum].

Sprint Retrospective – [monthly, at least an hour, for scrum] a meeting scheduled at the end of each sprint to be attended by team members only, will discuss improvements on product and process to enhance efficiency and effectiveness.

Retrospective Meeting vs Review Meeting

- The retrospective meeting is for the development team only (stakeholders are not invited) with the primary purpose of process improvement.
- The review meeting is for the demonstration of deliverables with management, product owner, and stakeholders; new backlog item(s) may be identified together with the customers to be added in the next iteration.

Retrospective Meeting vs Lessons Learned Meeting

- A retrospective meeting is carried out once per iteration to timely identify areas for improvement for immediate action to benefit the project itself.
- Lessons Learned meeting (in traditional project management) is carried out at the end of the project/phase as the project closure activity and all the lessons learned are to be identified and documented (according to PMBOK® Guide) so that they will benefit upcoming projects.

Agile Modeling

- Agile Modeling is a practice-based methodology (including a collection of values, principles, and practice) for effective modeling and documentation of software-based systems based on best practices.
  - a model is a pre-defined way of doing things.
- Agile Modeling is more flexible than traditional modeling methods to be used in traditional project management in order to fit the fast-changing environments of Agile projects.

Agile Planning Terms

- **Agile Themes** – a theme is usually assigned for an iteration in which similar functions are grouped to be done in a batch to maintain a focus of the team, e.g. bug fixes, reporting, etc.
- **Epic Story** – a large block of functionality (usually requires several iterations), epic stories will later be disaggregated into smaller user stories for estimation and implementation.
- **User Story** – usually takes the format of "As a [role], I want [need] so that [business value]"
  - User stories need to be Independent, Valuable, Estimable, Small, and Testable [INVEST].
  - User stories need to be Specific, Measurable, Attainable, Relevant, and Timely [SMART].
  - Ron Jeffries' three Cs of a user story Card, Conversation, Confirmation.
- **Story Maps** – are the overview of how different user stories are related to each other in the project.
- **Features** – a capabilities/group of functionalities that are of value to the end user.

- **Tasks** – the underlying jobs/development work to fulfill a user story, tasks are taken up by the team members through self-organization.
- **Spikes** – a short experimental test to help decisions making, e.g. trying a new technology for a feasibility study.
- **Architectural spikes** – an investigation taken to explore the architectural aspects of the setup.
- **Time-boxing** - time-boxing is a concept for time management by treating time as fixed blocks, once the allotted time (time-box) is up, the work must be stopped regardless of whether it has been finished with a fixed start time, fixed end time, and fixed duration for the activity to control the risk and progress time-boxing allows the team to focus on the essential works and reduce wastes.

Agile estimation and sizing

- **Relative Sizing** – Agile project management makes use of relative sizing (e.g. story points) as opposed to the use of exact units like money and time in traditional project management for estimation as Agile projects are more prone to changes, making use of relative sizing will be more flexible yet still give a reference for meaningful estimation.
- **Ideal Time** – a unit used in the estimation of Agile tasks: ideal time is a block of the uninterrupted period to focus solely on the task without any distractions e.g. email, phone call, toilet break, etc. Though ideal time is NOT realistic in the actual world, it does give an accurate unit to begin working with (e.g. by multiplication of a factor of 2 to 3 to give a reasonable estimate).
- **Wideband Delphi Estimating** – similar to the Delphi technique but discussion about details of the requirements is allowed in the beginning to allow each individual to have a common understanding of the scope of the tasks, each participant will then try to give an estimate for the user stories, etc. with relative sizing on their own; repeat the process until a consensus is reached.
- **Planning Poker** – each member need to select from a deck of cards (with ?, 0, 1, 2, 3, 5 …) to express their estimation of the story points for a user story, discussion follows until a consensus is reached.
- **Affinity Estimating / T-shirt Sizing** – assign a size (e.g. T-shirt sizing: S, M, L, XL, XXL) to user stories instead of giving a more concrete unit, this method is ideal if the scope/details of the task are not quite concrete.

Domain VI. Problem Detection and Resolution (5 tasks) - 10% ~12 questions among 120 Exam questions

Continuously identify problems, impediments, and risks; prioritize and resolve in a timely manner; monitor and communicate the problem resolution status, and implement process improvements to prevent them from occurring again.

Continuous Integration + Frequent Verification and Validation, Variance and Trend Analysis, Test-Driven Development / Test First Development

Acceptance Test-Driven Development, Risk-Based Spike, Risk-Adjusted Backlog, Architectural Spikes, Risk Burn Down Graphs, Cycle Time, Escaped Defects, Approved Iterations, The Five WHYs, Control Limits, pre-mortem (rule setting, failure analysis), Fishbone Diagram Analysis, Testing (including exploratory and usability), Problem-Solving, Process Analysis, Regulatory Compliance, Managing with Agile KPIs

Refactoring: Refactoring is the process of restructuring existing internal computer code without changing its external behaviors.

Burn-up Charts: Used to track the project's progress by comparing total scope against completed work over time. One line is the scope and the other is completed work. Can also help track scope changes.

Burn down Charts: Compares how much work is left to do with the timeline and helps the team predict when they will complete the work.

Risk Burn down Charts: Shows risk exposure over time. The lines should be trending down as risk is mitigated; if not, the team will have to allocate some time to mitigate the risks.

Architectural Spike: Quick experiment (Focus on architecture with high risk) used to help the team answer a question and determine a path forward.

Risk-adjusted backlog: Value-generating business features and risk-reduction actions.

Risk-based spikes: Quick experience used to help the team answer a question and determine a path forward.

Risk burn-down charts: A chart where the risk to project success associated with each feature is displayed.

Risk-adjusted backlog: A backlog that is adjusted to accommodate risk that needs to be mitigated while still accomplishing work on the features.
Risks are assessed by risk probability (how likely it occurs) and risk impact (how severe the risk impact is):

- **Risk Severity** = Risk Probability x Risk Impact
- Risk probability can be a percentage value or a number on a relative scale (the higher the more likely).
- Risk impact can be dollar value or a number on a relative scale (the higher the more costly to fix).

Discover and identify risks:
- Brainstorming and information-gathering techniques
- Delphi technique
- Checklist analysis
- Diagramming techniques
- Assumption analysis
- SWOT analysis
- Expert judgment

Risk categories:
- **Business risk**: e.g., value, priority, satisfaction
- **Technical risk**: e.g., code complexities/technical uncertainties
- **Logistical risk**: e.g., scheduling, resourcing

Strategies for Negative Risks: Avoidance, Transference, Mitigation

Strategies for Positive Risks or opportunities: Exploit, Share, Enhance

Strategies for Both Threats and Opportunities: Contingency Reserve & Management Reserve

Monitoring and Controlling Risks: 5 Whys, Fishbone Diagram Analysis, Divide and conquer, Expert judgment, Simulation, Brainstorming, Metaphors, Trial and error, Spikes, Trend analysis, Probing, Sandboxing, Project risk response audits, Periodic project risk reviews, Technical performance measurement, Additional risk response planning, Workarounds, Reserve analysis & Daily status meetings.

Domain VII. Continuous Improvement (Product, Process, People) (6 tasks) - 9% ~ 11 questions among 120 Exam questions

Continuously improve the quality, effectiveness, and value of the product, the process, and the team.

Retrospectives, Product-Feedback Loop, Process Tailoring/Hybrid Models, Value Stream Mapping, Continuous Integration, Kaizen, Developmental Mastery Models, Self-assessment Tools and Techniques, Continuous Improvement, Agile Hybrid Models, PMI's Code of Ethics, and Professional Conduct.

Product improvement

- Continuous Integration.
- Continuous improvement: PDCA Cycle (Plan – Do – Check -Act), code refactoring, and pair programming.
- Dissemination of knowledge.

Process improvement

- **Kaizen**: Kaizen is the practice of continuous improvement. The Japanese word Kaizen is a combination of two words Kai and Zen, which means to change for the better.
- **Process analysis:** Improving efficiency and effectiveness of processes by identifying and removing overheads, constraints, or anything that does not add value.
- **Lean 5S technique:** The acronym 5S stands for sort, set in order, shine, standardize, and sustain. The goal of this technique is to remove waste materials from the workspace, keep it clean periodically and follow standardized processes to facilitate a smooth flow.
- **Kanban Kata**: Kanban Kata is another continuous improvement strategy that uses a series of questions to help improve in small steps such that day-to-day work and improvements happen simultaneously. A set of Kata questions could look like as follows:
    - What is the target state that we are trying to achieve?
    - What is the current state?
    - What are the impediments or blockers on the way?
    - What is the next step to resolve the impediment?
    - When can we see what we learned from taking that step?
- **5 Why's technique**: The 5 Why's technique is used by teams to identify the root cause of a particular problem by asking a series of 'why' questions. The questions explore the cause-and-effect relationships between observations.
- **Fishbone diagram:** The fishbone diagram, also called the Ishikawa diagram, is an extension of the 5 why's technique to determine the root cause of a problem. It asks a series of questions to explore cause-effect relationships until the addressable root cause is determined.
- **Pareto Diagrams (80-20 rule):** The Pareto principle, also called the 80-20 rule is based on measuring the frequencies or occurrences that cause most of the problems. The rules state that 80% of the problems are due to 20% of the causes. Alternatively stated, 80% of the system errors can be removed by resolving 20% of the defects.
- **Control charts:** This is used to determine whether the performance of a process is within the expected range demarcated by upper and lower limits. Usually, the upper and lower limits are specified at a three-sigma value (sigma is the standard deviation) on either side of the mean.

Review and Retrospective

- Focus on what went well, what went wrong, and how the team can improve in the next iteration.
- Steps: Set the stage, Gather data, Generate insights, Decide what to do & Closing.
    - SAMOLO - Same as – more or – less of Same as' are those processes and practices that the team finds value and would continue to use. 'More of' are those processes and practices that the team finds value, in but feels it's not doing enough of. 'Less of' are those processes and practices that are team starting to find diminishing value and would like to discourage.
    - **Start doing – stop doing – keep doing**: 'Start doing' are the activities that the team feels are not done, but are worth doing because they will generate better outcomes. 'Stop doing' are the activities that the team feels are not useful, generally disliked, or not worth doing, so should be stopped. 'Keep doing' are the activities that are liked, adding value and the team feels are worth continuing.
- **Lessons learned vs retrospectives**
    - A retrospective meeting is carried out once per iteration and identifies areas for improvement.
    - Lessons Learned meeting is carried out once at the end of the project/phase as the project closure activity and all the lessons learned are to be identified and documented for future references (not as feedback to the project itself).
- **Retrospective Meeting vs Review Meeting**
    - The retrospective meeting is for the development team only with the primary aim of process improvement.
    - The review meeting is for demonstration / getting acceptance of deliverables with management, product owner, and stakeholders.

Pre-mortem / pre-failure analysis

Pre-mortem is logically opposite to retrospectives that are conducted after the iteration. In a pre-mortem, the team members are asked to determine possible reasons for what can go wrong or why the project could fail in the future.

Agile adoption

- **Agile hybrid models:** a model that is more of a hybrid between traditional waterfall and Agile approaches. Suitable for Large multi-year complex programs, Regulatory and compliance projects, etc.
- **Sidky Agile Maturity Index:** SAMI is used to rank the maturity of Agile practices in an organization. There are five levels: Collaborative, Evolutionary, Integrated, Adaptive, and Encompassing.
- **Virginia Satir change model:** organizations and teams go through the change curve when they transition from traditional methodologies to Agile. There could be a dip in performance for a little time, as the teams organize themselves, look to overcome resistance to change, and settle down into a sustainable operating rhythm.

Agile Glossary

Acceptance Test Driven Development (ATDD)

Test-first software development practice in which acceptance criteria for new functionality are created as automated tests. The failing tests are constructed to pass as development proceeds and acceptance criteria are met.

Acceptance Criteria

Those criteria by which a work item (user story) can be judged to have been successfully implemented and tested. A story is 'done' when all criteria pass testing; conversely, a story is not 'done' if any criteria fail to test. Acceptance Criteria are discreet testable features that relate to the Conditions of Satisfaction that describe a higher level of conditions that, when met, deliver business value.

Acceptance Testing

An acceptance test is a formal description of the behavior of a software product, generally expressed as an example or a usage scenario. A number of different notations and approaches have been proposed for such examples or scenarios. In many cases, the aim is that it should be possible to automate the execution of such tests by a software tool, either ad-hoc to the development team or off the shelf.

Agile

A movement for finding better ways of developing software. Scrum and Extreme Programming are two leading examples. Others, such as Kanban or Lean Startup do not define themselves in the Agile tradition but are based on compatible values and principles

Agile Development

Agile development is a conceptual framework and approach to software development based on principles in the Agile Manifesto. The term is an "umbrella" for a number of specific methodologies based on iterative development techniques where requirements and deliverables evolve through collaboration between self-organizing, cross-functional teams. The most popular agile methodologies include extreme programming (XP), Scrum, Crystal, Dynamic Systems Development (DSDM), Lean Development, and Feature Driven Development (FDD).**Agile Manifesto**

A Manifesto for Agile Software Development is a historical document authored in February of 2001 at a ski resort in Utah. The meeting was held to discuss different approaches to lightweight, responsive, adaptable software development. The Manifesto represents their combined best thinking. It comprises two parts: four value statements and twelve principles. Here are the four value statements of the Manifesto: "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

Agile Methods

Some well-known agile software development methods include:

- Agile modeling
- Agile Unified Process (AUP)
- Dynamic Systems Development Method (DSDM)
- Essential Unified Process (EssUP)
- Feature Driven Development (FDD)
- Open Unified Process (Open UP)
- Scrum
- Velocity Tracking

Agile Modeling

Agile Modeling is a practice-based methodology for Modeling and documentation of software-based systems. It is intended to be a collection of values, principles, and practices for Modeling software that can be applied on a software development project in a more flexible manner than traditional Modeling methods.

Antipattern

Anti-patterns are common solutions to common problems where the solution is ineffective and may result in undesired consequences

Application Lifecycle Management (ALM)

Also called ALM, Application Lifecycle Management is the management platform of the entire software application lifecycle, from planning to the final release. Key components of the platform include the ability to handle change management, workflow, source code management, task management, testing and bug tracking, reporting, and analytics.

Automated Build

In the context of software development, build refers to the process that converts files and other assets under the developers' responsibility into a software product in its final or consumable form. The build is automated when these steps are repeatable, require no direct human intervention, and can be performed at any time with no information other than what is stored in the source code control repository.

Backlog

A backlog is an ordered list of items representing everything that may be needed to deliver a specific outcome. There are different types of backlogs depending on the type of item they contain and the approach being used.

Backlog Grooming

Backlog grooming is when the product owner and some, or all, of the rest of the team refine the backlog on a regular basis to ensure the backlog contains the appropriate items, that they are prioritized, and that the items at the top of the backlog are ready for delivery.

Behavior Driven Development (BDD)

BDD is a practice where members of the team discuss the expected behavior of a system in order to build a shared understanding of expected functionality.

Branching

Branching is the duplication of objects under revision control (such as a source code file, or a directory tree) in such a way that the newly created objects initially have the same content as the original, but can evolve independently of the original. Branching can take two forms, static or dynamic. In static branches, copy and label operations are used to duplicate a given branch. The duplicate then can evolve independently. With dynamic branches, usually implemented in streams, only the label operation is used, to flag the point in time that a stream diverged from its parent stream. Both branching forms support some form of merging, so that code changes made on a branch can be re-integrated into another branch, as is typical in parallel development processes.

Burn-down Chart

A chart that shows the amount of work that is thought to remain in a backlog. Time is shown on the horizontal axis and work remains on the vertical axis. As time progresses and items are drawn from the backlog and completed, a plot line showing work remaining may be expected to fall. The amount of work may be assessed in any of several ways such as user story points or task hours. Work remaining in Sprint Backlogs and Product Backlogs may be communicated by means of a burn-down chart

Burn-up Chart

A chart which shows the amount of work which has been completed. Time is shown on the horizontal axis and work is completed on the vertical axis. As time progresses and items are drawn from the backlog and completed, a plot line showing the work done may be expected to rise. The amount of work may be assessed in any of several ways such as user story points or task hours. The amount of work considered to be in-scope may also be plotted as a line; the burn-up can be expected to approach this line as work is completed.

Business Agility

Business agility is the ability of an organization to sense changes internally or externally and respond accordingly in order to deliver value to its customers.

Coherent/Coherence

The quality of the relationship between certain Product Backlog items may make them worthy of consideration as a whole.

Collective Ownership

Collective code ownership is the explicit convention that every team member can make changes to any code file as necessary: either to complete a development task, to repair a defect, or to improve the code's overall structure.

Continuous Deployment

The continuous deployment aims to reduce the time elapsed between writing a line of code and making that code available to users in production. To achieve continuous deployment, the team relies on infrastructure that automates and instruments the various steps leading up to deployment, so that after each integration successfully meets these release criteria, the live application is updated with new code.

Continuous Integration

Continuous Integration is the practice of merging code changes into a shared repository several times a day in order to release a product version at any moment. This requires an integration procedure that is reproducible and automated.

CRC Cards

Class Responsibility Collaborator (CRC) Cards are an object-oriented design technique teams can use to discuss what a class should know and do and what other classes it interacts with.

Cross-functional Team

The team is comprised of members with all functional skills and specialties necessary to complete a project from start to finish.

Customer Development

Customer development is a four-step framework that provides a way to use a scientific approach to validate assumptions about your product and business.

Daily Meeting

Daily time-boxed event of 15 minutes, or less, for the Development Team to re-plan the next day of development work during a Sprint. Updates are reflected in the Sprint Backlog.

Definition of Done

The definition of done is an agreed-upon list of the activities deemed necessary to get a product increment, usually represented by a user story, to a done state by the end of a sprint.

Definition of Ready

The Definition of Ready involves creating clear criteria that a user story must meet before being accepted into an upcoming iteration. This is typically based on the INVEST matrix.

Development Team

The role within a Scrum Team accountable for managing, organizing and doing all development work required to create a releasable Increment of product every Sprint.

DMAIC

The five stages of a cycle of continuous improvement are associated with the "six sigma" approach.

- Define - Identify the stakeholders, the problem, and its scope.
- Measure - Establish the metrics for analyzing the problem and determining the impact of any proposed changes.
- Analyze - Review the collected data, establish
- performance gaps and variations, identify best practices
- Improve - Design and develop a solution, validate and then implement the solution.
- Control - Establish new standards, update measurement systems, and plan to maintain and improve

Emergence

The process of the coming into existence or prominence of new facts or new knowledge of a fact, or knowledge of a fact becoming visible unexpectedly.

Empiricism

Process control type in which only the past is accepted as certain and in which decisions are based on observation, experience, and experimentation. Empiricism has three pillars: transparency, inspection, and adaptation.

Engineering standards

A shared set of development and technology standards that a Development Team applies to create releasable Increments of software.

Epic

An epic is a large user story.

Estimation

In software development, an "estimate" is the evaluation of the effort necessary to carry out a given development task; this is most often expressed in terms of duration.

Exploratory Testing

Exploratory testing is, more than strictly speaking a "practice," a style or approach to testing software that is often contrasted to "scripted testing."

Extreme Programming

Extreme Programming (XP) is an agile software development framework that aims to produce higher-quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

Facilitation

A facilitator is a person who chooses or is given the explicit role of conducting a meeting.

Feature-Driven Development (FDD)

Feature Driven Development (FDD) is an Agile method for developing software based on an iterative and incremental software development process. The main purpose of FDD is to deliver tangible, working software repeatedly in a timely manner.

Forecast (of functionality)

The selection of items from the Product Backlog a Development Team deems feasible for implementation in a Sprint.

Four D's

The four types of tasks that can make up an Agile story.

- Discover - Gather/research information, and define sprint deliverables.
- Develop - Create implementation plans, and construct and test deliverables.
- Deliver - Train employees, execute plans, and release deliverables.
- Debrief - Monitor results, adjust deliverables, and closeout stories.

Frequent Releases

An Agile team frequently releases its product into the hands of end users, listening to feedback, whether critical or appreciative.

Given When Then

The Given-When-Then formula is a template intended to guide the writing of acceptance tests for a User Story: (Given) some context, (When) some action is carried out, (Then) a particular set of observable consequences should obtain.

Heartbeat Retrospective

The team meets regularly to reflect on the most significant events that occurred since the previous such meeting and identify opportunities for improvement.

Increment

A piece of working software that adds to previously created Increments, where the sum of all Increments -as a whole - form a product.

Incremental Development

In an Agile context, Incremental Development is when each successive version of a product is usable, and each builds upon the previous version by adding user-visible functionality.

Information Radiators

"Information radiator" is the term for any of a number of visual displays which a team places in a highly visible location so that all team members can see the latest information at a glance.

Inspect and Adapt

An Agile concept is where teams evaluate a project by looking at the product, listening to each other's feedback and ultimately improving the process or changing course.

Integration

"Integration" (or "integrating") refers to any efforts still required for a project team to deliver a product suitable for release as a functional whole.

INVEST

The acronym INVEST stands for a set of criteria used to assess the quality of a user story. If the story fails to meet one of these criteria, the team may want to reword it.

Iron Triangle

A concept of project management is to visualize a situation where all three project constraints of cost, scope, and time are fixed at the start of the project. As the project progresses, no one constraint may change without a change in at least one of the remaining two.

Iteration

An iteration is a timebox during which development takes place. The duration may vary from project to project and is usually fixed.

Iterative Development

Agile projects are iterative insofar as they intentionally allow for "repeating" software development activities, and for potentially "revisiting" the same work products (the phrase "planned rework" is sometimes used; refactoring is a good example).

Kanban

The Kanban Method is a means to design, manage and improve flow for knowledge work and allows teams to start where they are to drive evolutionary change.

Kanban Board

A Kanban Board is a visual workflow tool consisting of multiple columns. Each column represents a different stage in the workflow process.

Lead Time

Lead Time is the time between a customer order and delivery. In software development, it can also be the time between a requirement made and its fulfillment.

Milestone Retrospective

A Milestone Retrospective is a team's detailed analysis of the project's significant events after a set period of time or at the project's end.

Minimum Marketable Feature (MMF)

A Minimum Marketable Feature is a small, self-contained feature that can be developed quickly and that delivers significant value to the user.

Minimum Viable Product (MVP)

A Minimum Viable Product is the "version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort."

Mob Programming

Mob Programming is a software development approach where the whole team works on the same thing, at the same time, in the same space, and at the same computer.

Mock Objects

Mock Objects (commonly used in the context of crafting automated unit tests) consist of instantiating a test-specific version of a software component.

Niko-Niko Calendar

A Niko-Niko Calendar is updated daily with each team member's mood for that day. Over time the calendar reveals patterns of change in the moods of the team, or of individual members.

Pair Programming

Pair programming consists of two programmers sharing a single workstation (one screen, keyboard and mouse among the pair).

PDCA

The four stages of a cycle of continuous improvement were popularized by W. Edward Deming.

- Plan - Define the problem, and methods to measure it, and obtain management support for future stages.
- Do - Do the tests and prototypes to understand the problem, establish root causes, and investigate alternatives.
- Check - Analyze the results of the "do" stage to determine if a solution effectively resolves the problem while breaking nothing else.
- Act - Fully implement the identified solution.

Personas

Personas are synthetic biographies of fictitious users of the future product.

Planning Poker

An approach to estimation used by Agile teams. Each team member "plays" a card bearing a numerical value corresponding to a point estimation for a user story.

Points (estimates in)

Agile teams generally prefer to express estimates in units other than the time-honored "man-hours." Possibly the most widespread unit is "story points."

Product Backlog

An ordered list of the work to be done in order to create, maintain and sustain a product. Managed by the Product Owner.

Product Backlog refinement

The activity in a Sprint through which the Product Owner and the Development Teams add granularity to the Product Backlog.

Product Owner

The role in Scrum is accountable for maximizing the value of a product, primarily by incrementally managing and expressing business and functional expectations for a product to the Development Team(s).

Project Chartering

A high-level summary of the project's key success factors is displayed on one wall of the team room as a flipchart-sized sheet of paper.

Quick Design Session

When "simple design" choices have far-reaching consequences, two or more developers meet for a quick design session on a whiteboard.

Ready

A shared understanding by the Product Owner and the Development Team regarding the preferred level of description of Product Backlog items introduced at Sprint Planning.

Refactoring

Refactoring consists of improving the internal structure of an existing program's source code, while preserving its external behavior.

Relative Estimation

Relative estimation consists of estimating tasks or user stories by comparison or by the grouping of items of equivalent difficulty.

Role-feature-reason

The "role-feature-reason" template is one of the most commonly recommended aids to write user stories: As a ... I want ... So that ...

Rule of Simplicity

Rules of Simplicity is a set of criteria, in priority order, proposed by Kent Beck to judge whether some source code is "simple enough."

Sashimi

An Agile concept of delivering value in slices rather than in layers/stages. An Agile Story is sashimi because it can be proven it is done. It is not possible to prove that a requirements document is done.

Scrum

A framework to support teams in complex product development. Scrum consists of Scrum Teams and their associated roles, events, artifacts, and rules, as defined in the Scrum GuideTM.

Scrum Board

A physical board to visualize information for and by the Scrum Team, often used to manage Sprint Backlog. Scrum boards are an optional implementation within Scrum to make information visible.

Scrum Guide

The definition of Scrum, written and provided by Ken Schwaber and Jeff Sutherland, co-creators of Scrum. This definition consists of Scrum's roles, events, artifacts, and the rules that bind them together.

Scrum Master

The role within a Scrum Team is accountable for guiding, coaching, teaching, and assisting a Scrum Team and its environments in a proper understanding and use of Scrum.

Scrum of Scrums

A technique to scale Scrum up to large groups (over a dozen people), consisting of dividing the groups into Agile teams of 5-10.

Scrum Team

A self-organizing team consisting of a Product Owner, Development Team and Scrum Master.

Scrum Values

A set of fundamental values and qualities underpinning the Scrum framework; commitment, focus, openness, respect, and courage.

Self-organization

The management principle is that teams autonomously organize their work. Self-organization happens within boundaries and against given goals. Teams choose how best to accomplish their work, rather than being directed by others outside the team.

Sign Up for Tasks

Members of an Agile development team normally choose which tasks to work on, rather than being assigned work by a manager.

Spike

Timeboxed investigation of feasibility via a basic implementation experiment or prototype to test out new, unknown, risky or complex technical solutions. The result of a Spike is to inform future implementation decisions.

Sprint

Time-boxed event of 30 days, or less, that serves as a container for the other Scrum events and activities. Sprints are done consecutively, without intermediate gaps.

Simple Design

A team adopting the "simple design" practice bases its software design strategy on a set of "simple design" principles.

Sprint Backlog

An overview of the development work to realize Sprint's goal, typically a forecast of functionality and the work needed to deliver that functionality. Managed by the Development Team.

Sprint Goal

A short expression of the purpose of a Sprint, is often a business problem that is addressed. Functionality might be adjusted during the Sprint in order to achieve the Sprint Goal.

Sprint Planning

Time-boxed event of 8 hours, or less, to start a Sprint. It serves for the Scrum Team to inspect the work from the Product Backlog that's most valuable to be done next and design that work into the Sprint backlog.

Sprint Retrospective

Time-boxed event of 3 hours, or less, to end a Sprint. It serves for the Scrum Team to inspect the past Sprint and plan for improvements to be enacted during the next Sprint.

Sprint Review

Time-boxed event of 4 hours, or less, to conclude the development work of a Sprint. It serves the Scrum Team and the stakeholders to inspect the Increment of the product resulting from the Sprint, assess the impact of the work performed on overall progress, and update the Product backlog in order to maximize the value of the next period.

Stakeholder

A person external to the Scrum Team with a specific interest in and knowledge of a product that is required for incremental discovery. Represented by the Product Owner and actively engaged with the Scrum Team at Sprint Review.

Scrum of Scrums

A technique to scale Scrum up to large groups (over a dozen people), consisting of dividing the groups into Agile teams of 5-10.

Sign Up for Tasks

Members of an Agile development team normally choose which tasks to work on, rather than being assigned work by a manager.

Simple Design

A team adopting the "simple design" practice bases its software design strategy on a set of "simple design" principles.

Story Mapping

Story mapping consists of ordering user stories along two independent dimensions.

Story Splitting

Splitting consists of breaking up one user story into smaller ones, while preserving the property that each user story separately has measurable business value.

Sustainable Pace

The team aims for a work pace that they would be able to sustain indefinitely.

Task Board

The most basic form of a task board is divided into three columns labeled "To Do," "In Progress," and "Done." Cards are placed in the columns to reflect the current status of that task.

Test Driven Development (TDD)

"Test-driven development" is 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).

Team

A "team" in the Agile sense is a small group of people, assigned to the same project or effort, nearly all of them on a full-time basis.

Team Room

The team (ideally the whole team, including the product owner or domain expert) has the use of a dedicated space for the duration of the project, set apart from other groups' activities.

Three C's

"Card, Conversation, Confirmation" is a formula that captures the components of a User Story.

Three Amigos

Three amigos refer to the primary perspectives to examine an increment of work before, during, and after development. Those perspectives are Business, Development, and Testing.

Three Questions

The daily meeting is structured around some variant of the following three questions: What have you completed? What will you do next? What is getting in your way?

Timebox

A timebox is a previously agreed period of time during which a person or a team works steadily towards the completion of some goal.

Ubiquitous Language

Strive to use the vocabulary of a given business domain, not only in discussions about the requirements for a software product but in discussions of design as well and all the way into "the product's source code itself."

Unit Testing

A unit test is a short program fragment written and maintained by the developers on the product team, which exercises some narrow part of the product's source code and checks the results.

Usability Testing

Usability testing is an empirical, exploratory technique to answer questions such as "how would an end user response to our software under realistic conditions?"

User Stories

In consultation with the customer or product owner, the team divides up the work to be done into functional increments called "user stories."

Values

When the values of Commitment, Courage, Focus, Openness, and Respect are embodied and lived by the Scrum Team, the Scrum pillars of transparency, inspection, and adaptation come to life and build trust for everyone. The Scrum Team members learn and explore those values as they work with the Scrum events, roles, and artifacts.

Velocity

An optional, but often-used, indication of the average amount of Product Backlog turned into an Increment of a product during a Sprint by a Scrum Team, tracked by the Development Team for use within the Scrum Team.

Version Control

Version control is not strictly an Agile "practice" insofar as it is now widespread in the industry as a whole. But it is mentioned here for several reasons.

XP (Extreme Programming)

"Extreme Programming," is one implementation of the Agile methodology that focuses on producing the simplest coding situation for application requirements and includes practices such as pair programming, incremental design, and continuous integration.