

Edward Chung, PMP, PMI-ACP

# PMI-ACP Study Notes



Covering the latest PMI-ACP  
Exam Syllabus

*© 2017 Edward Chung*

# Table of Contents

1. Introduction

2. Domain I Agile Principles and Mindset

3. Domain II Value-Driven Delivery

4. Domain III Stakeholder Engagement

5. Domain IV Team Performance

6. Domain V Adaptive Planning

7. Domain VI Problem Detection and Resolution

8. Domain VII Continuous Improvement (Product, Process, People)

9. More PMI-ACP Free Resources

# Introduction

*I am indebted to many PMI-ACP holders who helped me a lot during my certification journey. This free study notes and sharing on my blog is my contribution to the Agile project management community.*

Hi, I am Edward Chung. I passed the PMI-ACP exam with proficient in all domains ("Proficient" is the best grade given by PMI indicating the domain knowledge is above the average level). I am also PMP certified.

In this book, I have included my PMI-ACP exam study notes which has been updated to align with the most recent PMI-ACP exam syllabus (for those taking the exam after July 2015).

And if you need additional tips and resources on your PMI-ACP exam prep, application and exam taking, I have documented my PMI-ACP Certification Exam process in much details on my personal blog at [edward-designer.com](http://edward-designer.com). It is hoped that my blog will be able to remove some roadblocks for fellow PMI-ACP aspirants in the quest of the PMI-ACP Certification (as 'correct' information on PMI-ACP is difficult to find on the internet).

**Wish you PMI-ACP success!**

A handwritten signature in black ink that reads "Edward Chung". The signature is fluid and cursive, with the first letters of the first and last names being capitalized and prominent.

Edward Chung, PMP, PMI-ACP

# 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)*

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, behaviors 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-organization and empowerment**.
9. Support and encourage team members to perform their best by being a **servant leader**.

## PMI-ACP Study Notes: Domain I Agile Principles and Mindset

Below is a collection of the key knowledge addressed in Domain I Agile Principles and Mindset and the nine tasks related to the domain:

- **Agile Manifesto and 12 Agile Manifesto Principles**
  - 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 Project Management Fundamentals**
  - Users Involvement
  - Team Empowerment
  - Fixed Time Box
  - Requirements at a High Level
  - Incremental Project Releases
  - Frequent Delivery
  - Finish Tasks One by One
  - Pareto Principle
  - Testing – Early and Frequent
  - Teamwork
- **Agile Methodologies**
  - The following are the common Agile methodologies in practice these days, these are listed in order of importance for the PMI-ACP Exam. An understanding of the process and terminologies of these Agile methodologies will help ensure Agile practices to be carried out effectively.

- Scrum
  - XP (eXtreme Programming)
  - Kanban
  - LSD (Lean Software Development)
  - Crystal Family
  - FDD (Feature Driven Development)
  - ASD (Adaptive Software Development)
  - DSDM (Dynamic Systems Development Method) Atern
- **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.
- **Agile Experimentations**
    - 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
    - Introduce spike (including architecture spike) to carry out a technical investigation to reduce risks by failing fast
- **Sharing of Knowledge**
    - 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
    - Make use of daily stand-up, review and retrospectives
    - Make use of Agile tooling to enhance sharing of knowledge:
      - Kanban boards
      - white boards
      - bulletin boards
      - burn-down/burn-up charts
      - wikis website
      - instant messaging – Skype, web conferencing, etc.
      - online planning poker
    - Since documentations are not encouraged, co-located teams can share tacit knowledge more readily

- **Self-organization and Empowerment**

- 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"
- In Agile projects, the project manager/Coach/ScrumMaster practice servant leadership to remove roadblocks and obstacles and to enable the team to perform best



# Domain II Value-Driven Delivery

The PMI-ACP Exam consists of 120 questions which can be categorised into seven domain. The second domain: **Domain II Value-Driven Delivery** is the knowledge about "delivering valuable results by producing high-value increments for review, early and often, based on stakeholder priorities and collecting feedback from the stakeholders on these increments, and using this feedback to prioritize and improve future increments" (*source: PMI-ACP Examination Content Outline*).

*Domain II Value-Driven Delivery accounts for **20%** of all questions in the PMI-ACP Exam (i.e. **~24 questions** among 120 PMI-ACP Exam questions)*

According to the PMI-ACP Exam Content Outline, Domain II Value-Driven Delivery consists of 14 tasks grouped within 4 sub-domains:

## Define Positive Value

1. Deliver work **incrementally** to gain competitive advantage and early realization of value.
2. **Maximize value** delivered to stakeholders while at the same time **minimize non-value added work**
3. Reach consensus on the **acceptance criteria** of the deliverables.
4. **Refine project processes** based on factors like team experience and organization preferences.

## Avoid Potential Downsides (Control Risk)

1. Make use of the concept of **minimally marketable features (MMF)**



/ **minimally viable products (MVP)** to deliver releasable increments fast.

2. Solicit **feedback** from stakeholders and review frequently to enhance value.
3. **Fail fast** by carrying out experiments / spikes early on to reduce risk.

## Prioritization

1. **Collaborate with stakeholders** to prioritize features in order to realize value early on.
2. **Review the backlog prioritization** with stakeholder frequently to optimize value delivery.
3. Identify and prioritize **continuously** the various changing factors affecting the project in order to enhance quality and increase value.
4. Both value producing and risk reducing work are prioritized in into the backlog in order to **balance value and risks** (non-value).
5. Non-functional requirements (e.g. security, operations) will need to be considered and prioritized in order to minimize the likelihood of failure.

## Incremental Development

1. **Develop the product incrementally** to reduce risk and deliver value fast.
2. Inspections, testings and **reviews with stakeholders** are carried out periodically to obtain feedback and make corrections as necessary.
3. Retrospectives allow overall **improvements** to be made to the project process.

# PMI-ACP Study Notes: Domain II Value-Driven Delivery

Below is a collection of the key knowledge addressed in Domain II Value-Driven


Delivery and the 14 tasks related to the domain:

- **Value-driven delivery** is an overarching principle for Agile projects. Projects are carried out to realize values (e.g. economic benefits, competitive advantages, reducing risks, regulatory compliance, etc.)
- In terms of Agile project management (and the PMI-ACP exam), prioritization is the process where customers **organize / select** product backlog / user stories for implementation based on the perceived **values**
- **Value-based Prioritization** is to organize things so that the most important ones that deliver values are to be dealt with first
- Return on investment (ROI) / Net present value (NPV) / Internal rate of return (IRR)
  - are metrics to assess prioritization based on **monetary values**
    - **return on investment (ROI)** – the values a project realized (using **present value**) compared to the investment; a positive ROI means the project is profitable
    - **net present value (NPV)** – the net future cash flow (**profit – expenditure**) in terms of today's value (adjusted for future inflation, etc.); a positive NPV means the project is profitable
    - **internal rate of return (IRR)** – this is somewhat like the interest rate of the investment; the higher the positive IRR, the more profitable the project
- **Customer-valued Prioritization**
  - deliver the **highest value** to the customers as **early** as possible
  - the backlog should be customer-valued prioritized while taking into accounts technical feasibilities, risks, dependencies, etc. in order to win customer support
- **Value Prioritization Schemes**
  - simple schemes – rank from high to low (priority 1, 2, 3, ...)
  - **MoSCoW** prioritization scheme – **M**ust have, **S**hould have, **C**ould have, **W**ould like to have, in future
  - **Monopoly money** – ask customers to give out (fake) money to individual business features in order to compare the relative priority
  - **100-Point method** – customers are allowed to give, in total 100 points, to various features
  - **Dot voting / Multi-voting** – everyone is given a limited number of dots (~20% of the number of all options) to vote on the options
  - **Kano analysis** – plot the features on a graph with axes as Need Fulfilled /

Not fulfilled vs Satisfied / Dissatisfied, each feature will then be classified as "exciters, satisfiers, dissatisfiers, indifferent". **Exciters** are of highest values.

- **Requirements Prioritization model** – rate each feature by benefits for having, penalty for not having, cost of producing, risks, etc. and calculate a score using a pre-defined weighted formula
- **CARVER** (Criticality, Accessibility, Return, Vulnerability, Effect, and Recognizability) relative to the objective and mission of the project
  - **Criticality** – how important to be done upfront
  - **Accessibility** – can work on it immediately? or depends on other work / skills?
  - **Return** – ROI / NPV / IRR
  - **Vulnerability** – how easy to achieve the desired results?
  - **Effect** – what are the effects on the project (help moving towards the goal of the project)?
  - **Recognizability** – have the goals been clearly identified?
- **Relative Prioritization / Ranking**
  - an ordered list of all user stories / features to be completed with 1 being the highest priority
  - when new features are to be added, it has to be compared, in terms of priority, to all current features
  - the schemes listed above can be used to assist the relative prioritization / ranking tasks
- **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 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 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.
- **Compliance**

- conforming to a rule, such as a specification, policy, standard or law (e.g. regulatory compliance)
  - compliance usually requires documentation, which is somewhat against the principles of agile (working software over documentation)
  - a balance has to be struck (maybe with the help of Agile compliance management systems)
- **Requirements Review**
    - a group of users and/or recognized experts carrying out reviews on the documented requirements
    - to ensure that:
      - requirements accurately reflect the needs and priorities of stakeholders
      - requirements are technical feasible to be fulfilled
- **Earned Value Management (EVM) for Agile Projects**
    - Earned Value Management is a tool used in **traditional project management** to measure the progress (in terms of realization of values) of the project
    - In Agile, EVM is the measure of **cost performance** of the Agile project
      - though the estimate and actual costs (money spent) can be plotted as a S-curve graph to clearly show whether the project is under- or over-budget, it does not tell whether the progress of the project is ahead of or behind schedule
    - Value (story points and money) is calculated at the end of each iteration for work done
    - For the construction of the **graph for EVM**:
      - The baseline for comparison:
        - number of planned iterations in a release
        - planned story points in the release
        - planned budget for the release
      - Actual measurements:
        - total story points completed
        - number of iterations completed
        - Actual Cost (actual spending) to date
        - *any story points added / removed from the plan (as Agile project requirements are ever-changing)*
    - Plotting the chart:

- **x-axis:** iterations / date
  - **y-axis:** i) story points planned; ii) story points completed, iii) planned budget; iv) actual costs
  - discrepancies between i) and ii) / iii) and iv) reflect the performance of the release
  
  - **Formulas for EVM** (not to be tested on the PMI-ACP exam)
    - Schedule Performance Index (SPI) =  $\text{Earned Value} / \text{Planned Value}$
    - Cost Performance Index (CPI) =  $\text{Earned Value} / \text{Actual Cost}$
  
  - EVM does not indicate the quality of the project outcome, i.e. whether the project is successful or not
  
  - **Approved Iterations**
    - After the time assigned to an iteration has been used up, the team will hold a review meeting with the stakeholders (mainly management and customers) to demonstrate the working increments from the iteration.
    - If stakeholders approve the product increment against the backlog selected for the iteration, the iteration is said to be approved.
    - Approved iterations may be used in the contract of work as a way to structure the release of partial payment to the contractor.
- 

# Domain III Stakeholder Engagement

The PMI-ACP Exam consists of 120 questions which can be categorised into seven domains. The third domain: **Domain III Stakeholder Engagement** is the knowledge about "engaging 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" (*source: PMI-ACP Examination Content Outline*).

*Domain III Stakeholder Engagement accounts for **17%** of all questions in the PMI-ACP Exam (i.e. **~20 questions** among 120 PMI-ACP Exam questions)*

According to the PMI-ACP Exam Content Outline, Domain III Stakeholder Engagement consists of 9 tasks grouped within 3 sub-domains:

## Understand Stakeholder Needs

1. **Identify continually** who are the key stakeholders in order to understand stakeholders' interests and expectations
2. Engage stakeholders through early and continuous **knowledge sharing** and **active listening** throughout the project lifespan

## Ensure Stakeholder Involvement

1. **Build relationships** with key stakeholders with a working agreement to allow effective collaboration.
2. Ensure all stakeholders are **engaged appropriately** by updating

the stakeholders registry upon changes to the project.

3. Foster **group decision** making and conflict resolution in order to maintain a good relationship among stakeholders.

## Manage Stakeholder Expectations

1. Create a shared vision of the different project aspects (e.g. deliverables, iterations, releases, etc.) with the use of **project vision and objectives** that align with stakeholders' expectations.
2. Agree mutually on the **success criteria** for the project deliverables / project.
3. **Communicate key information** of the project (including progress, risks, quality, etc.) with stakeholders to provide **transparency** into the project status.
4. Provide appropriately detailed project **forecast** to facilitate planning with stakeholders.

## PMI-ACP Study Notes: Domain III Stakeholder Engagement

Below is a collection of the key knowledge addressed in Domain III Stakeholder Engagement and the 9 tasks related to the domain:

- **Stakeholder management**
  - definition of stakeholders: anyone who have an impact on /will be impacted by the project (e.g. sponsor, vendors, final customers, community, etc.)
  - the project team is considered stakeholders in traditional project management (according to PMBOK Guide) but **not** in Agile projects
  - stakeholder management processes:
    - identify all the stakeholders periodically (in particular the key stakeholders who will have a big impact on project success)
    - communicate with selected stakeholders for requirements and needs gathering
    - enhance stakeholder involvement by active communication and information sharing
      - the type and level of details of the information should be

- appropriate for the type of stakeholders
  - show project progress (just detailed enough) with demos / presentations
  - as project evolves, the interests of key stakeholders must be managed actively
  - discuss updated estimates and projections timely and openly (even in case of bad news) so as to facilitate future planning
  - keep a good relationship with all stakeholders by disseminating necessary information and collecting feedback from them
- may need to educate stakeholders about the processes and benefits of **Agile** project management to solicit their support
- stakeholders may be invited to review and planning meetings in order to update them about the project progresses
- **Knowledge sharing**
  - knowledge sharing / transfer is a key component of Agile project management
  - knowledge should be shared across the team, customer, community and organization
- **Active listening** – there are 3 levels of listening skills:
  - Internal Listening (thinking about how things will affect me)
  - Focused Listening (trying to understand what are the speaker is really trying to say)
  - 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)
- **Participatory decision models**
  - encourage and facilitate stakeholders involvement in **decision-making process through simple techniques** such as
    - simple voting
    - thumbs up / down / sideways
    - Jim Highsmith’s Decision Spectrum – pick a value among a spectrum of feeling ranging from “in favour”, “OK with reservation” to “veto”
    - fist-of-five voting – vote with 1 to 5 fingers to express the degree of agreement (i.e. 1 – totally support, 5 – object completely)
  - the simple technique will also every stakeholders to voice out their



opinion with an aim to reach a consensus on the issue

- **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, 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 shippable
- the definition of done (a.k.a. success criteria) must be agreed upon collectively with key stakeholders before carrying out the project works
- the definition of done will align the expectations of the stakeholders and project team to reduce the risk of wasted work
- the definition of done includes acceptance criterion and acceptable risks

- **Workshop**

- workshops can be a great way to encourage active participation of all stakeholders
- better make use of low-tech high-touch tools like whiteboard or post-its to show ideas

- **Conflict resolution**

- There are 5 stages of conflict - in the order of light to severe:
  1. **A Problem to solve** – a problem occur or is presented
  2. **Disagreement** – everyone tries to protect their own interests
  3. **Contest** – people begin taking sides (a you-vs-me situation)
  4. **Crusade** – people in conflict will make over-generalization in judgement, not just about the problem but also about the persons
  5. **World War** – the problem is now unresolvable, either one side will survive
- It is advisable to try to resolve conflicts early in the stage to reach a consensus with effective conflict resolution strategies:
  - **Confronting** – open dialogue (everyone is able to voice out their opinions) leading to problem resolution to create a win-win situation
  - **Collaboration** – working together to reach mutually agreed solution

- **Project charter**

- The project charter is a must-have for Agile project management to help

creating a common understanding of the project objectives, mission and success criteria

- It is the 1st documentation created for the Agile project to help kicking off the project formally
- The project charter will be progressively elaborated as the project evolves
- can be detailed or barely sufficient (for most cases as at the project begin, it is usually little known that what the final product will be)
- Barely Sufficient Project Charter: usually include at least 3 elements:
  1. **Vision**: the purpose of the Agile project - answering the "why" of the project
  2. **Mission**: describes what will be achieved or done - answering the "what" of the project
  3. **Success Criteria**: describe how the project will be considered a success or reach an end
- Detailed Project Charter:
  - Background, objectives, vision (why) and mission (what), stakeholders of the project
  - Preliminary direction, scope
  - High-level budget, timeline
  - High-level risk and constraints
  - Communication plan
  - Success criteria
- Agile charters **address more about the "How" instead of "What" of the project** - such that the Project Charter will not impose unnecessary boundary for the project to evolve
- Can be in the form of an elevator statement adopting the format of
  - For – (target customers)
  - who – (need to do what)
  - , the – (product / service)
  - is a – (product category)
  - that – (key benefits)
  - . Unlike – (competitive products)
  - , we – (primary differentiation)
- **Social media-based communication**
  - social media are a great way to collect ideas, requirements and feedback from the community
    - convenient

- instantly
- two-way communication



# Domain IV Team Performance

The PMI-ACP Exam consists of 120 questions which can be categorised into seven domain. The fourth domain: **Domain IV Team Performance** is the knowledge about "creating 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" (*source: PMI-ACP Examination Content Outline*).

*Domain IV Team Performance accounts for **16%** of all questions in the PMI-ACP Exam (i.e. **~19 questions** among 120 PMI-ACP Exam questions)*

According to the PMI-ACP Exam Content Outline, Domain IV Team Performance consists of 9 tasks grouped within 3 sub-domains:

## Team Formation

1. The team works together to **establish ground rules** and processes to strengthen sense of belonging and create a shared goal of the team members.
2. Form the team with **members who possess all the necessary skills** (interpersonal and technical) to deliver the intended outcomes and values of the project.

## Team Empowerment

1. Create a high performing team in which members can perform as **generalizing specialists** carrying out **cross-functional** tasks.
2. **Empower** team members to make decisions and to lead in order to

create a **self-organizing team**.

3. Understand **motivators and demotivators** of the team and individuals to ensure high team morale.

## Team Collaboration and Commitment

1. Make use of **collaboration tools and colocation to enhance communication** within the team and between team and stakeholders.
2. **Shield the team from external distraction** and pressure to ensure performance.
3. **Align the goals of the project and the team members** with a shared project vision.
4. Measure the **team velocity** by tracking work performance in previous iterations to allow more accurate forecasts.

## PMI-ACP Study Notes: Domain IV Team Performance

Below is a collection of the key knowledge addressed in Domain IV Team Performance and the 9 tasks related to the domain:

- Individuals and interactions over processes and tools - i.e. in Agile project management, the team members and their interaction are considered far more valuable than following pre-defined processes or toolsets.
- Business people and developers must work together daily throughout the project / Build projects around motivated individuals / Agile processes promote sustainable development / Self-organizing teams / The team reflects on how to become more effective
- **Team Formation Stages**
  - Tuckman model (Tuckman's stages of group development)
    1. **Forming** - the team is formed, everyone behaves independently
    2. **Storming** - disagreements arise between team members
    3. **Norming** - team members accept each other by emphasising the team goal
    4. **Performing** - the team is highly motivated and efficient
    5. **Adjourning** - tasks completed

- At stage 4, the team is considered to be most efficient and best performing. However, not every team goes through every stage of the Tuckman model, some may stay at stage 2 and jump to stage 5 without going through stages 3 and 4.
- **Building Empowered Teams**
  - Agile teams are, ideally, highly motivated by practising self-management and self-organization. The organization gives the Agile team a high level of trust.
    - a team is *"a small number of people with complementary skills who are committed to a common purpose, performance goals and approach for which they hold themselves mutually accountable"* ~Jon Katzenbach and Douglas Smith
  - empowered teams are:
    - **self-organizing**: since the team has the best knowledge about the project and is in the best position to organize project works
    - **self-directing**: the team can make their own decisions, not to be directed from the management
    - empowered team is more productive and efficient than teams with top-down decision making
    - mutual accountability and collective project ownership promote empowerment so that the team work as one whole
- **Tabaka's model for high-performing team**
  - self-organization
  - empowered to make decision
  - belief in vision and success
  - committed team
  - trust each other
  - participatory decision making
  - consensus-driven
  - construction disagreement
- **High Performing Team vs Low Performing Team**
  - maximize performance by
    - clear and realistic goals
    - building trust
    - open and honest communication – even in case of disputes or conflicts
    - taking ownership, empowered, self-organizing
    - coaching and mentoring

- choose teammates with complementary skills to perform all tasks
- sense of belonging (identity)
- limiting each team to have 12 members or below, break down the team if needed
- make decisions through consensus (participatory decision model)
- full-time, dedicated members
  
- low performing teams are:
  - absence of trust
  - fear of conflict
  - lack of commitment
  - avoidance of accountability
  - inattention to results
  
- **Team Participation**
  - The whole project team would discuss in details about the requirements of customers through face-to-face communication:
    - **Brainstorming** – everyone can voice out their opinions freely **without immediate judgement**
    - **Innovation games** – games are used to engage the team members and customer, e.g. 20/20 Vision, the Apprentice, Buy a Feature, Product Box, Prune the Product Tree (reference: Innovation Games: Creating Breakthrough Products Through Collaborative Play by Luke Hohmann).
    - **Parking lot chart** – a piece of paper to put important but off-topic issues / queries for later investigation / discussion, e.g. in requirement gathering
  
- **Two-way Communication**
  - Agile project management emphasizes feedback as feedback can help reducing mis-understanding and risks and generate better ideas.
  - Communication must always be two-way, all the parties are given the opportunity to voice out their concerns and points of views
  
- **Cross-functional Team**
  - a group of people with different functional expertise working together toward a common goal
  - often function as self-directed teams
  - members must be well versed in multi-tasking as they are simultaneously responsible for various functions
  
- **Agile Coaching and Mentoring**

- Coaching and mentoring are needed to help steer the team in the right direction:
  - coaching– help achieving (personal / organization) goals
  - mentoring – pass on skills, knowledge and experience
- **Agile Leadership Style: Servant Leadership**
  - traditional leadership and management emphasizes on command-and-control (i.e. Theory X – workers are lazy and need to be monitored closely)
  - servant leaders will lead by serving to ensure the needs of team members are met and roadblocks are cleared (Theory Y – team members are self-motivated)
  - an Agile servant leader needs to:
    - protect the team from interference, distractions and interruptions
    - remove impediments to the team’s performance
    - communicate and re-communicate project vision – maintain a common vision to drive the team to perform
    - carry food and water – i.e. provide all the resources for the team to perform, including motivate the team, provide trainings
- Important tools/processes/concepts to enhance team communication:
  - **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
  - **team space** - prefer all team members to be collocated in the same room facing each other for pro-active support, free discussion, open collaboration, tacit knowledge sharing and osmotic communication. If physical co-location is impossible, can make use of virtual co-location tools (e.g. instant messaging, video conferencing, etc.)
  - **Agile tooling**
    - these are tools to promote more effective communication (e.g. reduce roadblocks for collecting, maintaining and disseminating information)
    - types: **low-tech high-touch tools**, digital tools
      - low-tech high-touch tools are preferred because these tools can promote collaboration and communication and everyone knows how to participate
      - Examples:
        - co-located teams:
          - war room and/or a dedicated conference room (walls filled with information radiators like whiteboards, billboards, post-its,



- charts, task boards, etc.)
  - distributed teams:
    - virtual shared space using digital tools (wikis website, instant messaging, online planning poker, card meeting, version control, CASE tools, other Agile tools for building/configuring/deploying deliverables)
- **osmotic communications** for colocated and/or distributed teams
  - team members in a co-located space can overhear conversations/discussion of other members
  - the team members will be able to extract useful parts from the conversations or to join in if necessary
- **daily stand-ups**
  - daily stand-ups are a **time-boxed** (~ 15 minutes) and focused meeting to be held at the same time and in the same place for all team members to do a quick update on the project
  - stakeholders may attend but are **not** allowed to talk during the meeting
  - Each member answer the following 3 questions:
    - *What have you done since last meeting?*
    - *What are you planning to do by next meeting?*
    - *What impediments (obstacles) are impacting your work progress?*
- **Motivational Theories**
  - **Maslow's Hierarchy of Needs** – five levels of personal needs, from the fundamental at level 1 to the ultimate need at level 5
    1. Physiological
    2. Security
    3. Social
    4. Esteem
    5. Self Actualization
  - **Herzberg's Hygiene Theory**
    - **satisfaction (motivators)**: such as recognition, achievement or personal growth
      - these are key factors to make team members motivated
      - salary is not an effective motivator
    - **dissatisfaction (demotivators)**: such as bad working conditions, unfairness, etc.
      - hygiene factors are factors that must be present to avoid

dissatisfaction but do not provide satisfaction, also called KITA factors

- e.g. Company policies, supervision, relationship with supervisor and peers, work conditions, **salary**, status, job security

- **Expectancy Theory**

- an individual will decide to behave or act in a certain way because they are motivated to select a specific behavior over other behaviors due to what they expect the result of that selected behavior will be
  - for a person to be motivated, efforts/performance/outcome must be matched – will only work hard for achievable goals
- key elements of Expectancy Theory
  - Expectancy (extra work will be rewarded)
  - Instrumentality (good results will be rewarded)
  - Valence (the individual's expected reward)

- **Productivity**

- both velocity and throughput can be used to measure the productivity of a team

- **Team Velocity**

- Velocity is a capacity planning tool used in Agile project
  - usually defined as the number of story points that are completed in an iteration
- Velocity usually increases gradually over the first few iterations as the team becomes more "performing" but stabilises afterwards as the product becomes more complicated (more bugs, more documentations, more dependencies, etc.)

- **Cycle Time and Throughput**

- Cycle time is the time necessary to get a single item of work done from start (idea) to finish (as a shippable product that delivers value)
  - Cycle time can be reduced by shortening iteration time (breaking down task sizes), limiting Work In Progress and reducing wastes
- Throughput is the number of things that can be done in an iteration
- **Cycle Time = WIP / Throughput**
- **Defect cycle time** is the time between defect injection and defect remediation, the shorter the defect cycle time the better

- **Emotional Intelligence** – people with higher emotional intelligence (E.Q.) can

relate to the feeling of people so that they deal with people issues more effectively

- According to Higgs & Dulewicz, emotional intelligence includes seven components
  - Self-awareness
  - Emotional resilience
  - Motivation
  - Interpersonal sensitivity
  - Influence
  - Intuitiveness
  - Conscientiousness
- **Negotiation**
  - collaboration over contract negotiation
  - communicate with two or more parties to reach an agreement and resolve conflicts
  - Negotiation strategies
    - **Distributive negotiation**: adopt extreme positions initially and work to reach a deal through tactics (the assumption is value is limited, everyone needs to fight for the best value they can get)
    - **Integrative negotiation**: work together collaboratively to achieve greater successes by creating more values for a **win-win solution** (value can be created)
- **Conflict Resolution**
  - conflict is inevitable and is good for project success when controlled
  - focusing on turning conflicts into a win-win situation, often need to make use of emotional intelligence and active listening
  - conflict resolution tactics:
    - Accommodation – identify points of agreements and play down disagreement
    - Avoidance – ignore the conflict
    - Compromise – both sides to give up something, a lose-lose situation
    - Forcing – force one side to accept something, a win-lose situation
    - **Confronting** – open dialogue leading to problem resolution, a win-win situation
    - **Collaboration** – work together for mutually consented solution



# 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 improve 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 / 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 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, Estimatable (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 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 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 **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
  - also refers to the various **modeling 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 Alistair 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 contribute **0 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 behavior 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.



# Domain VI Problem Detection and Resolution

The PMI-ACP Exam consists of 120 questions which can be categorised into seven domain. The Sixth domain: **Domain VI Problem Detection and Resolution** is the knowledge about "*Continuously identifying problems, impediments, and risks; prioritizing and resolving in a timely manner; monitoring and communicating the problem resolution status; and implementing process improvements to prevent them from occurring again.*" (source: PMI-ACP Examination Content Outline).

*Domain VI Problem Detection and Resolution accounts for **10%** of all questions in the PMI-ACP Exam (i.e. **~12 questions** among 120 PMI-ACP Exam questions)*

According to the PMI-ACP Exam Content Outline, Domain VI Problem Detection and Resolution consists of 5 tasks:

1. **Encourage experimentation and communication** in order to discover problems or impediments that prevent maximal value delivery.
2. Identify and resolve issues and threats timely by **engaging the whole team**.
3. Issues should be **resolved by appropriate team member(s)**. In the case the issue cannot be resolved, the team should communicate with appropriate stakeholders to adjust project expectations and priorities.
4. Keep a **prioritized list of issues/threats/risks** to track assignment/ownership and the current status and to provide transparency.

5. Incorporate **resolution activities into task backlog** for future planning.

## PMI-ACP Study Notes: Domain VI Problem Detection and Resolution

Below is a collection of the key knowledge addressed in Domain VI Problem Detection and Resolution and the 5 tasks related to the domain:

### Risk / Threat Management

- Risk is uncertainty that could affect the success/failure of the project. Risks become **problems or issues** once they actually occur.
- Risks can be threats or opportunities, negative project risks are considered as "anti-value".
- In order to maximize values, negative risks must be minimized while positive risks should be utilized. But once problems or issues arise, they must be resolved timely in order to reduce effects on value creation.
- Risk identification should involve the customer, project team and all relevant stakeholders
- **Five Core Risks** mentioned in the book "The Software Project Manager's Bridge to Agility"
  - **productivity variation** (difference between planned and actual performance)
  - **scope creep** (considerable additional requirements beyond initial agreement)
  - **specification breakdown** (lack of stakeholder consensus on requirements)
  - **intrinsic schedule flaw** (poor estimates of task durations)
  - **personnel loss** (the loss of human resources)
- 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)
- As a general rule, "**riskier**" features (with high values) should be tested in

- earlier sprints** to allow the project to "**fail fast**" as failing during the earlier phase of the project is much less costly than failing during a later phase
- Risk is high at the beginning of the project (both for traditional and Agile projects) but Agile projects have higher success rates as the very nature of Agile project management tends to reduce risks as changes are inherent to the projects
- Risk can be categorized into the following:

  - Business – related to business value
  - Technical – about technology use and/or skill sets
  - Logistic – related to schedule, funding, staffing, etc.
  - Others – Political, Environmental, Societal, Technological, Legal or Economic (PESTLE)
- To tackle risks: **Identify Risks -> Assess Qualitatively and Quantitatively -> Plan Response -> Carry Out Responses Should Risks Arise -> Control and Review**
- Risk Adjusted Backlog**

  - Prioritization criteria for backlog: **value, knowledge, uncertainty, risk**
  - Backlog can be **re-prioritized** by customers as needed to reduce risks while still realizing values
    - The customer can give each feature / risk response actions (non-functional requirements) on the backlog a value by, for features, assessing ROV and, for the case of risks, the costs involved (by multiplying the probability of the risk in %)
    - The backlog of features and risk response activities can then be prioritized based on the dollar values
  - Risk adjustment tries to identify and mitigate the risk at an early stage of development
  - 'Fail fast' allows the team to learn and adjust course
- Risk Burn Down Graphs / Charts**

  - to show the risk exposure of the project
  - created by plotting the sum of the agreed risk exposure values (**impact x probability**) against iterations
  - to be updated regularly (per iteration) to reflect the change in risk exposure
  - general recommendation: top 10 risks are included
  - the risk burn down chart should have the total risks heading down while the project progresses



- **Risk-based Burn Up Chart**
  - tracks the targeted and actual product delivery progress
  - includes estimates of how likely the team is to achieve targeted value adjusted for risk by showing the optimistic, most likely and the worst-case scenario
- **Risk-based Spike**
  - **Spike**: a rapid time-boxed experiment (by the developers) to learn just enough about the “unknown” of a user story for estimation / establishing realistic expectations / as a proof of concept
  - the “unknown” can be: new technologies, new techniques
  - can be a “proof of concept”
  - **spikes are carried out between sprints and before major epics / user stories**
  - products of a spike are usually intended to be thrown away
  - types :
    - **architectural** spike: associated with an unknown area of the system, technology or application domain
    - non-architectural spike: others
  - **Risk based spike**: a spike to allow the team to eliminate or minimize some major risks
    - if the spike fails for every approach available, the project reaches a condition known as “fast failure”, the cost of failure is much less than failing later

## Problem Detection

- **Definition of Done (DoD)**
  - **Done** usually means the feature is 100% complete (including all the way from analysis, design, 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 shippable
  - The exact definition of done has been be agreed upon by the whole team (developer, product owner / customer, sponsor, etc.)
  - The definition of done includes acceptance criterion and acceptable risks
- **Frequent Validation and Verification**

- early and frequent testing both within and outside the development team to reduce the cost of quality (change or failure)
  - **validation**: (*usually external*) the assurance that a product, service, or system meets the **needs** of the customer
  - **verification**: (*usually internal of team*) the evaluation of whether or not a product, service, or system complies with a regulation, **requirement**, specification, or imposed condition
- Agile measure to ensure frequent validation and verification:
  - **testers** are included in the development team from the beginning taking part in user requirements collection
  - **unit testings** are created for continuous feedback for quality improvement and assurance
  - **automated testing tools** are used allowing quick and robust testing
  - examples: peer reviews, periodical code-reviews, refactoring, unit tests, automatic and manual testing
  - feedback for various stages: team (daily) -> product owner (during sprint) -> stakeholders (each sprint) -> customers (each release)
- **Variance and trend analysis**
  - variance is the measure of how far apart things are (how much the data vary from one another)
    - e.g. the distribution of data points, small variance indicates the data tend to be close to the mean (expected value)
  - **trend analysis** provides insights into the future which is more **important** for problem detection
    - though measurements are lagging, they will provide insights should trends be spotted
  - variance and trend analysis is important for controlling (**problem detection**) and continuous improvement, e.g. the process to ensure quality
  - **Control limits for Agile projects**
    - by plotting the time to delivery / velocity / escaped defects / etc. as a control chart
    - if some data fall outside the **upper / lower control limits**, a root cause analysis should be performed to rectify the issue
      - **common cause** – systematic issue, need to be dealt with through trend analysis
      - **special cause** – happens once only due to special reasons

- another example is the WIP limit in Kanban boards
- **Escaped Defects**
  - Agile performance (**on quality**) can also be measure by the number of escaped defects (defects found by customers)
    - defects should be found and fixed during coding and testing
    - defects found early are much less expensive to fix than defects found late

## Problem Resolution

- **The Five WHYs**
  - a systematic approach to analysing identifying the root cause of a problem / cause-and-effect for the problem or issue
  - perform by repeatedly asking the question "Why" for at least 5 times until the root cause has been identified
  - *imaginary example*: Looking for the root cause for failing the PMI-ACP Exam
    1. **Why** did I fail the PMI-ACP Exam?
      - Because I got a lower mark than the passing mark
    2. **Why** did I get a lower mark?
      - Because I was not sure about the answers to many questions.
    3. **Why** was I not sure about the answers to many questions?
      - Because I could not remember some facts for the exam.
    4. **Why** couldn't I remember some facts for the exam?
      - Because I was not familiar with the PMI-ACP Exam content.
    5. **Why** was I not familiar with the PMI-ACP Exam content?
      - Because I did not spend enough time revising the PMI-ACP Exam notes.
- **Fishbone Diagram Analysis**
  - another tool for carrying out cause and effect analysis to help discover the root cause of a problem or the bottle-necks of processes
  - aka cause and effect diagrams/Ishikawa diagrams
  - to use Fishbone diagram technique:
    1. write down the problem/issue as the "fish head" and draw a horizontal line as the "spine" of the fish
    2. think of major factors (at least four or above) involved in the problem/issue and draw line spinning off from the spine representing each factor
    3. identify possible causes and draw line spinning off the major factors

(your diagram will look like a fishbone now)

4. analyze the fishbone diagram to single out the most possible causes to carry out further investigation



# Domain VII Continuous Improvement (Product, Process, People)

The PMI-ACP Exam consists of 120 questions which can be categorised into seven domain. The Seventh domain: **Domain VII Continuous Improvement (Product, Process, People)** is the knowledge about "*Continuously improving the quality, effectiveness, and value of the product, the process, and the team.*" (source: PMI-ACP Examination Content Outline)

*Domain VII Continuous Improvement (Product, Process, People) accounts for 9% of all questions in the PMI-ACP Exam (i.e. ~11 questions among 120 PMI-ACP Exam questions)*

According to the PMI-ACP Exam Content Outline, Domain VII Continuous Improvement (Product, Process, People) consists of 6 tasks:

1. **Review product, processes and practices periodically** to look for rooms for improvement and efficiency enhancement.
2. **Conduct frequent retrospectives and experiments** to continually improve team processes and effectiveness.
3. Gather **feedback from stakeholders** on product increments and demonstrations to enhance value delivery.
4. Develop a team of generalising specialists by providing **learning and practising opportunities**.
5. Perform **value stream analysis** on existing processes to **remove wastes** and improve efficiency.
6. **Disseminate knowledge** gained during carrying out the project works to the whole organization for organizational improvement.

# PMI-ACP Study Notes: Domain VII Continuous Improvement (Product, Process, People)

Below is a collection of the key knowledge addressed in Domain VII Continuous Improvement (Product, Process, People) and the 6 tasks related to the domain:

## Integration, Testing and Experiments

- **Continuous Integration** (as a core practice in XP)
  - to continuously integrate changes (usually in small trunks) to the codebase by merging the new codes as soon as practicable (i.e. once ready)
  - to **avoid code conflicts** and minimize risks of incompatibility
  - on every integration, the codebase needs to be tested (usually by unit testing with automated testing tools / regression testing tools)
  - typical setup for continuous integration:
    1. A source code repository
    2. A check-out and check-in process
    3. An automated build process (compiles codes, runs tests and deploys)
  - if errors are found, **fixing** the broken build is of **top priority**
- **Continuous Improvement**
  - the Deming's **PDCA Cycle** (plan – do – check -act)
  - make use of process improvement and self-assessment for improved product
  - e.g. code refactoring and pair programming
- **Testing (Exploratory and Usability)**
  - Exploratory testing - seeks to find out how the software actually works, and to ask questions about how it will handle difficult and easy cases by asking test subjects to try the software
  - Usability testing - a special type of exploratory testing with emphasis on the usability of the software interface (whether the test subject will be able to perform core tasks on the interface without instructions and help)
  - help to provide insights on the design of the software:
    - Users' expectations / habits
    - Users' ability to understand / comprehend the design of the interface
    - Users' value of the functions of the software

- both will provide valuable feedback early in the project to enhance value delivery and avoid failure later on
- **Learning Cycle**
  - Agile software development is about learning - from little known about the end product in the beginning to (hopefully) delivering the maximal value in the end
  - understanding of the requirements as well as the technology to make the product feasible increase incrementally during the project
  - each retrospective/review is an opportunity to learn
  - it is recommended to keep learning cycles short so that new knowledge gained can be fed into the project as soon as possible

## Review and Retrospective

- **Retrospective**
  - an Agile process for self-evaluation to be performed at the end of each iteration (somehow similar to the "postmortem" meeting or "lessons learned" meeting in traditional project management)
  - a continuous process improvement for timely implementation
  - involved the **Agile development team only with a timebox of up to 1 hour**
    - a valuable learning opportunity for the Agile team
    - analyze, adapt and improve the entire development process
    - improve productivity, capability, quality and capacity
  - actionable improvement tasks are to be implemented right in the next iteration
    - for instant improvements
  - focus on **what went well, what went wrong and how the team can improve** in next iteration and beyond without finger-pointing
  - typical agenda:
    1. set the stage – get people comfortable to speak and outline the topics for discussion
      - check-in – everyone express in 1 or 2 words about the expectation of the retrospective
      - focus on / focus off – which side to focus on (e.g. dialogue vs debate)
      - ESVP – choose 1 from among “explorers, shoppers, vacationers and prisoners” that describes their feeling anonymously
      - working agreements – work on different topics in small groups first

2. gather data
  3. generate insights
  4. decide what to do – identify the high priority items to devise an action plan
  5. close the retrospective – express appreciation and feelings
    - Plus / Delta – what should be done more / what should be changed
    - Helped, Hindered, Hypothesis – three flip charts for participants to add ideas on
- the improvement stories chosen in the retrospective will be treated as **non-functional backlogs**
  - **Retrospective Meeting vs Review Meeting**
    - Retrospective meeting is for the **development team only** with the primary aim for process improvement
    - Review meeting is for **demonstration** / getting acceptance of deliverables with management, product owner and stakeholders
  - **Retrospective Meeting vs Lessons Learned Meeting**
    - 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 (according to PMBOK Guide) for future references (not as a feedback to the project itself)
  - **Intraspectives**
    - intra = inside / within, (per)spective = a particular way of viewing things / inspection: intraspective = inspecting within / seeing inwardly
    - intraspectives in Agile project management is an *ad hoc* discussion / meeting by the Agile team to review on the team practices or teamwork during the sprint, often called for when something went wrong
  - **Pre-mortem** (rule setting, failure analysis)
    - A pre-mortem is the hypothetical opposite of a postmortem in which team members are asked to **generate plausible reasons** for the project's assumed failure.
    - To make it safe for team members to voice out their reservations about the plan / project direction / etc.
    - Can identify possible causes of failure which are missed during risk analysis



# Value Stream Analysis and Mapping

- The objectives of value stream analysis:
  - to provide **optimum value flow to customers** through value creation processes
  - by eliminating wastes in every process through analysis (e.g. value stream mapping) and enhancements
- **Value Stream Mapping**
  - **Simplicity** – the art of maximizing the amount of work not done –**is essential**
  - Value stream mapping is originally a graphical tool for analyzing the **flow** of materials in manufacturing from its beginning through to the customer (e.g. in lean manufacturing).
  - It is later adopted for value creation of services (e.g. IT development projects).
  - usually involves the following steps:
    1. understand the current state (**current state**)
      - create a visual map of the value flow of the current state
      - distinguish between value-adding processes and non-value-adding operations (including wastes)
      - find delays, wastes and constraints
    2. analyze and modify (the ideal **future state**)
      - create a new value stream map for the desired state after optimization (e.g. removing delays, wastes and constraints)
    3. communicate and carry out the improvements
      - ensure all team members understand the values of the improvement work
      - develop a roadmap for implementing the actions
    4. verify and validate the improvements



# More PMI-ACP Free Resources

Additional FREE PMI-ACP study resources, study and exam tips and my PMI-ACP Exam lessons learned can be found on my website below:

[edward-designer.com](http://edward-designer.com)