Agile Mindset Reminders for Traditional Project Managers

If you come from a traditional project management environment, it may be useful to review the following reminders of the agile mindset before taking the exam. (These are just tips to jog your memory; to do well on the exam, you'll need an in-depth understanding of agile mindset, as explained in the book.) Since a short list like this can only cover selected highlights, feel free to add more tips you want to remember at the end of this list.

The Agile Approach

1. Agile isn't a project management method or a specific set of tools; it is a mindset and approach to working on projects that ideally should be understood by all participants in the project.

2. Agile practices were designed to deal with the fast-moving, complex environment of knowledge work projects (although they can also offer benefits on other kinds of projects).

3. The agile value proposition is that agile practices bring improved visibility, improved adaptability, more value, and less risk to projects.

4. To use agile approaches effectively, you need to first understand the Agile Manifesto values and principles and use them to guide your adoption of agile practices.

5. The Agile Manifesto values focus our attention on certain elements of a project OVER others. This doesn't mean that the less-valued elements aren't necessary or important on an agile project.

6. Agile approaches are able to deliver earlier return on investment than traditional projects because they use incremental releases to start delivering valuable functionality earlier.

7. The most common and influential agile methodologies are Scrum, XP, Kanban, and Lean.

8. Agile projects require more collaboration and trust between stakeholders than traditional projects.

9. Agile projects typically fix time and cost and vary scope, as shown by agile's inverted triangle of constraints. One way they do this is by using timeboxes for most activities (even meetings).

Teams and Team Roles

1. Agile projects have a small, dedicated delivery/development team (12 or fewer people) who build and test the product or solution. If a project requires more people to complete, they are divided into small teams that coordinate their work.

2. An agile delivery team stays together between projects. In other words, projects are brought to an existing team; a new team isn't created for each new project.

3. New or altered teams progress through a tumultuous Storming stage as they learn to work together. Whenever any changes are made to the team, it is likely to throw them back into Storming.

4. Keeping the team intact during and between projects allows the team members to build their relationships with each other and maintain a high level of performance.

5. Even on a high-performing team, divergence (disagreement) is important since it helps team members reach convergence (agreement) on the best decisions and solutions.

6. The members of an agile delivery team have all the skills (such as design, analysis, graphics, coding, testing, etc.) needed to build a complete increment of the product or solution in each iteration.

7. Agile team members are generalizing specialists—they can perform different tasks as needed. Each team member is a "king of many trades and master of each."

8. The delivery team is empowered to be self-organizing and self-directing; they make their own commitments and local decisions about building the product or solution.

9. An agile project's "whole team" includes the delivery team as well as the team coach/ScrumMaster and product owner/customer/business representative/value management team.

10. The team coach/ScrumMaster works within the team but also watches for and removes barriers and impediments that arise outside the team.

11. The product owner/customer/business representative interfaces with the delivery team on a daily basis.

12. The product owner is responsible for prioritizing the product features based on business value, which includes providing an updated, prioritized product backlog for each planning session.

13. The product owner communicates the project requirements to the team, as well as any external deadlines. In other words, they decide WHAT will be built.

14. The team decides HOW the solution will be built; they also determine how many of the product owner's top-priority features can be completed within the next timebox.

15. The team works in short iterations (1–4 weeks), using an empirical development process to converge on the best solution.

Working Collaboratively

1. The whole team is aligned with a common vision, as well as common goals, values, success measures, and working agreements.

2. The whole team uses exercises such as "design the product box" to make the vision visible, explicit, and shared by everyone on the project.

3. The product owner and the members of the delivery team collaborate to create a shared definition of "done" for the project and for each feature, user story, and task.

4. The project delivers working increments of the product early and frequently.

5. The team identifies the project risks and works to minimize them as early as possible.

6. Agile leadership is based on servant leadership, respect, and empowering the team members rather than command-and-control-style management.

7. Although co-location and face-to-face communication are preferred, distributed teams can use digital tools such as videoconferencing to mimic the effects of co-location (i.e., virtual co-location).

8. Agile teams rely on information radiators to share information with stakeholders; these low-tech, high-touch tools might include storyboards, roadmaps, defect tracking, and velocity charts.

9. Burndown and burnup charts (burn charts) are used to track the team's velocity and monitor the project's risk profile. Velocity trends are used to make estimates and forecast completion dates.

Planning and Estimating

1. The team knows upfront that their initial plans are imperfect and will be subject to change, so they progressively elaborate their plans throughout the project rather than only planning upfront.

2. Each time, planning is done to the level of "barely sufficient" for that point in the project. The estimates are progressively refined over time to become more accurate based on the team's evolving velocity and knowledge of the project.

3. Estimates are typically made in story points, which are relative rather than absolute units of measure.

4. High-level estimates are stated in ranges that reflect their degree of uncertainty for stakeholders. (This doesn't apply to the estimates used for scheduling tasks, which are the smallest unit of work.)

5. Sizing and estimating is done by the team members, using tools such as T-shirt sizing and planning poker.

6. In the first few iterations, the team may add a buffer to their estimates to cover unknown contingencies; this buffer is eliminated as more work is done and the team's velocity stabilizes.

User Stories

1. Project requirements are recorded as user stories (i.e., stories written from the standpoint of the user or customer). These stories are compiled in a backlog, or master list of work.

2. A user story is a chunk of functionality within a feature that involves roughly 4 to 40 hours of work. Each user story consists of a card, a conversation, and a confirmation.

   – The story **cards** may be written by the delivery team, but often the whole team gathers to compose them in a user story workshop. They are typically written the format "As a <*Role*>, I want <*Functionality*> to get <*Business benefit*>."

   – The acceptance criteria the team will use to test the story are defined during a **conversation** between the team and the product owner during iteration planning.

   – The **confirmation** is the customer's final acceptance of the story as done.

3. User stories should be written so that they are Independent, Negotiable, Valuable, Estimatable, Small, and Testable (INVEST).

4. During release planning, large user stories are broken down (sliced) into chunks of work that can be completed in one iteration.

5. Even if a user story passes all the agreed-upon acceptance tests, only the product owner can confirm that the functionality is "done."

6. At the end of the iteration, the team demonstrates the product increment they have built for the product owner to try out. In that demo, the product owner may uncover changes or request new functionality. These requests are recorded as user stories and added to the backlog based on value.

Exam Tips

Here are some more tips about the exam itself.

1. The exam questions will approach the topics being tested indirectly through brief scenario questions rather than testing specific facts or information that can be memorized.

2. Most of the questions on the exam will focus on the agile mindset rather than the details of agile tools or techniques. However, you still need to understand agile tools and techniques to do well, since they will be included in the scenarios and listed as answer options.

3. Roughly equivalent terms from different methodologies (such as "sprint" and "iteration" or "product owner" and "customer") will be used interchangeably on the exam.

4. Unless a question indicates otherwise, assume that it refers to an internal project. (Remember this if your firm is a contractor or you have external customers or product owners on your own projects.)

5. Unlike the PMP exam, the PMI-ACP exam doesn't test your understanding of a specific project role; instead, it focuses on agile theory and practice, transcending any particular role. So approach each scenario from the perspective of the agile mindset, rather than as a project manager. (The correct answer might not be the "right" one from a PMP standpoint.)

6. Some scenarios will involve activities done by delivery team members, while others will involve activities done by the ScrumMaster or product owner—and the role involved might not be explicitly stated, since you are expected to know who does what on an agile project. To figure that out, draw upon your understanding of the agile mindset and practices. For example, if a scenario refers to using planning poker to estimate stories, recall that agile estimating is done by the development team, and then answer from their perspective, not that of the product owner or ScrumMaster.