Affinity estimating

Estimation poker can be effective, but what if you have many user stories? Playing estimation poker for, say, 500 user stories could take a long time. You need a way to focus on only the user stories you must discuss to gain consensus.

TIP

Affinity estimating can be a fast and furious activity — the development team may choose to have the scrum master help facilitate affinity estimating sessions. To estimate by affinity, follow these steps:

1. **Taking no more than 60 seconds for each category, the development team agrees on a single user story in each of the following categories:**
   - Extra-small user story
   - Small user story
   - Medium user story
   - Large user story
   - Extra-large user story
   - Epic user story that is too large to come into the sprint
   - Needs clarification before estimating

2. **Taking no more than 60 seconds per user story, the development team puts all remaining stories into the categories listed in Step 1.**

   If you're using index cards or sticky notes for your user stories, you can physically place those cards into categories on a table or a whiteboard, respectively. If you split the user stories among the development team members, having each development team member categorize a group of stories, this step can go quickly!

3. **Taking another 30 minutes, maximum, for each 100 stories, the development team reviews and adjusts the placement of the user stories.**

   The entire development team must agree on the placement of the user stories into size categories.

4. **The product owner reviews the categorization.**

5. **When the product owner's expected estimate and the team's actual estimate differ by more than one story size, they discuss that user story.**

   The development team may or may not decide to adjust the story size.

6. **The development team plays estimation poker on the user stories in both the epic and the needs clarification categories.**

   The number of user stories in these categories should be minimal.

   Note that after the product owner and the development team discuss clarifications, the development team has the final say on the user story size.

SIZE	POINTS
Extra small (XS)	1
Small (S)	2
Medium (M)	3
Large (L)	5
Extra large (XL)	8

FIGURE 8-4: Story sizes as T-shirt sizes and their Fibonacci numbers.

Release Planning

A release is a group of usable product features that you deploy to the market. A release does not need to include all the functionality outlined in the product roadmap but should include at least the *minimal marketable features*, the smallest group of product features that you can effectively deploy and promote in the marketplace. Your early releases will exclude many of the medium- and low-priority requirements you identified during the product roadmap stage.

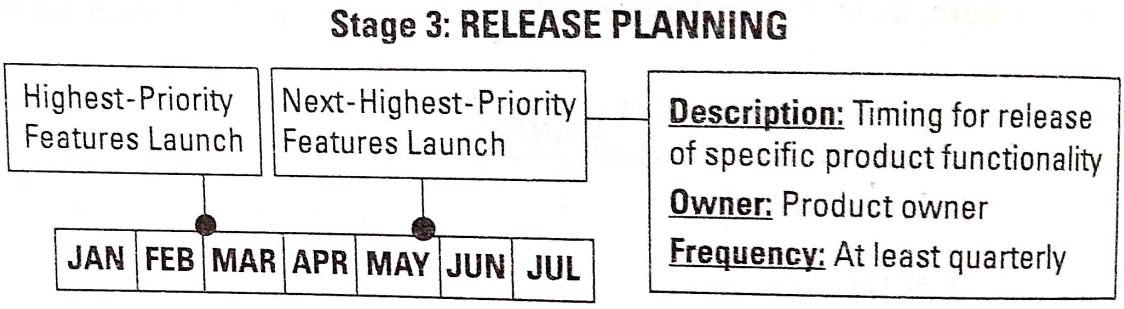

FIGURE 8-5: Release planning as part of the Roadmap to Value.

(Stages 1-3 are common practices outside of scrum)

The release plan contains a release schedule for a specific set of features. The product owner creates a release plan at the start of each release. To create a release plan, follow these steps:

1. **Establish the release goal.**

   The release goal is an overall business goal for the product features in your release. The product owner and development team collaborate to create a release goal based on business priorities and the development team's development speed and capabilities.

2. **Identify a target release date.**

   Some scrum teams determine release dates based on the completion of functionality; others may have hard dates, such as March 31 or September 1.

3. **Review the product backlog and the product roadmap to determine the highest-priority user stories that support your release goal (the minimum marketable features).**

   These user stories will make up your first release.

   We like to achieve releases with about 80 percent of the user stories, using the final 20 percent to add robust features that will meet the release goal while adding to the product's "wow" factor.

4. **Refine the user stories in your release goal.**

   During release planning, dependencies, gaps, or new details are often identified that affect estimates and prioritization. This is the time to make sure the

the release goal and scope with the product owner.

5. **Estimate the number of sprints needed, based on the scrum team's velocity.**

   Scrum teams use velocity to plan how much work they can take on in a release and sprint. *Velocity* is the sum of all user story points completed within a sprint. So, if a scrum team completed six user stories during its first sprint with sizes 8, 5, 5, 3, 2, 1, their velocity for the first sprint is 24. The scrum team would plan its second sprint keeping in mind that it completed 24 story points during the first sprint.

   After multiple sprints, scrum teams can use their running average velocity as an input to determine how much work they can take on in a sprint, as well as to extrapolate their release schedule by dividing the total number of story points in the release by their average velocity. You learn more about velocity in Chapter 13.

6. **Identify work necessary to release that can't be completed within a sprint. Plan a release sprint, if necessary, and determine how long it should be.**

   Some project teams add a *release sprint* to some releases to conduct activities that are unrelated to product development but necessary to release the product to customers. If you need a release sprint, be sure to factor that into the date you choose. You can find more about release sprints in Chapter 11.

   Some tasks, such as security testing or load testing a software project, can't be completed within a sprint, because the security or load testing environments take time to set up and request. Although release sprints allow scrum teams to plan for these types of activities, doing so is an anti-pattern, or the opposite of being agile. Your goal should be to complete all work required for functionality to be shippable at the end of each sprint.

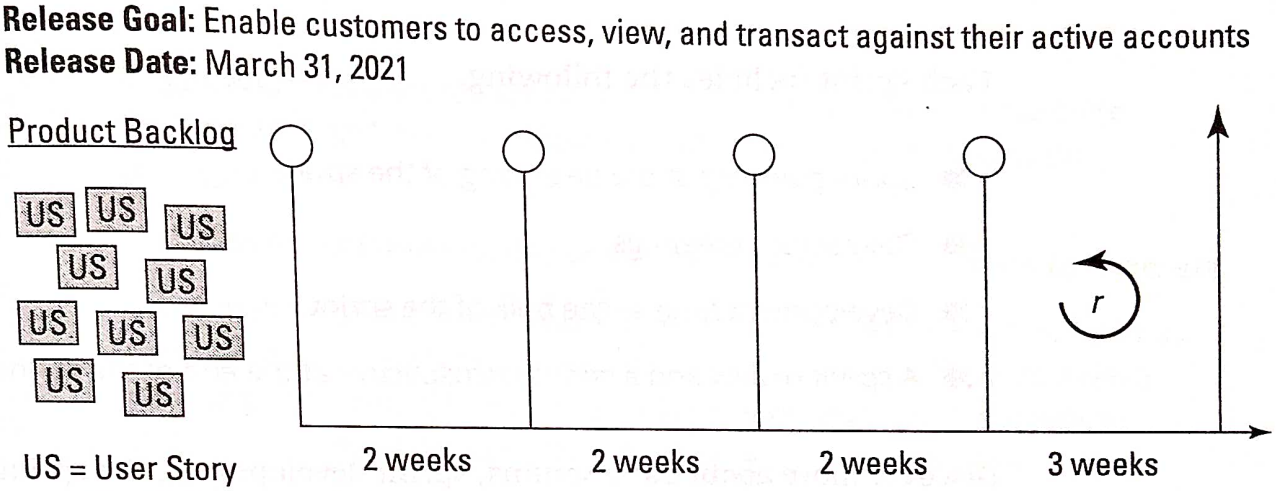

FIGURE 8-6: Sample release plan.