Domain I Agile Principles and Mindset

The PMI-ACP Exam consists of 120 questions which can be categorised into seven domains. The first domain: Domain I Agile Principles and Mindset is the knowledge about "how to explore, embrace, and apply agile principles and mindset within the context of the project team and organization" (source: PMI-ACP Examination Content Outline).
Domain I Agile Principles and Mindset accounts for 16% of all questions in the PMI-ACP Exam (i.e. ~19 questions among 120 PMI-ACP Exam questions)

Why Use Agile?

When faced with a new or uncertain process, such as building an underwater home for the first time of using carbon nanotubes instead of steel, there will be many unknowns and uncertainties involved in the risks and solutions required for the new environment or materials. Then agile is useful.

Project Classifications:

Projects can be of two types: ***Industrial Work Projects*** and ***Knowledge Work Projects***. Knowledge workers are not only found in the IT industry; they are also engineers, teachers, scientists, lawyers, doctors, writers, and many others employed today.

Characteristics of Industrial Work Projects	Characteristics of Knowledge Work Projects
Work is visible	Work is invisible
Work is stable	Work is changing
Emphasis is on running things	Emphasis is on changing things
More structure with fewer decisions	Less structure with more decisions
Focus on the right answers	Focus on the right questions
Define the task	Understand the task
Command and control	Give autonomy
Strict standards	Continuous innovation
Focus on quantity	Focus on quality
Measure performance to strict standards	Continuously learn and teach
Minimize cost of workers for a task	Treat workers as assets, not as costs

The Agile Mindset:

1. We **increase return on investment** by making continuous flow of value our focus.
2. We **deliver reliable results** by engaging customers in frequent interactions and shared ownership.
3. We **expect uncertainty** and manage for it through iterations, anticipation, and adaptation.
4. 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.
5. We **boost performance** through group accountability for results and shared responsibility for team effectiveness.
6. We **improve effectiveness and reliability** through situationally specific strategies, processes, and practices.

Core principles of agile:

1. Welcoming change
2. Working in small value-added increments
3. Using build and feedback loops
4. Learning through discovery
5. Value-driven development
6. Failing fast with learning
7. Continuous delivery
8. Continuous improvement

9 Agile Tasks:

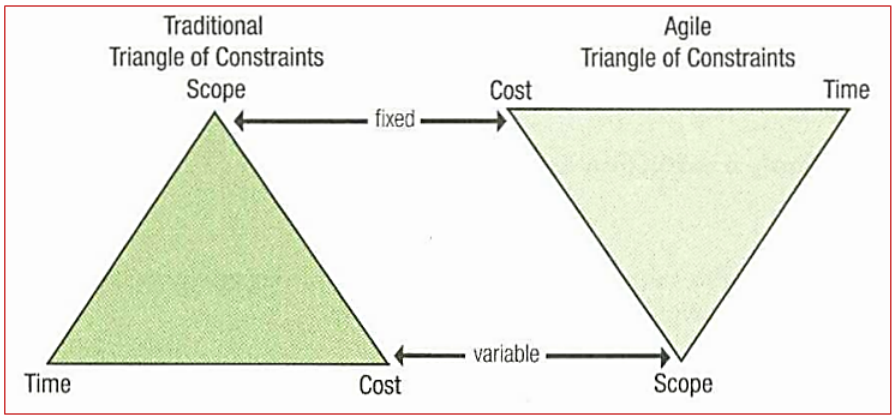
According to the PMI-ACP Exam Content Outline, Domain I Agile Principles and Mindset consists of nine tasks:

1. **Act as an advocate for Agile principles** with customers and the team to ensure a shared Agile mindset
2. Create a **common understanding** of the values and principles of Agile through practising Agile practices and using Agile terminology effectively.
3. **Educate** the organization and **influence** project and organizational processes, behaviours, and people to support the change to Agile project management.
4. Maintain highly visible **information radiators** about the progress of the projects to enhance transparency and trust.
5. Make it **safe** to experiment and make mistakes so that everyone can benefit from empirical learning.
6. Carry out **experiments** as needed to enhance creativity and discover efficient solutions.
7. **Collaborate** with one another to enhance knowledge sharing as well as removing knowledge silos and bottlenecks.
8. Establish a safe and respectful working environment to encourage emergent leadership through **self-organizing and empowerment**.
9. Support and encourage team members to perform their best by being a **servant leader**.

Creating Organizational Change:

1. **Think** - Means individually learning and internalizing agile principles.
2. **Do** - Doing is the practice of agile. For example, this might involve visualizing work items, using short iterations, or building in feedback and improvement steps.
3. **Encourage Others** - This final step involves encouraging others to become agile.

The Agile Triangle:



The Agile Manifesto – 4 Values:

1. **Individuals and interactions** over Processes and tools
2. **Working software** over Comprehensive documentation
3. **Customer collaboration** over Contract negotiation
4. **Responding to change** over Following a plan

The format of the four values – "**A over B**" means addressing intention, focus, and effort more on A rather than B. Agile documentation should be "*barely sufficient*", "*just in time*" and "*just because*".

The Agile Manifesto – 12 Principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. ***(Satisfy customer with great systems)***
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. ***(Welcome change)***
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. ***(Deliver frequently)***
4. Businesspeople and developers must work together daily throughout the project. ***(Work with business)***

5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done. ***(Motivate people)***
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. ***(Face-to-face communication)***
7. Working software is the primary measure of progress. ***(Measure systems done)***
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. ***(Maintain sustainable pace)***
9. Continuous attention to technical excellence and good design enhances agility. ***(Maintain design)***
10. Simplicity—the art of maximizing the amount of work not done—is essential. ***(Keep it simple)***
11. The best architectures, requirements, and designs emerge from self-organizing teams. ***(Team creates architecture)***
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly. ***(Reflect and adjust)***

Agile Project Management Fundamentals:

1. Users Involvement
2. Team Empowerment
3. Fixed Time Box
4. Requirements at a High Level
5. Incremental Project Releases
6. Frequent Delivery
7. Finish Tasks One by One
8. Pareto Principle
9. Testing – Early and Frequent
10. Teamwork

Agile Methodologies:

The following are the common Agile methodologies in practice these days. An understanding of the process and terminologies of these Agile methodologies will help ensure Agile practices to be carried out effectively.

1. Scrum
2. XP (eXtreme Programming)
3. Kanban
4. LSD (Lean Software Development)
5. Crystal Family
6. FDD (Feature Driven Development)
7. ASD (Adaptive Software Development)
8. DSDM (Dynamic Systems Development Method)

Sharing of Knowledge:

1. Ideally, Agile teams are best to be co-located (working within the same room with seats facing each other) to enhance pro-active support, free discussion, open collaboration, and osmotic communication. Face-to-face communication is always encouraged practice. Pair programming if feasible.
2. Make use of daily stand-up, review, and retrospectives.
3. Make use of Agile tooling to enhance sharing of knowledge:
    a. Kanban boards
    b. White boards
    c. Bulletin boards
    d. Burn-down/burn-up charts
    e. Wikis website
    f. Instant messaging – Skype, web conferencing, etc.
    g. Online planning poker
4. Since documentations are not encouraged.

5. Co-located teams can share tacit (unwritten) knowledge more readily.

Information Radiators:

Information radiators are highly visible charts and figures displaying project progress and status, e.g. Kanban boards, burn-down charts. These shows the real progress and performance of the project and team which enhances transparency and trust among team members and other stakeholders.

Self-organization and Empowerment:

1. Self-organizing teams are the foundation for Agile project management. Self-organization includes team formation, work allocation (members are encouraged to take up works beyond their expertise), self-management, self-correction and determining how to work is considered "done". Agile team is given the power to self-direct and self-organize by making and implementing decisions, including work priority, time frames, etc. as they believe "*the best person to make the decision is the one whose hands are actually doing the work*".
2. In Agile projects, the project manager/Coach/ScrumMaster practice servant leadership to remove roadblocks and obstacles and to enable the team to perform best.

Scrum:

Scrum is a popular agile model and framework that is lightweight and easy to understand—but like all agile methods, it is difficult to truly master.

Scrum 3 Pillars and 5 Values:

1. **Transparency:** This involves giving visibility to those responsible for the outcome. An example of transparency would be creating a common definition of what "done" means, to ensure that all the stakeholders are in agreement.
2. **Inspection:** This involves doing timely checks of how well the project is progressing toward its goals, looking for problematic deviations or differences from the goals.
3. **Adaptation:** This involves adjusting the team's process to minimize further issues if an inspection shows a problem or undesirable trend.

In addition to the 3 pillars, Scrum also recognizes 5 fundamental values—**focus**, **courage**, **openness**, **commitment**, and **respect**.

Scrum Process Terminologies:

1. **Sprints:** A sprint is a timeboxed (time-limited) iteration of 1 to 4 weeks of time during which the team builds a potentially releasable product. (Scram term is "***sprint***", and XP term is "***iteration***"). A sprint can be cancelled before the timebox is over if the sprint goal becomes obsolete because of a change in business direction or technology conditions. However, only the product owner can cancel the sprint. When that happens, any incomplete product backlog items are re-estimated and returned to the product backlog
2. **Scrum Team Roles:** Scrum teams are made up of the following roles –
    a. **Development Team** – Self-organizing, empowered, cross-functional and having project goal shared vision
    b. **Product Owner** – maximizing the product value, managing the product backlog
    c. **ScrumMaster** – ensuring that the Scrum methodology is understood and used effectively, removing any impediments, facilitating events (meetings), and coaching the team members, facilitating its adoption of Scrum on a wider scale throughout the organization

Scrum Activities (Events, Ceremonies):

The Scrum methodology defines 5 "activities", which are actually meetings/ceremonies that are focused on a specific purpose. These include ***Product Backlog Refinement***, ***Sprint Planning Meetings*** *(Less than 8 Hours)*, ***Daily Scrums*** *(15 Minutes)*, ***Sprint reviews*** *(2 to 4 Hours)*, and ***Sprint Retrospectives*** *(30 Minutes to 3 Hours)*.

In "***Daily Scrum***" meeting following 3 questions are answered –

1. What have done since the last daily scrum?

2. What do I plan to do today?
3. Are there any impediments to my progress?

In "***Scrum of Scrums***" meeting following 4 questions are answered –

1. What has your team done since we last met?
2. What will your team do before we meet again?
3. Is anything getting in your team's way?
4. Are you about to put something in another team's way?

Larger endeavours may even warrant a "***scrum of scrum of scrums***" where the teams repeat this pattern with a third-level meeting. Here, a representative from each scrum of scrums will attend a "***scrum of scrum of scrums***" to coordinate the work across a larger set of teams.

There are 5 Scrum Ceremonies:

They are ***Product Backlog Refinement***, ***Sprint Planning***, ***Daily Standup*** (***Daily Scrum***), ***Sprint Review***, ***Sprint Retrospective*** -

1. **Sprint Planning:**
   a. Product Owner: Defines the goals and priorities for the upcoming Sprint.
   b. Scrum Master: Facilitates the planning meeting.
   c. Development Team: Collaborates to determine which items from the Product Backlog will be included in the Sprint and creates a plan for achieving the Sprint goal.
2. **Daily Standup (Daily Scrum):**
   a. Product Owner: Usually attends as an observer but does not actively participate.
   b. Scrum Master: Facilitates the daily standup, ensuring it stays focused and timely.
   c. Development Team: Shares updates on progress, discusses obstacles, and plans for the next 24 hours.
3. **Sprint Review:**
   a. Product Owner: Presents the increment to stakeholders and discusses the Product Backlog.
   b. Scrum Master: Facilitates the review and ensures that feedback is collected.
   c. Development Team: Demonstrates the completed work and gathers feedback.
4. **Sprint Retrospective:**
   a. Product Owner: May attend to understand team dynamics and improvement suggestions.
   b. Scrum Master: Facilitates the retrospective, encouraging open communication about the Sprint and identifying areas for improvement.
   c. Development Team: Reflects on the Sprint, discusses what went well, what could be improved, and plans actions for the next Sprint.

It's important to note that while certain ceremonies involve the active participation of specific roles, collaboration and communication among all team members are crucial throughout the Scrum framework.

Scrum Artifacts

There are 3 Scrum Artifacts - **Product Increment**, **Product Backlog**, and the **Sprint Backlog**

Extreme Programming (XP):

Scrum focuses at the project management level on prioritizing work and getting feedback, **XP** focuses on software development best practices.

XP 5 Core Values:

1. **Simplicity:** This value focuses on reducing complexity, extra features, and waste.
2. **Communication:** This value focuses on making sure all the team members know what is expected of them and what other people are working on.
3. **Feedback:** The team should get impressions of suitability early. Failing fast can be useful.
4. **Courage:** It takes courage to allow our work to be entirely visible to others and should have the confidence to make important changes.

5. **Respect:** Respect is essential on XP projects, where people work together as a team, and everyone is accountable for the success or failure of the project.

XP 4 Team Roles:

XP defines team roles differently than Scrum—the XP roles are ***coach***, ***customer***, ***programmer***, ***tester.***

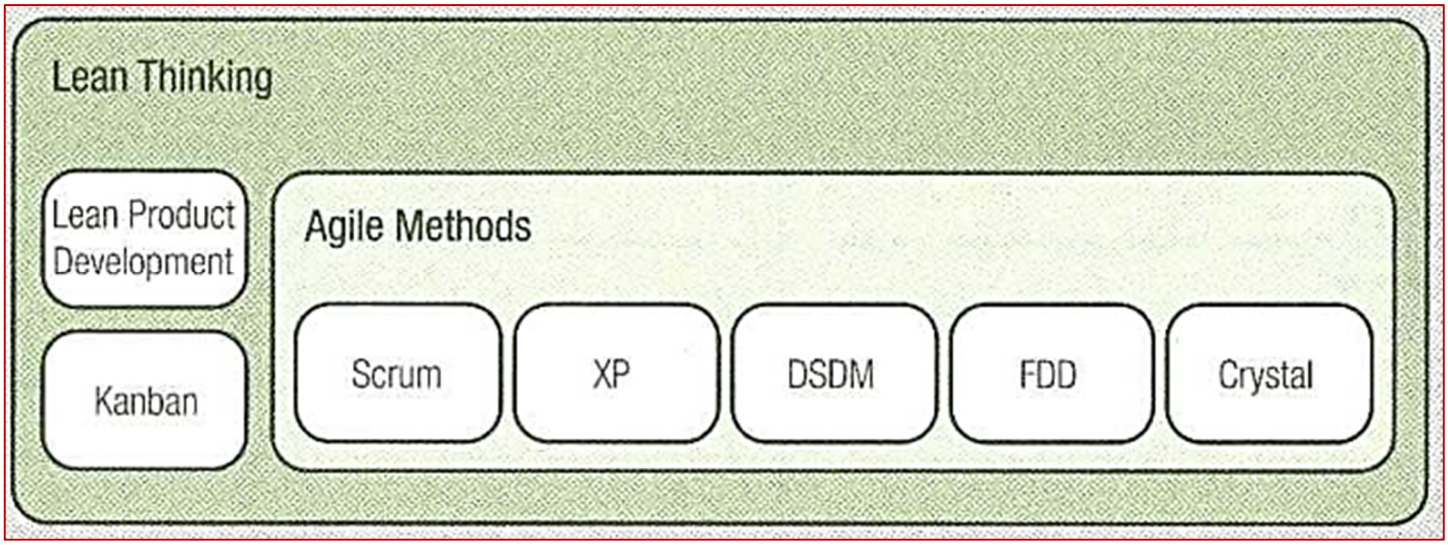
XP 13 Core Practices:

Whole Team: Generalizing specialists, sit together, responsible and committed to their roles.
Planning Games: XP has 2 primary planning activities/games—release planning and iteration planning.
Small Releases: Frequent, small releases to a test environment are encouraged in XP, both at the iteration level, to demonstrate progress and increase visibility to the customer, and at the release level, to rapidly deploy working software to the end users.
Customer Tests: As part of defining the required functionality, the customer describes one or more test criteria that will indicate that the software is working as intended.
Collective Code Ownership: Any pair of developers can improve or amend any code.
Code Standards: XP teams follow a consistent coding standard so that all the code looks as if it has been written by a single, knowledgeable programmer.
Sustainable Pace: XP recognizes that the highest level of productivity is achieved by a team operating at a sustainable pace.
Metaphor: XP uses metaphors and similes to explain designs and create a shared technical vision.
Continuous Integration: Integration involves bringing the code together and making sure it all compiles and works together.
Test-Driven Development: To ensure good test coverage so that problems can be highlighted early in development, XP teams often use the practice of test-driven development.
Refactoring: Improving the design of existing code without altering its external behaviour or adding new functionality, design efficient, removing duplicated code, lowering coupling, and increasing cohesion.
Simple Design: By focusing on keeping the design simple but adequate, XP teams can develop code quickly and adapt it as necessary.
Pair Programming: Production code is written by two developers working as a pair. While one person writes the code, the other developer reviews the code as it is being written—and swap roles frequently.

Lean Product Development:

The high-level principles of lean product development include:

1. Using visual management tools
2. Identifying customer-defined value
3. Building in learning and continuous improvement



Lean 7 Core Concepts:

1. **Eliminate waste:** To maximize value, we must minimize waste.

2. **Empower the team:** Rather than taking a micromanagement approach, we should respect the team members' superior knowledge of the technical steps required on the project and let them make local decisions to be productive and successful.
3. **Deliver fast:** We can maximize the project's return on investment (ROI) by quickly producing valuable deliverables and iterating through designs.
4. **Optimize the whole:** We aim to see the system as more than the sum of its parts. We go beyond the pieces of the project and look for how it aligns with the organization.
5. **Build quality in:** Lean development doesn't try to "test in" quality at the end; instead, we build quality into the product and continually assure quality throughout the development process, using techniques like refactoring, continuous integration, and unit testing.
6. **Defer decisions:** We balance early planning with making decisions and commitments as late as possible. E.g. avoiding being tied to an early technology-bounded solution.
7. **Amplify learning:** This concept involves facilitating communication early and often, getting feedback as soon as possible, and building on what we learn.

The 7 Wastes of Lean:

Waste	Description	Example
Partially done work	Work started, but not complete; partially done work can entropy	Code waiting for testing Specs waiting for development
Extra processes	Extra work that does not add value	Unused documentation Unnecessary approvals
Extra features	Features that are not required, or are thought of as "nice-to-have"	Gold-plating Technology features
Task switching	Multitasking between several different projects when there are context-switching penalties	People assigned to multiple projects
Waiting	Delays waiting for reviews and approvals	Waiting for prototype reviews Waiting for document approvals
Motion	The effort required to communicate or move information or deliverables from one group to another; if teams are not co-located, this effort may need to be greater	Distributed teams Handoffs
Defects	Defective documents or software that needs correction	Requirements defects Software bugs

Categorizing Wastes:

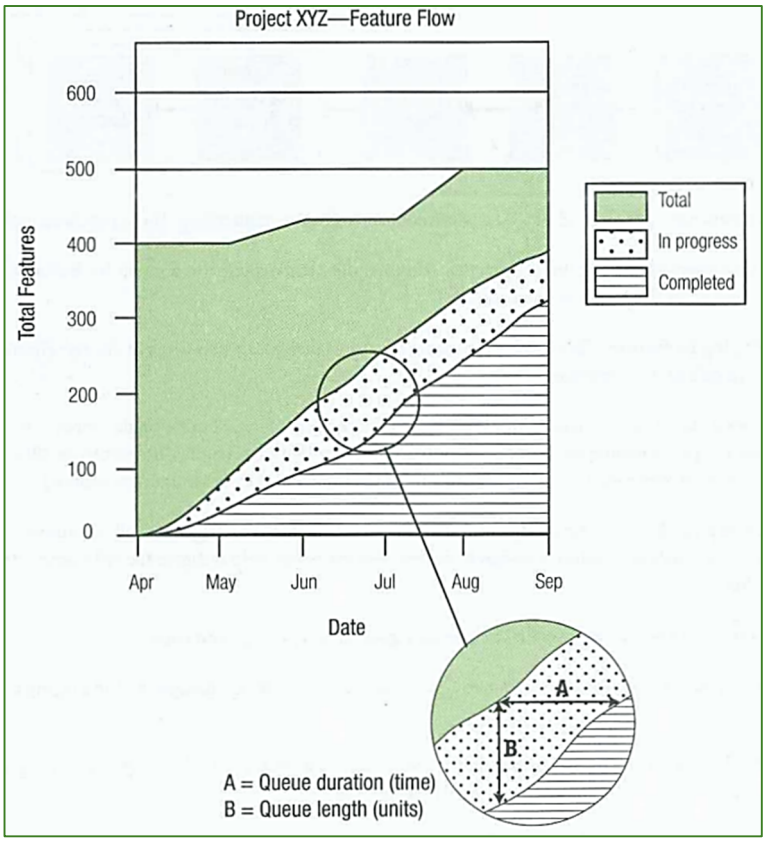
Activity	Type of Waste
Queuing for elevator	Waiting
Rebooting a computer after a program crash	Defects, waiting
Saving documents in old formats for compatibility	Extra processes, motion
Creating notices in French and Spanish to comply with company standards, even though nobody at your location speaks these languages	Extra processes, extra features
Submitting stationery and letterhead orders for approval	Extra processes, motion, partially done work

Kanban:

"***Kanban***" is a Japanese word meaning "***signboard***". The signboard, or Kanban board as it is called, plays an important role in the Kanban methodology.

Kanban 5 Principles:

1. **Visualize the workflow.** Knowledge work projects, by definition, manipulate knowledge, which is intangible and invisible.
2. **Limit WIP (work in Progress).** Restricting the amount of work in progress improves productivity, increases the visibility of issues and bottlenecks and facilitates continuous improvement.
3. **Manage flow.** By tracking the flow of work through a system, issues can be identified, and changes can be measured for effectiveness.
4. **Make process policies explicit.** It is important to clearly explain how things work so the team can have open discussions about improvements in an objective, rather than an emotional or subjective way.
5. **Improve collaboratively.** Through scientific measurement and experimentation, the team should collectively own and improve the processes it uses.



Kanban's Pull System:

Kanban teams employ a "***pull system***" to move work through the development process, rather than planning their work in timeboxed iterations. Each time a Kanban team completes an item of work, it triggers a "**pull**" to bring in the next item they will work on.

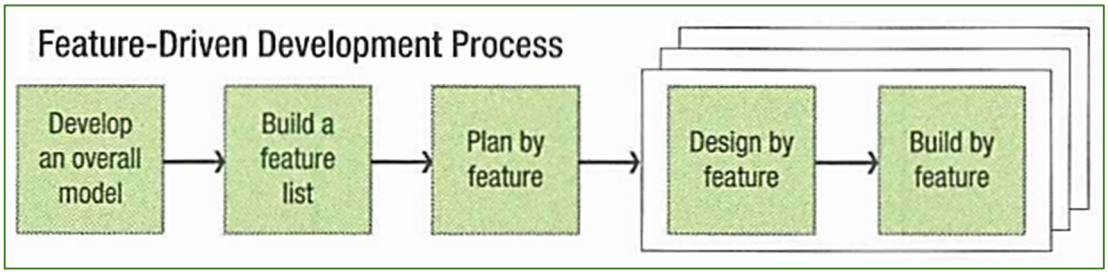
WIP Limits in Kanban:

WIP actually increases a team's productivity—it speeds up the rate at which the work is completed. ***Little's Law*** demonstrates that the duration of a queue (how long it will take to complete the work) is proportional to its size (how much work is in progress).

Feature Driven Development Process:

Feature-Driven Development (FDD) is a simple-to-understand yet powerful approach for building products or solutions. It includes following 8 practices:

1. **Domain object modelling:** In this practice, teams explore and explain the domain (or business environment) of the problem to be solved.
2. **Developing by feature:** This involves breaking functions down into two-week or shorter chunks of work and calling them features.
3. **Individual class (code) ownership:** With this practice, areas of code have a single owner for consistency, performance, and conceptual integrity.
4. **Feature teams:** These are small, dynamically formed teams that vet designs and allow multiple design options to be evaluated before a design is chosen. Feature teams help mitigate the risks associated with individual ownership.
5. **Inspections:** These are the reviews that help ensure good-quality design and code.
6. **Configuration management:** This involves labelling code, tracking changes, and managing the source code.
7. **Regular builds:** Through regular builds, the team makes sure the new code integrates with the existing code. This practice also allows them to easily create a demo.
8. **Visibility of progress and results:** This practice tracks progress based on completed work.



Dynamic Systems Development Method (DSDM):

DSDM was one of the earlier agile methods, and it started out quite prescriptive and detailed. Its coverage of the project life cycle is broad, encompassing aspects of an agile project ranging from feasibility and the business case to implementation. DSDM 8 principles are:

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

DSDM has influenced the development of agile by helping to popularize early architectural considerations, agile suitability filters, and agile contracts.

Crystal:

Crystal isn't just one method; it consists of a family of situationally specific, customized methodologies that are coded by colour names. Each methodology is customized by criticality and team size, which allows Crystal to cover a wide range of projects, from a small team building a low-criticality system (Crystal Clear) to a large team building a high-criticality system (Crystal Magenta).

Crystal Methods

Criticality	Crystal Clear	Crystal Yellow	Crystal Orange	Crystal Red	Crystal Magenta
Life	L6	L20	L40	L100	L200
Essential funds	E6	E20	E40	E100	E200
Discretionary funds	D6	D20	D40	D100	D200
Comfort	C6	C20	C40	C100	C200
	1–6	*7–20*	*21–40*	*41–100*	*101–200*

Team Size

Note: This matrix shows all the potential versions of Crystal—however, the unshaded options aren't recommended because of the project criticality.

Servant Leadership:

Agile promotes a servant leadership model that recognizes it is the team members, not the leader, coach, or ScrumMaster, who get the technical work done and achieve the business value. The servant leadership approach redefines the leader's role in relation to the team.

1. **Shield the team from interruptions.** Servant leaders need to isolate and protect the team members from diversions, interruptions, and requests for work that aren't part of the project.
2. **Remove impediments to progress.** Servant leaders need to clear obstacles from the team's path that would cause delay or waste.
3. **Communicate (and re-communicate) the project vision.** This may seem like an odd duty to place in the category of servant leadership, but communicating and re-communicating the project vision is critical to successfully leading a team.

4. **Carry food and water.** This duty isn't literally about food and water; it's about providing the essential resources a team needs to keep them nourished and productive.

12 Principles for Leading Agile Projects:

1. Learn the team members' needs.
2. Learn the project's requirements.
3. Act for the simultaneous welfare of the team and the project.
4. Create an environment of functional accountability.
5. Have a vision of the completed project.
6. Use the project vision to drive your own behaviour.
7. Serve as the central figure in successful project team development.
8. Recognize team conflict as a positive step.
9. Manage with an eye toward ethics.
10. Remember that ethics is not an afterthought, but an integral part of our thinking.
11. Take time to reflect on the project.
12. Develop the trick of thinking backwards.

4 Agile Leadership Practices:

1. Honesty.
2. Forward-looking.
3. Competent.
4. Inspiring.

5 Leadership Tasks:

1. Practice Transparency through Visualization.
2. Create a Safe Environment for Experimentation.
3. Experiment with New Techniques and Processes.
4. Share Knowledge through Collaboration.
5. Encourage Emergent Leadership via a Safe Environment.

Agile Experimentations:

1. Agile projects make use of empirical process control for project decisions, ongoing observation and experimentation are carried out during project execution to help and influence planning.
2. Introduce spike (including architecture spike) to carry out a technical investigation to reduce risks by failing fast.