ADAPTIVE PLANNING
ADAPTIVE PLANNING

Adaptive planning is the conscious acceptance that early plans are both necessary and likely to be flawed; therefore, replanning and adaptation activities should be scheduled into the project.

An adaptive approach acknowledges that planning is an ongoing process and has multiple mechanisms in place to proactively update the plan.

Adaptive planning differs from more static planning approaches that create most of the plan upfront. The more static approaches are reactive, rather than proactive; after the initial plan is created, planning is typically done only in response to exceptions to the plan and change requests.
PLANNING CONCEPTS

The planning concepts embodied by agile methods promote an acceptance that details will emerge as the project progresses, that it is necessary to adapt the project based on feedback, and that frequent reprioritization is the norm.

The PMI-ACP exam will test your knowledge and ability to apply the following planning concepts:

1. Plan at multiple levels.
2. Engage the team and the customer in planning.
3. Manage expectations by frequently demonstrating progress and extrapolating velocity.
4. Tailor processes to the project’s characteristics.
5. Update the plan based on the project’s priorities.
6. Ensure encompassing estimates that account for risks, distractions, and team availability.
7. Use appropriate estimate ranges to reflect the level of uncertainty in the estimate.
8. Base projections on completion rates.
9. Factor in diversions and outside work.
PLANNING CONCEPTS

TIMEBOXING

Timeboxes are short, fixed-duration periods of time in which activities or work are undertaken. If the work planned for the timebox is not complete when the time runs out, then we stop what we are doing and move the uncompleted work into another timebox. Examples of timeboxes include:

- Daily stand-up meetings that are timeboxed to 15 minutes
- Iterations that are timeboxed to (typically) two weeks

Timeboxes help bring some level of order and consistency to an otherwise highly variable work environment. They offer an opportunity to assess results, gather feedback, and control the costs and risks associated with an endeavor. They provide frequent checkpoints to gauge progress and replan the ongoing approach.
PLANNING CONCEPTS

PROGRESSIVE ELABORATION

Progressive elaboration is the name given to the process of adding more detail as information emerges. We use progressive elaboration to evolve and create increasingly accurate:

- Plans
- Estimates
- Risk assessments
- Requirements definitions
- Architectural designs
- Acceptance criteria
- Test scenarios
PROGRESSIVE ELABORATION IN PLANS AND ESTIMATES

At the beginning of a project, we need to plan and estimate the work involved to determine how big the endeavor is likely to be and to create a reasonable strategy and execution approach. However, we also need to understand that the beginning of a project is when we know the least about the endeavor. At this early point, there has not yet been any “learning by doing” on the project.

Instead of limiting our planning and estimation activities to the start of the project, we must continually refine our plans and estimates as the project progresses and new details emerge. This process of continual updates is the essence of progressive elaboration.
**PLANNING CONCEPTS**

**PROGRESSIVE ELABORATION CONTD**

In figure 1, the “Now” arrow indicates that we are in the third iteration. The color in the early iterations show how detailed the plans for those iterations are. As you can see, the first four iterations have a lot of details, but there is less and less detail for the subsequent iterations.

This process of refining the plans as we get closer to working on the iterations continues throughout the project.
PLANNING CONCEPTS

PROCESS TAILORING

Just as our plans evolve as we learn more about a project, its environment, and its stakeholders, so too does our approach. Retrospective reviews are the main trigger for driving process changes. At the end of each iteration, we meet with the extended team (our development team and other business stakeholders) to ask the following questions:

- What is going well?
- What areas could use improvement?
- What should we be doing differently?

As problems are identified, we engage the team in brainstorming solutions. We then commit to trying the selected solutions for one or two iterations before meeting again to discuss whether the situation has improved. If the actions helped – great! We adopt them as part of our processes on the project. If they did not help, we consider the effort a learning opportunity and decide whether to try something else or revert to the earlier process.

Through this cycle of regular inspection, reflection, and adaptation, we tailor our project processes to the unique situation of the project and organization.
MINIMALLY MARKETABLE FEATURE (MMF)

When planning a release of features to customers, the release has to make sense, be useful, and be valuable. This applies to all types of agile projects, whether it’s a release of software, a new electrical product, or an engineering increment. The term “minimally marketable feature” (MMF) refers to this package of functionality that is complete enough to be useful to the users or market, yet small enough that it does not represent the entire project.

For a cell phone, for example, a minimally marketable feature could be a phone that can be used to make and receive calls, store contact names and numbers, and access voice mail, but the phone would not need to have a camera, Internet connectivity, or a music player in its first release. Instead, these sets of functionality could be added in subsequent releases and evaluated independently. Keep in mind, however, that the functionality of the phone that is released in the MMF needs to be complete. So all the attributes related to making phone calls should be present as part of the MMF to allow the customer or business to comprehensively review this functionality.
VALUE-BASED ANALYSIS

Value-based analysis is the process of considering the business value of work items and then acting accordingly. It affects the full life cycle of agile projects and is holistic in the sense that the business value impacts how we scope, plan, schedule, develop, test, and release work. At every stage of the project, we are asking, “What is the business value of this item or practice?” and “What items in this set have the highest business value?” We then prioritize the work to deliver the highest-value items first.

To fully understand an item’s value, we also need to understand the development and delivery cost. For example, a feature that is worth ₦500,000 to the business but costs ₦400,000 to develop (or even ₦600,000 to develop) is not as valuable as something that delivers ₦300,000 in value to the business and only costs ₦100,000 to develop.
So we need to factor in likely development costs when performing value-based analysis. This is one reason why we estimate the backlog items at a high level early in the project. Doing so allows us to make benefit-versus-cost value comparisons. This information is then used in prioritizing items for development.
The other component we need to consider is payback frequency. Does the feature generate business value every week or month or just once. Therefore, when we evaluate the return over a one- to five-year period, the extended view of the return should be used to assess the true business value.

Keeping these components in mind, we analyze the real business value of features so that we can then prioritize the features appropriately. However, we need to remember that some high-business-value items may be dependent on some lower-business-value items, and so these lower-business-value items will need to be undertaken first, to allow the team to deliver the high-business-value items as soon as possible.
Value-based decomposition and prioritization is the process of eliciting requirements from stakeholders, ranking those requirements, and then pulling the prioritized requirements into the development process.

The way the project is kicked off may vary from project to project, but regardless of the approach taken, there should be an initial effort to define the vision. The process of identifying additional requirements, grouping or breaking them down into functional elements, and prioritizing those elements repeats over and over at an increasingly refined and more detailed level, similar to a fractal leaf pattern (see figure 3).

This is an example of how agile methods tackle what the PMBOK® Guide calls “progressive elaboration”.

In value-based decomposition and prioritization, we pull forward elements of the project – in whatever size and level of detail is appropriate for the project phase we are in and how much we currently know about the project – into a format that we can work with for the next process. We then continue to refine and elaborate those elements, adding increasing levels of detail and transforming them into a format that can be further refined in subsequent processes. The end product of the progressive elaboration is a highly detailed deliverable that is still true to the original design objectives.
With agile methods, we do not attempt to specify fully detailed requirements upfront. Instead, we initially keep the requirements “coarse grained,” and then progressively refine them as the process progresses. This approach has a number of advantages:

- It helps keep the overall design balanced so the product does not become lopsided by over-development in any particular area.
- It delays decisions on implementation details until the “last responsible moment.” This means we are not rushing to develop things that may later need to be changed as a result of new information or late-breaking change requests.
AGILE GAMES (COLLABORATIVE/INNOVATION GAMES)

Innovation games, also known as collaborative games, are facilitated workshop techniques that agile teams use to help stakeholders better understand complex or ambiguous issues and reach consensus on an agreed-upon solution.

The following are examples of collaborative games used on agile projects:

- **Remember the future**: This is a vision-setting and requirements-elicitation exercise.
- **Prune the Product Tree**: This exercise helps gather and shape requirements.
- **Speedboat**: The purpose of this exercise is to identify threats and opportunities (risks) for the project.
- **Buy a Feature**: This is a prioritization exercise.
- **Bang-for-the-Buck**: This exercise looks at value versus cost rankings.
PLANNING CONCEPTS

AGILE GAMES (COLLABORATIVE/INNOVATION GAMES) CONTD

REMEMBER THE FUTURE: This facilitated exercise engages project stakeholders in imagining that the release or iteration is now complete. They then describe what they imagine has occurred for the iteration or release to be successful.

HOW IT WORKS: We get the project stakeholders, including the team, users, and sponsors, together and ask them to imagine that it is now six months plus two weeks from the current date using the example of planning a release six months out. The reason we tell them to imagine two weeks after the end of the release is because that’s often how long it takes for all the acceptance and implementation “dust” to settle.

The exercise starts with each stakeholder working independently. For the next 20 minutes, each person’s job is to list what was completed for the release to be successful. The stakeholders should record the completed and delivered items on sticky notes, with one item on each note.
Once the 20 minutes are up, everyone transfers their sticky notes to a wall. Then, as a team, the stakeholders work together to group the sticky notes into associated clusters and remove any duplicates. This process can take another 20 minutes as people clarify the meaning of their sticky notes and create headings to correctly identify each group of items.

The purpose of the Remember the future game is not really to predict the future. Instead, we are trying to better understand the stakeholders’ definition of success and how we can achieve that successful outcome.
PLANNING CONCEPTS

AGILE GAMES (COLLABORATIVE/INNOVATION GAMES) CONT'D

PRUNE THE PRODUCT TREE: This exercise engages the participants in brainstorming a product’s functionality and features.

HOW IT WORKS: For this game, we start by drawing a big tree with a trunk and branches on a whiteboard or flip chart. Artistic ability does not matter here – we are just creating a placeholder for features. We then invite the participants to add the features as leaves to the product tree. A good way to explain this part of the exercise is to say that the tree is the product, in that the trunk represents what we already know or have built so far, and the outer branches represent new functionality that has yet to be designed. Encourage the stakeholders to group related features close to each other on the tree. Supporting features should be closer to the trunk, and features that are dependent upon those supporting features should be further out or higher up on the tree.
PLANNING CONCEPTS

AGILE GAMES (COLLABORATIVE/INNOVATION GAMES) CONTD

Fig. 4: Prune the Product Tree
AGILE GAMES (COLLABORATIVE/INNOVATION GAMES) CONTD

SPEEDBOAT OR SAILBOAT: This game uses the features and user stories identified in the Prune the Product Tree game. It focuses on gathering project risks – both bad risks, which are threats to the project, and good risks, which are potential opportunities. This exercise is very quick to set up and facilitate, and it typically results in a good list of project risks.

HOW IT WORKS: To start, we place a whiteboard or flip chart to the left of the Product Tree diagram from the previous exercise. We then draw a waterline and a picture of a boat, with the boat facing in the direction of the Product Tree. We explain to participants that the boat represents the project. For example, we may say “Here is the project heading toward the goal we just developed. What are the anchors (or threats) that could slow us down or even sink us? And what other factors (or opportunities) would be wind in our sails and help propel us toward our goal?”

The stakeholders then work as a group to create “anchor” sticky notes for the threats and impediments to the project. Anchor notes are posted below the waterline. The stakeholders also create “wind” sticky notes for opportunities, which they post above the waterline.
PLANNING CONCEPTS

AGILE GAMES (COLLABORATIVE/INNOVATION GAMES) CONTD

Fig. 5: Speedboat or Sailboat
ESTIMATION

The following are some general points and good practices for agile estimation to keep in mind:

- **Why do we estimate?** Estimates are necessary for sizing and approving projects, calculating ROI and IRR, and determining which pieces of work can be done within a release or iteration.

- **How are estimates created?** Estimates are created by progressing through the stages of determining the project’s size, effort, schedule, and finally cost.

- **How should estimates be stated?** We use the term “estimates” and not “predictions” because there is some degree of uncertainty in an estimate. Therefore, estimates should be stated as ranges (e.g., “₦400,000 to ₦450,000” or “16 to 18 months”) to manage expectations about the project’s uncertainty.

- **When do we estimate?** We should not reserve estimating for when we know the least about the project – at the beginning. This means we need to estimate continuously throughout the project, factoring in the actual costs or durations to date to create better estimates for the project going forward.

- **Who estimates?** We shouldn’t reserve estimating for the *person* who knows the least about the project’s execution or acceptance – the project manager. Instead, we need to get the team members who will be doing the work involved with the estimation process.
ESTIMATION

WIDEBAND DELPHI AND PLANNING POKER

Wideband Delphi is a group-based estimation approach. This technique asks a panel of experts to submit estimates anonymously so no one knows which estimates belong to whom. The anonymous approach produces improved estimates because it minimizes both the “bandwagon effect” (where people tend to agree with a prominent viewpoint) and the “halo effect” (where people gravitate to the ideas of experts or superiors, rather than judging the ideas on their own merit.

A wideband Delphi estimation session starts with a planning effort to define the problem. Instead of estimating the whole project in one meeting, the group breaks down the project or large problem into more manageable chunks. The team creates a problem specification, identifies the assumptions and constraints, and outlines the process for subsequent rounds of estimation.
ESTIMATION

WIDEBAND DELPHI AND PLANNING POKER CONTD

Before they begin creating estimates, participants read the problem specification and have an opportunity to raise and discuss qualification questions. They also receive sheets of paper with spaces where they can enter their estimates for different tasks. The facilitator then gathers the estimates and plots them on a chart, without identifying which estimate is from which estimator.

The participants then discuss the different tasks and any assumptions or other significant factors that influenced their estimates before repeating the estimating process. Once all the tasks, assumptions, and significant factors have been discussed, the group repeats the anonymous estimation process. After several rounds of this process, we usually start to see more consensus around the estimates.
ESTIMATION

WIDEBAND DELPHI AND PLANNING POKER

Wideband Delphi estimation is a good technique for agile teams to use, because it is:

- **Iterative:** The process is repeated several times
- **Adaptive:** Based on feedback from other participants, team members have a chance to update and improve their next round of estimates.
- **Collaborative:** It is a team-based collaborative process that improves participants’ buy-in to the results.
ESTIMATION

WIDEBAND DELPHI AND PLANNING POKER CONTD

PLANNING POKER

The wideband Delphi technique is often implemented as “planning poker.” This variation of the technique combines all of the essential elements of wideband Delphi in a fast, collaborative process. Planning poker uses playing cards with numbers on them. The numbers represent sizing units, such as developer days or story points. A set of cards, as shown in figure 6, is given to each planning participant.

Fig. 6: Planning Poker Cards
ESTIMATION

WIDEBAND DELPHI AND PLANNING POKER CONTD

PLANNING POKER CONTD

Once the cards have been distributed, a moderator (often the product owner or customer) reads a user story, which the group then discusses briefly before each estimator selects a card to represent his or her estimate for the user story. The participants all turn over their cards simultaneously so that everyone can see the numbers.

If there is a group of four people, and three of them turn over cards with the number 5 and one person turns over a card with the number 3, the task is recorded as a “5”. Since the range was small and there is little debate about the estimate, the process moves onto the next story to keep the game moving quickly. If, however, there were three cards with the number 5 and one card with the number 13, then the outlier (the “13”) would be discussed.
ESTIMATION

WIDEBAND DELPHI AND PLANNING POKER CONTD

PLANNING POKER CONTD

The process is then repeated – everyone picks up their card and estimates again, taking the new information into account. In doing so, we might see a new consensus emerge around the number 13.

Like fist-of-five voting, planning poker is an example of a participatory decision model. As stakeholders’ involvement in the process increases, so too does their commitment to the outcome.

Planning poker is faithful to the wideband Delphi approach in that it combines the following elements:

- It has multiple iterations of estimation, where needed.
- It requires participants to submit estimates at the same time to help counter the bandwagon effect and the halo effect.
- It allows group convergence on supported estimates.
ESTIMATION

IDEAL TIME

When asking a team to estimate work, the topic of how best to factor in interruptions, diversions, and nonproject work usually crops up. Such nonproject activities may include attending staff meetings, checking e-mails, and going to the occasional doctor’s visit. For example, for salaried team members who have 40-hour workweeks, should we estimate 35 hours a week for them to work on the project and allow 5 hours for them to do these other activities? And what about contract staff? They have fewer staff meetings and seem to come in when sick anyway, so should we count on 38 hours for them?

To discuss about ideal time, we ask team members to estimate as if there were no interruptions. In an eight-hour day, we tell team members to assume all eight hours would be available for work. In doing so, we can get a more accurate sense of the effort involved in the work.

So ideal time is how long something would take, if all the other peripheral work and distractions were removed.
ESTIMATION

RELATIVE SIZING/STORY POINTS

Relative sizing and story points help solve two common problems with estimation:

1. People are not very good at predicting the absolute size of work.
2. The estimation process is difficult and unpopular.

Oftentimes on projects, what should be trivial work takes much longer to complete than anticipated, and sometimes new and unforeseen tasks appear. The whole process takes longer than we expected. We could try to factor these issues into all estimates, but then we would be criticized for padding the estimates. This apparent no-win situation is why estimating has become so unpopular with the team members.

It turns out that while people are not very accurate at making absolute estimates, they are better at (and at least more comfortable with) making comparative estimates.
ESTIMATION

RELATIVE SIZING/STORY POINTS CONTD

If we have known chunks of work already done, we can estimate new pieces of work more quickly and accurately by referencing the known entities. Using the creation of a software system as an example, imagine we have developed a simple input screen and have given that task a relative score of 2 story points. We can then estimate other tasks in reference to the input screen score.

Taking a comparative approach to estimating does not stop weird things from happening or keep activities from taking longer than anticipated on our projects, but switching the estimation unit from hours to story points does make it easier to accept.

Another advantage of estimating in story points versus hours or days is that we remove the artificial ceiling of “hours per week”.

The term you use for the relative measures of work – whether it be “story points,” just “points”, or “gummy bears” – does not matter. The idea is to get away from estimating in hours, which not only creates artificial ceilings but can also create issues when some people have other demands on their time and never really stand a chance of delivering 40 hours’ worth of work functionality in a week.

We should keep the following points in mind when estimating user stories:

- **Teams should own the story point definition:** The story point sizing being used on the project should be created and owned by the team. For example, the team can decide 1 story point equals the effort involved in creating a simple screen. Whatever size unit the team chooses, that is what should be used.
ESTIMATION

RELATIVE SIZING/STORY POINTS CONT'D

- **Story point estimates should be all inclusive:** We should not need to add time to the project for unit testing or refactoring. Instead, the story point estimates should include all known activities.

- **When disaggregating, the totals do not need to match:** When breaking epics into user stories, it is okay if the sum of the estimates for the user stories exceeds the estimate for the epic. The same goes for breaking down user stories into tasks; the sum of the estimates for the tasks might not equal the estimate for the user story.

- **Sizes should be relative:** A 2-point user story should be equivalent to about twice as much effort or time as a 1-point story. A 3-point story should be equivalent to about three times as much effort or time as a 1-point story and about equal to the combination of a 1-point story and a 2-point story.

- **Complexity, work effort, and risk should all count:** The total time required to complete work is a function of the work’s complexity, the effort involved, and its risk. We need to ensure all three attributes are assessed when the team is estimating user stories.
ESTIMATION

RELATIVE SIZING/STORY POINTS CONTD

THE FIBONACCI SEQUENCE

Planning poker cards and story points are often estimated using the Fibonacci sequence, or variations of this sequence. The first numbers in the Fibonacci sequence are “0,1,1,2,3,5,8,13,21 …” This sequence is derived by adding the previous two numbers together to get the next number in the sequence. So 0 +1 = 1, 1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5, and so on. Teams use the Fibonacci sequence to estimate size. It is a naturally occurring sequence that crops up frequently in connection with how things get bigger.

For the exam, you don’t need to understand how to apply the Fibonacci sequence, but it is helpful to understand that this sequence can be used in estimating. With this sequence, there is enough variation in the number values to eliminate most of the squabbling over slight differences in estimates.
AFFINITY ESTIMATING

Affinity estimating is the process of grouping requirements into categories or collections. For agile projects, this technique is often used to group similarly sized user stories together. Affinity estimating is a form of triangulation; it provides a comparative view of the estimates and gives the team an opportunity to do a reality check.

By placing user stories into size categories, it is easier to see whether user stories that are assigned similar estimates are in fact comparable in size. This technique helps the team make sure they have not gradually altered the measurement value of a story point.
ESTIMATION

TIME, BUDGET, AND COST ESTIMATION (VIA SIZE, EFFORT, AND COST ESTIMATION)

Steps involved in estimating:

1. **Determine the size** of the project in story points or ideal time.
2. **Calculate the effort** for the work in hours or person days (or person weeks or months) by determining the availability and capacity of the team.
3. **Convert the effort into a schedule** by factoring in the team size, required resources, and dependencies.
4. **Calculate the cost** by applying labor rates and adding in other project cost elements. The formula for this calculation is:

   \[
   \text{Total cost} = (\text{Time} \times \text{Resource rate}) + \text{Other project costs}
   \]
AGILE PROJECT ACCOUNTING PRINCIPLES

When we consider the costs of running a knowledge worker project, labor costs often comprise a large portion of the expenses. So while there will also be costs associated with equipment, travel, licenses, specialized services, etc, labor costs typically make up the largest segment.

There are different ways to account for labor expenses on the project. Many companies take a team member’s annual salary and divide that number by 52 to determine a weekly salary cost. They then use that figure for project estimates and for budgeting purposes. Other companies consider the fully burdened labor cost, which is the team member’s salary plus the cost of providing office space, computers, benefits, etc. This fully burdened labor cost can easily be 50 percent higher than the salary expense alone for employees.
ESTIMATION

AGILE PROJECT ACCOUNTING PRINCIPLES CONT'D

As project managers trying to estimate projects, we need to know what the organization’s policy is for using burdened versus unburdened costs. We also need to consider the team member’s involvement on the project. Once we understand these variables, we can calculate labor costs for the project.

ESTIMATE RANGES

Estimates should be presented in ranges to indicate our level of confidence in the estimate and to manage stakeholder expectations. We want to avoid single point estimates. We should present estimates as ranges e.g. ₦77,500,000 - ₦82,000,000.

Estimate ranges should be narrower when we are more certain about the estimates and wider when we are less certain.
ESTIMATION

AGILE PROJECT ACCOUNTING PRINCIPLES CONTD

PARKINSON’S LAW AND STUDENT SYNDROME

Agile practices help mitigate the effects of Parkinson’s Law and Student Syndrome. Parkinson’s Law states that work tends to expand to fill the time available. In other words, if we have three months designated for gathering requirements, we will definitely spend all three months gathering requirements. Because agile projects focus on short time spans, such as iterations of one or two weeks, there is less “time to fill”.

The idea behind Student Syndrome is that when people are given a deadline, they tend to wait until they have nearly reached the deadline before starting work. Again, the two-week iterations on agile projects mean the deadline is never far away, and this helps keep people focused.
AGILE PLANS

Agile planning varies from traditional planning in three key ways. On agile projects:

1. **Trial and demonstration uncover true requirements, which then require replanning.** Instead of creating very detailed specifications and plans, agile projects build a prototype to better understand the domain and use this prototype as the basis for further planning and elaboration.

2. **Agile planning is less of an upfront effort, and instead is done more throughout the project.** Agile methods recognize that the level of risk and uncertainty on knowledge worker projects can make heavy upfront planning problematic, so they spread the planning out more evenly throughout the project’s life cycle.

3. **Midcourse adjustments are the norm.** When aiming at a moving target that is not following a predictable path, we need more of a guided-missile approach, making a lot of mid-flight adjustments to ensure we reach our target.
AGILE PLANS

AGILE CHARTERS

A project charter describes the project’s goal, purpose, composition, and approach, and it provides authorization from the sponsor for the project to proceed.

Agile charters acknowledge that scope may change and that initially some aspects of a project may be unknown. Therefore, rather than trying to fully specify the scope, agile charters characterize the goals envisioned for the project. They describe the processes and approaches that the team should use to iterate toward the final product, as well as the acceptance criteria that will be used to verify the project outcomes.

The process of chartering helps align stakeholders with the project.
AGILE PLANS

BUSINESS CASE DEVELOPMENT

The business case describes why the sponsoring organization should undertake the project. When there are multiple projects competing for a limited supply of sponsorship funding, the organization will assess the business cases to determine which projects to undertake.

The amount of detail in an agile business case varies from project to project. For a small agile project, there may not be a distinct business case document. For large agile projects, we may see a document that includes:

- Project overview
- Anticipated cost
- Anticipated benefits
- Business models and indexes
- ROI assumptions and risks associated with the project
- Risks of not undertaking the project
- SWOT/PEST analysis
- Recommendations
AGILE PLANS

ITERATION AND RELEASE PLANNING

Agile projects are divided into releases and iterations. An iteration is a short development period, typically two to four weeks in duration. A release is a group of iterations that results in the completion of a valuable deliverable on the project. So a project has one or more releases, and a release contains one or more iterations.

Fig. 7: Project Broken into Releases and Iterations
AGILE PLANS

ITERATION AND RELEASE PLANNING CONT'D

RELEASE PLANNING

Releases are planned around delivering useful and valuable increments of functionality to the business or customer. A release may be *date driven* or *functionality driven*. Whatever drives the release, we need to determine the functionality that can be developed and turned over or delivered for the planned release.

STORY MAPS

Story maps can be useful ways to show release plans. Story maps allow us to lay out and group stories, first by dependencies and then by functionality. When using story maps to show release plans, we still need to add up the story point values with each release and check whether the plan is viable, given the team’s development capacity and the amount of time designated for the release.
ITERATION PLANNING

Iteration planning follows the same idea as release planning. In iteration planning, we ask the team to select user stories that the customer has indicated and are high-priority items and that can be developed, tested, and delivered within the iteration.