Story Pointing Guidelines

Introduction

Story point estimates will naturally vary from team to team and person to person, simply down to varying subjective opinions of the work being estimated. But we can set out some loose guidelines to help us reach a consensus. We aim to score stories based on the expected/understood complexity

Before Pointing - Does the team understand what's required?

Before anyone dives headfirst into estimating things, have you done the following:

1. The scope is well defined including if needed, mention of anything out of scope
2. Acceptance Criteria is populated
3. Any useful notes added on possible implementation or testing details that have been found from researching the story
4. Walkthrough of the story with the whole squad, preferably by whoever wrote it

Also, consider if there's any significant unknowns that you should play a spike or just investigate further before the story

Pointing Guidelines

Story points are a measure of complexity, as opposed to time. This helps us better compare the relative size of stories without worrying about individual expertise.

Points are assigned from the Fibonacci scale: 1,2,3,5,8... There's plenty of material to be found discussing why to use this scale, but put simply it helps distinguish the relative effort and prevent getting too bogged down; for example why bother trying to decide whether something is 5 or 6 points? the difference is so small its not worth discussion

Each item here is something to consider when coming up with a point estimate, it doesn't necessarily follow that you could check off each related item and total them up, but its a good place to start

1. How many code touchpoints are there? (eg a feature for the frontend might need changes to a page and a backend service)
2. How complex is the business logic?
3. Does a new app and its associated deployment pipeline need creating?
4. Will there be any changes in a "legacy" or brittle application, perhaps one that has low test coverage (We're looking at you Agg)
5. What kind of testing will be involved? coded acceptance tests, manual checks in an environment, checks across multiple browsers/devices, etc...

Whatever the estimate you reach, you should be happy that the story can be completed within a sprint, and if not, look at breaking it down to smaller stories.

If in doubt, cheat! (aka compare to other stories)

No two stories are the same, but there are similarities. creating an app and its deployment steps will always follow the same sequence, a frontend change will usually involve a similar range of logic, content, and styling. So look back at previous stories use them as your ruler for sizing new ones.

Reaching consensus

The whole squad should agree on the points for a story, which may involve some back and forth. Planning poker can be a useful way of starting things off. If disagreement cant be reached then that may be a sign the story isn't well defined, or perhaps needs breaking down into smaller stories

Further Reading

A quick search will turn up plenty of articles, blog posts, and more on this subject. But here's a decent starting point from Atlassian if you want to delve into this topic further: https://www.atlassian.com/agile/project-management/estimation