Transcript

Completing the Release Plan

Learning Objectives

After completing this topic, you should be able to

- *recognize the guidelines for splitting and combining user stories*
- *plan iterations based on calculations using velocity and iteration length*

1. Splitting and combining stories

Once you've estimated the sizes of user stories for a project, estimated the development team's velocity, and prioritized the user stories, you can assign stories to iterations and complete the release plan.

Most agile teams use iterations of fixed length, typically ranging from two to four weeks.

Ideally, a team should allocate between four and ten user stories per iteration, and each story should take from two to five days to develop.

User stories are initially developed during the requirements gathering phase of release planning. As the team becomes more familiar with a product during the estimation and prioritization processes, however, it may find that some user stories are either too large or too small.

Select each type of user story for more information.

Too large

User stories that are very large are also known as "epics." These stories will take longer than the ideal two to five days to develop, and tend to describe required functionality too broadly – making it difficult to break the stories down into discrete tasks or to estimate them accurately. Stories that are too large should be split to create multiple smaller stories.

Too small

User stories that are too small describe features that will be very quick and easy to develop. Breaking required product functions into too many user stories can make the planning and development process unnecessarily granular and complicated. So when possible, you should merge related, small stories to create larger stories.

An example of an epic story describes developing a full online e-commerce system.

Graphic

The example epic story is "As a customer, I must be able to review, choose, and buy products online, and track my purchases."

This could take longer than a full iteration to develop, and should be broken down into multiple, more detailed user stories, each describing discrete features that make up the larger system.

Graphic

The single epic is split into three smaller user stories – "As a customer, I must be able to review products online", "As a customer, I must be able to choose and buy products online", and "As a customer, I must be able to track my purchases."

Two small user stories might involve creating a log-in interface that prompts users to supply their usernames and passwords, and building the functionality that actually verifies users' credentials.

Graphic

Two examples of user stories are "As a user, I must be able to access a user-friendly login screen" and "As a manager, I must be able to guarantee that only valid users can log on to the application."

If the team estimates that each of these stories will take less than two days to complete or that they're closely related to the same functionality, it's preferable to combine them into a single story.

Graphic

The two stories are combined to create the single story "As a user, I must be able to log on to the application using my username and password."

Question

Identify an appropriate guideline for combining user stories.

Options:

1. Aim to ensure that each user story will take 10 to 14 days to develop
2. Combine stories however necessary to ensure that they're all of the same length
3. Combine small, related stories, each estimated to require less than two days to develop
4. Combine related stories whenever possible so that each story fully describes required functionality

Answer

Option 1: *Incorrect. An agile team typically aims to develop 10 to 14 user stories per iteration, given an iteration length of two weeks. A single user story shouldn't generally take longer than two to five days to complete.*

Option 2: *Incorrect. You can include user stories of various lengths in each iteration. However, it's best to ensure that each user story will take approximately two to five days to complete.*

Option 3: *Correct. It's recommended that you combine small stories, provided they're related in some way. Ideally, each story should take between two and five days to develop, with between four and ten user stories handled in each iteration.*

Option 4: *Incorrect. You should combine related stories only if the stories are too small on their own and will take less than two days to develop. Stories that are too large are difficult to break down and estimate, and should be split.*

Correct answer(s):

3. Combine small, related stories, each estimated to require less than two days to develop

Splitting user stories, also called *disaggregation*, is usually more difficult than combining stories, and requires more discretion on the part of the team. In general, you can split user stories based on data or operation boundaries, cross-functional features, non-functional performance requirements, or the priorities of associated sub-stories.

Select each way to split user stories for more information about it.

Data boundaries

One way a team can split a large user story is according to the types of data involved.

For example, consider the user story "As a user, I want updates on stock exchange information." A team might split this story into three smaller stories – "As a client, I want to receive constant stock price updates," "As a client, I want to be able to track orders for stock I buy or sell," and "As a user with administrator privileges, I want to enable RSS feeds."

Operation boundaries

You can split large user stories according to the operations that are performed as part of the story, using operation boundaries. One efficient way of doing this is to use the Create, Read, Update, and Delete, or CRUD, approach. With this approach, you split a story based on whether it involves adding, changing, or removing something.

For instance, you can split the user story "As a client, I want to buy or sell orders for stock" into two smaller stories based on whether data is added or removed – in this case stock that's bought or sold. The two stories you create as a result might be "As a client, I want to buy orders for stock" and "As a client, I want to sell orders for stock."

Cross-functional features

When developing a software application, there are typically cross-functional features that don't operate in isolation. You can use these cross-functional features as a guide to where to split large user stories.

For example, if an application behaves differently depending on the privileges of the current user, you can split the story according to these different user privileges. You could split a user story such as "As a user with administrator privileges, I want to enable RSS feeds" into two smaller stories – one relating to users with administrator privileges, and another for regular users.

Non-functional performance requirements

A team may occasionally find it necessary to include non-functional performance requirements – user stories that don't directly relate to a particular customer requirement, but that are necessary if a product is to provide the level of performance that the customer expects. You can use these non-functional requirements as a measure for splitting user stories.

For instance, the user story "As a user with administrator privileges, I want to enable RSS feeds" can be split by separating its functional requirements from those that are non-functional. The resulting user stories can then be "As a user, I want access to an RSS service provider to receive RSS feeds" and "As an RSS feeds user, I want to log on using administrator privileges."

Sub-stories of mixed priority

You may occasionally find that a user story is unnecessarily large because it contains sub-stories of mixed priority. You can then split such large user stories by removing the components that relate to a different priority level and deal with those components as separate user stories.

Consider the user story "As a client, I want to log onto a server containing NASDAQ data to receive constant stock price updates." This story can be split into at least two smaller user stories with varying priority levels, like "As a client, I want access to NASDAQ data" and "As a client, I want to receive regular stock price updates."

Question

What factors is it appropriate to use as the basis for deciding how to split large user stories?

Options:

1. The total duration of a project
2. The different types of data they involve accessing or using
3. The different types of operations they involve
4. The estimated costs of developing the stories
5. The different priority levels of associated sub-stories

Answer

Option 1: *Incorrect. The total duration of a project won't help you decide how to split a large user story. However, a story should be split if it can't be developed within a single iteration, which typically lasts for a fixed period ranging from two to four weeks.*

Option 2: *Correct. A development team can split large user stories according to the types of data involved in the operations they describe.*

Option 3: *Correct. You can split large user stories according to the operations that are performed. You can use the CRUD approach to decide how to split a story along operational boundaries.*

Option 4: *Incorrect. Estimated costs won't help determine how best to split a large user story.*

Option 5: *Correct. If a user story contains sub-stories of mixed priority, you can group sub-stories with similar priority levels and use these to define new, smaller stories.*

Correct answer(s):

2. The different types of data they involve accessing or using
3. The different types of operations they involve
5. The different priority levels of associated sub-stories

2. Assigning user stories to iterations

Once a team has grouped and arranged user stories that are of similar and appropriate size, the next step is to assign the stories to particular iterations. The way this is done will depend on whether the planning for your project is feature-driven or date-driven.

Select each planning approach to learn more about it.

Feature-driven

When planning is feature-driven, a team undertakes to develop an agreed set of features, rather than to meet a fixed release date.

With a feature-driven approach, you use developers' estimates and the features that the customer requested as the basis for compiling an estimated completion date.

Date-driven

In a date-driven project, the team agrees to meet a fixed release date. It works with the customer to negotiate which features and how many story points it can reasonably develop by the specified deadline.

Because the deadline is fixed, you calculate approximately how many features the team will have to complete per iteration. Within budgetary constraints, you should ensure that the team is large enough to achieve the velocity required to finish the project on time.

For a feature-driven project, the easiest way to determine what will be developed in each iteration is to go back to the user story map and reflect on how you've grouped user stories into themes. This is because themes help identify which user stories can be grouped so that each iteration will result in a release of functioning software, or a working product feature, for review.

You should also generally focus on including stories of high priority in the earliest iterations.

When completing the release plan for a feature-driven project, you need to estimate a release date. To do this, you first total all the story points of the required features and then divide them by the team's velocity. This tells you the number of iterations the team will require to complete the required development.

You then multiply this number by the number of weeks per iteration to calculate the project completion date.

For example, 300 story points divided by 20 points per iteration gives you 15 iterations. Multiplying this value by 4 – given the use of four-week iterations – gives you 60 weeks, fifteen months. You can then add fifteen months to the start date for the first iteration to estimate a final release data.

Question

You're currently compiling the release plan for a feature-driven project. You've already determined that the team will need to complete 54 story points to develop the agreed features, and that the team's velocity is roughly 9 story points per two-week iteration.

How many iterations will the team require to finish the project?

Options:

1. 12 iterations
2. 18 iterations
3. 8 iterations
4. 6 iterations

Answer

Option 1: *Incorrect. The team requires only six iterations. You'd multiply this value by two only when determining the number of weeks required to complete the project.*

Option 2: *Incorrect. You need to divide the story points by the team's velocity – in other words, to divide 54 by 9 – rather than multiplying 9 by the number of weeks in each iteration.*

Option 3: *Incorrect. You need to divide the total number of story points by the team's velocity to calculate the required number of iterations. In this instance, 54 divided by 9 equals 6, so six iterations are required.*

Option 4: *Correct. To calculate the required number of iterations, you divide the total number of story points by the team's velocity. So in this case, you divide 54 by 9, which gives the result of 6.*

Correct answer(s):

4. 6 iterations

In a date-driven project, you calculate how many weeks are left before the fixed release date and then determine how many iterations of a particular length you can fit into that time.

For example, if the deadline is 24 weeks away and you're using three-week iterations, the team has eight iterations in which to complete the work.

You can then multiply the number of iterations by the team's velocity to calculate how many story points the team should aim to develop per release. For example, if a team's velocity is 15 story points per iteration, it can complete 120 points in the course of eight iterations.

You can then assign user stories to each iteration roughly in order of their priority, until you reach 120 story points in total. The stories you've managed to fit into this period are the ones the team will have time to develop.

Question

You're doing the release planning for a date-driven project. The project deadline is 52 weeks from now, and the team has agreed to use two-week iterations. Based on previous projects, you've calculated the development team's velocity at 12 points per iteration.

What's the maximum number of story points the team can complete by the deadline?

Options:

1. 624 story points
2. 1,248 story points
3. 312 story points
4. 8.67 story points

Answer

Option 1: *Incorrect. The team can complete 12 story points per two-week iteration and it can complete 26 two-week iterations during the project. So rather than multiplying 52 weeks by 12, you need to divide it by 2 and then multiply that resulting value – 26 – by 12.*

Option 2: *Incorrect. The team can complete 12 story points every two weeks, which means its velocity is 12 points per iteration. And if you divide the project time into the number of weeks in each iteration – 2 – you know the team has 26 iterations in which to complete the work. So rather than multiplying 52 weeks by 24, you need to multiply 26 iterations by the team's velocity – 12.*

Option 3: *Correct. Because there are 52 weeks in the project and the team uses 2-week iterations, you divide the number of weeks by 2 to calculate the number of iterations; in this instance 26. You then multiply the team's velocity – 12 – by the number of iterations – 26 – to get the result of 312.*

Option 4: *Incorrect. You've divided 52 by 12, and then multiplied the result by 2. Instead you should divide 52 by 2 – to determine how many iterations the team has in which to complete the work – and then multiply the result by the team's velocity – in this case 12.*

Correct answer(s):

3. 312 story points

Summary

After estimating the sizes of user stories, estimating team velocity, and prioritizing the user stories, you complete the release planning process by assigning stories to particular iterations. During this process, it may be necessary to combine user stories that are too small and to split stories that are too large. Generally each story should take two to five days to develop, and roughly ten to 14 stories should be assigned to each iteration.

The way a team assigns user stories to iterations depends on whether a project is feature-driven or date-driven. In a feature-driven project, the scope – or required features – are fixed but the release date is flexible. In a date-driven project, the release date is fixed but it's possible there won't be time to finish developing all user stories.