Domain V Adaptive Planning

The PMI-ACP Exam consists of 120 questions which can be categorised into seven domains. The fifth domain: Domain V Adaptive Planning is the knowledge about "Produce and maintain an evolving plan, from initiation to closure, based on goals, values, risks, constraints, stakeholder feedback, and review findings." (source: PMI-ACP Examination Content Outline).
Domain V Adaptive Planning accounts for 12% of all questions in the PMI-ACP Exam (i.e. ~14 questions among 120 PMI-ACP Exam questions)

According to the PMI-ACP Exam Content Outline, Domain V Adaptive Planning consists of 10 tasks grouped within 3 sub-domains:

- **Levels of Planning:**
  1. Planning of Agile projects occurs at **multiple levels**: strategic, release, iteration, daily using **rolling wave planning** and **progressive elaboration** to allow flexibility and adaptability.
  2. Encourage key **stakeholders' participation** in planning and ensure clear communication of planning results to improvement commitment and reduce risks.
  3. **Manage stakeholder expectations** by communicating appropriately detailed information to create a common understanding of the deliverables.
- **Adaptation**
  1. Perform periodic **retrospectives** to allow adaptation of the planning process to maximize value creation.
  2. **Adapt the project plan** continually to reflect changes in project requirements, priorities, stakeholder feedback and environmental factors.
- **Agile Sizing and Estimation**
  1. Make use of **progressive elaboration** to estimate project efforts more accurately.
  2. Update the **team capacity** to factor in maintenance and operations demands.
  3. At the very beginning of the project, make **initial rough estimate ranges** on scope, schedule and cost based on the high-level requirements to kick off the project.
  4. The initial estimates must be refined to provide more accurate figures based latest understanding of the project.
  5. As the project (team velocity, scope, etc.) changes, the **estimates must be updated continually**.

PMI-ACP Study Notes: Domain V Adaptive Planning:

Below is a collection of the key knowledge addressed in Domain V Adaptive Planning and the 10 tasks related to the domain:

Agile Planning:

- Traditional project management triangle: "Time, Cost **and Functionality (Scope)**"
- Agile project management inverted triangle model: "Resources, Time and Functionality (Scope)"
- Core Agile project management phases:
  1. **envisioning**
  2. **speculating**
  3. **exploring**
  4. **adapting**
  5. **closing**
- **Agile Planning** (using Deming Cycle: Plan-Do-Check-Act)
  - Agile projects must be planned at multiple levels (strategic, release, iteration, daily) while ensuring stakeholder engagements
  - though Agile project are not plan-driven, planning is an important activity:
    - initial planning (usually with less planning upfront) - strike a balance of balancing risks and planning investments

- re-planning and midcourse adjustments with gained knowledge from the execution of the project
- carry out only just-in-time planning as the project is ever evolving, making use of
  - ✓ rolling wave planning - break down the planning into stages, with the near-time planning to be carried out first
  - ✓ progressive elaboration -knowledge about the product and requirements will be clearer as the time goes by, those will be revisited and refined continually

- **Agile Planning Stages**
  1. **Product Vision** – a document created by product owner describing what the product is, who will and why use it, and how the product supports company strategy
  2. **Product Roadmap** – a document created by 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**
     - may be in the form of story maps – a diagram indicating the sequences of **backbone**, **walking skeleton** and optional features to be released over time
  3. **Release Plan** – a document created by product owner describing the high-level timeline for product releases (features with higher values are given higher priority in the releases)
     - release planning is the meeting to plan the project schedule at a strategic level for delivering an increment of product value to the customers (can be date driven or feature driven)
  4. **Sprint Plan / Iteration Plan** – a document created by **product owner**, **scrum master and development team** describing sprint goals, tasks and requirements and how those tasks will be completed
     - **iteration 0** 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 a time used to test and prepare the launch software
     - Iteration Planning vs Release Planning
       - ✓ Iteration planning involves schedule development at lower/detailed level about tasks and time
       - ✓ Release planning involves schedule development at high level about features and iterations
  5. **Daily Stand-up / Daily Scrum** – a planning meeting to be attended by project team and stakeholders (as observers only) to discuss on the following 3 questions:
     1. What was completed yesterday?
     2. What will be done today?
     3. Any roadblocks/impediments found
  6. **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
  7. **Sprint Retrospective** – 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
       - ✓ Retrospective meeting is for the development team only (stakeholders are not invited) with the primary purpose of process improvement
       - ✓ Review meeting is for 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
       - ✓ 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 / phrase 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 Planning Artifacts and Meetings**
  - The **Product Vision Statement** is developed by the Product Owner (with the help of the team) to state clearly the purpose and value of the product The Product Roadmap contains the about the schedule and cost milestones which acts as an overview of the planned releases of the project, the Product Roadmap will **change** over time
  - **Personas –** a tool used in requirement collection and testing in which realistic depiction of likely users for the product are created, these users can be real or fictitious
  - **Extreme Persona –** extreme persona is persona taken to the extreme (with extreme characters / requirements) in order to identify user stories which 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
  - **Release Plan –** responsible by the customer / Product Owner with the help of project team in release planning to document the availability of features over time, subject to **changes** depending on actual progress / change requests
- **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 focus of the team, e.g. bug fix, 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 , I want so that**"
    - to describe the requirements in real world scenarios
    - often written on **Story Cards**
    - User stories need to be Independent, Negotiable (can be discussed on implementation), Valuable, Estimable (with adequate info for meaningful estimation), Small, Testable Card, Conversation, Confirmation
  - **Story Maps** – are overview of how different user stories are related to each other in the project
  - **Features** – a capabilities / group of functionalities that is of **value** to the end user
  - **Tasks** – the underlying jobs / development work to fulfil 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 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 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 Modeling**
  - **Agile Modeling**
    - Agile Modeling a **practice-based** methodology (including a collection of **values, principles, and practice**) for effective **modelling 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 modelling methods to be used in traditional project management in order to fit the fast-changing environments of Agile projects
  - also refers to the various **modelling techniques** that are commonly used on Agile projects
    - Agile models are often lightweight (often hand sketched without being polished), easy to change and barely sufficient, e.g. use case diagrams, data models, screen designs

Agile Adaption:

- Make changes to the project, product and processes to deliver best customers value and the special circumstances of the project environment

- may involve process tailoring, continuous integration, adaptive leadership, soft skills negotiations, delivering business value, revised vendor management, change management

- **Process Tailoring**
  - Agile framework or methodologies are not intended to be "one-size-fit-all"
  - the Agile methodology and processes can be altered according to different projects (e.g. in terms of team size, nature, resources, criticality, etc.)
    - the adaptation / process tailoring can be raised in the iteration retrospective to be carried out in the next iteration
  - Note: Kanban is very tailoring-friendly while Scrum / XP do not recommend tailoring
  - However, in the beginning of any projects, it is generally recommended **to implement the Agile methodologies as it** is for the first few iterations for assessment of the suitability before changes / process tailoring are introduced
    - to have better understanding of the values of standard Agile methods and the relationship between different processes of Agile methodologies as some processes are mutually dependent
  - It is recommended to follow the **Shu-Ha-Ri model** (by Alistar Cockburn) if you would like to make changes – Shu-Ha-Ri originates from masters of Japanese Noh theater
    1. **Shu** - Obeying the rules
    2. **Ha** - Consciously moving away from the rules
    3. **Ri** - Unconsciously finding an individual path

- **Agile Adaptation Terms**
  - **Velocity** – a unit to measure the speed of the team (e.g. the **number of story points in each iteration**) which is very useful for planning future releases
    - partially finished tasks in an iteration will NOT be counted towards velocity
    - e.g. if a task with 100 story points is 90% complete, it will contribute0 story point to the iteration
  - **Cycle time** – the amount of time for a feature from start (entering into the product backlog) to finish (done), the shorter the cycle time, the better Burn rate – a measure of how fast money is burnt / the amount of cost estimated over a given period of time (e.g. $1000 per day)
  - **Escaped defects** – defects that are not discovered by the team but by the customer, escaped defects are a measure of the effectiveness of testing and quality control measures
    - i.e. if the escaped defects increase, root cause analyses such as five WHYs or fishbone diagram should be carried out with a view to reduce escaped defects
  - **Agile smells** – Agile term for "symptoms of a problem", they are signs that the Agile project is not running properly, and problems are in the making
  - **Verification** – ensures functionality meets requirements as described in the requirement documentation
  - **Validation** – ensures the deliverable works as intended
  - **Refactoring** – a technique used in programming project to review codes, re-organize and simplify the code without changing the behaviour for easier maintenance
  - **Kaizen** – a Japanese management philosophy in which the project and processes are being checked and improved continuous for better value delivery
- **Agile Project Artifacts**
  - **Product backlog** – the product backlog in an Agile project is a prioritized listing of all features/user stories for the project, the priority is usually based on the **perceived value** of the customer
    - the product backlog is to be created and updated by the **customer**
    - **grooming** – add items based on new user stories and delete old items by considering the relative values of all tasks
    - should be "Detailed appropriately, Estimable, Emergent, Prioritized (DEEP)"
    - **Risk-adjusted backlog** – the product backlog would be re-prioritized with the help of risk analysis input from the team and stakeholders to balance the "risk vs value" factor (as risks are "de-value")

        - ✓ risks are usually considered to be possible negative impacts (though there are many possible positive impacts)
    - **Iteration backlog** – responsible by the **team**, the iteration backlog contains tasks for the iteration in which tasks are allocated through self-organization (team members select the tasks they are most interested in) **Burn-down charts** – a chart showing tasks remaining (in story points, etc.) over the project life
    - **Burn-up charts** – a chart showing tasks completed (in story points, etc.) over the project life
        - **preferred to burn-down charts** as scope changes are clearly visible in burn-up charts
    - **Kanban boards** – a task board showing the progress of tasks through the project processes, can be tailor-made to suit individual projects
        - a **WIP limit** (work-in-progress) would be enforced to ensure efficiency
        - WIP means "Work In Progress" – a work that has been started but not yet reach "done"
        - WIP can also be considered to be the size of the job queue
    - **Cumulative flow diagrams** – a chart showing the state of all tasks through the project processes
        - a **widening band** indicates "Agile smells" and would need investigation to tackle the issues of a particular process
    - **Little's Law** – cycle time is proportional to the size of queue (WIP)
        - cycle time – the time from the very beginning of a task to reaching done status
        - larger WIP would mean longer cycle time
        - a way to improve efficiency is to limit WIP

Agile Sizing and Estimation:

- 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 estimation of Agile tasks: ideal time is a block of 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 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 members 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
- The accuracy of estimation will improve over time in the form of "**Cone of Uncertainty**" (25%-400% from the very beginning to within several percentage near complete) as more knowledge of the project is gained at a later stage.