What is Planning Poker in Agile?

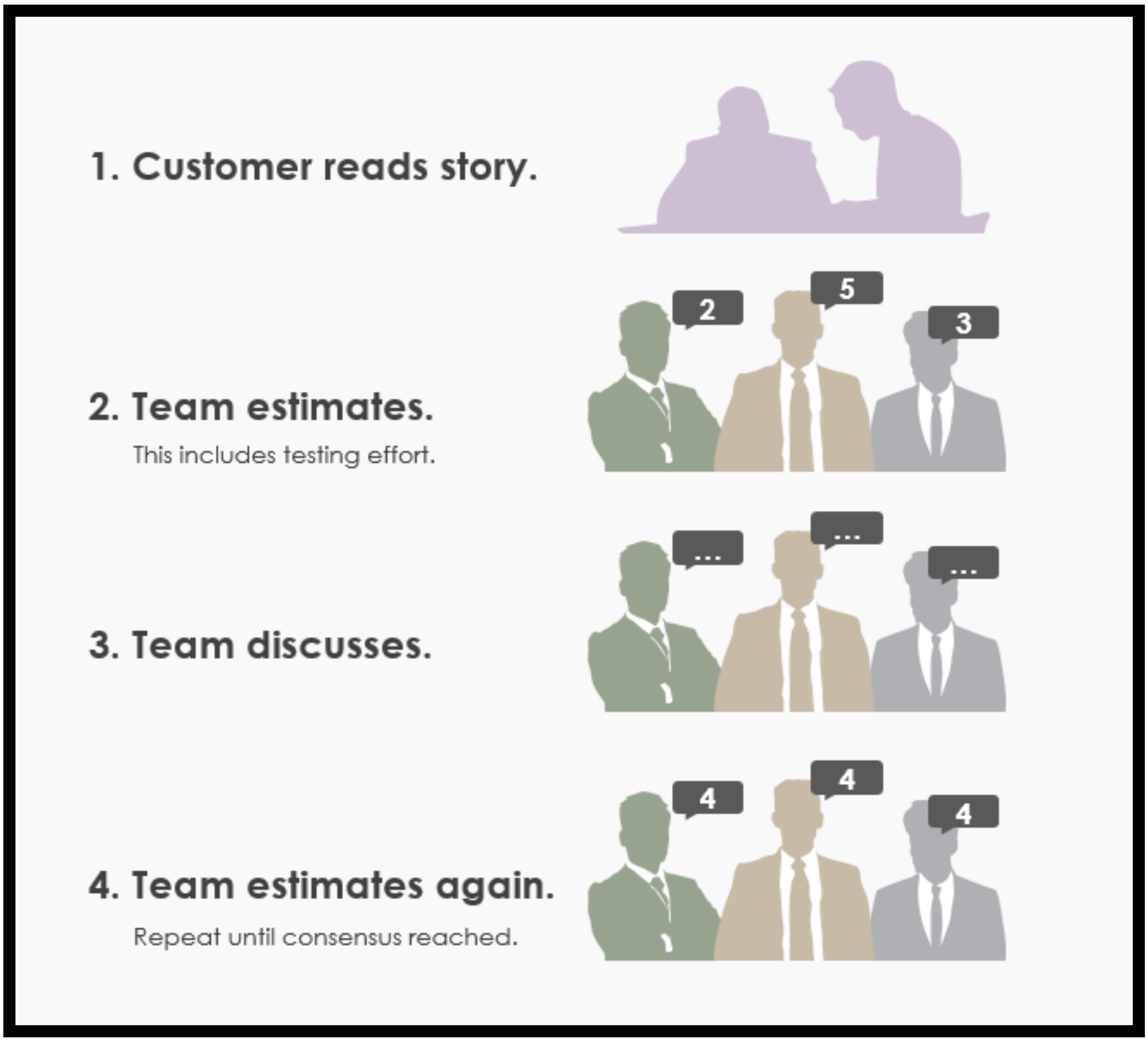
Effective estimation is one of the toughest challenges software developers face in their jobs. Regardless of team size, they need to define, estimate, and distribute work throughout a team. As teams get larger, it becomes even more important to build good habits around planning and estimating work. Lack of planning and estimating reduce confidence in a program, breaks down relationships between the team and the business, and makes development harder on everyone.

The Accuracy of Group vs. Individual Estimation

According to some study on the accuracy of estimation of effort between individual and group in an experiment for a software project. 20 software professionals from the same company individually estimated the work effort required to implement the same software development project. The participants had different background and roles and the software project had previously been implemented. After that, they formed five groups. Each group agreed on one estimation by discussing and combining of the knowledge among them.
Result – The estimates based on group discussions were more accurate than the individual estimates.

What is Planning Poker?

Planning poker (also known as Scrum poker) is a consensus-based, gamified technique for estimating, mostly used to estimate effort or relative size of development goals in software development.



Scrum Planning Poker

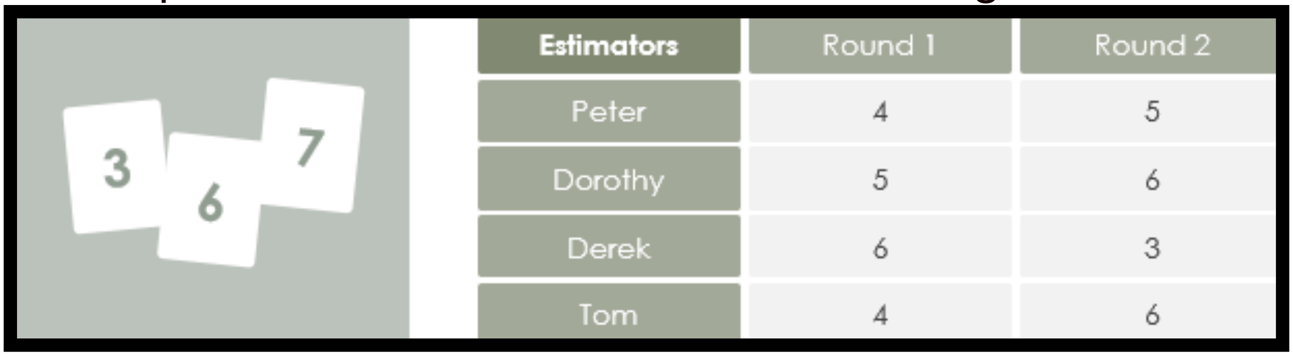
Steps for Planning Poker

To start a poker planning session, the product owner or customer reads an agile user story or describes a feature to the estimators.

For example:

“Customer logs in to the reservation system”

"Customer enters search criteria for a hotel reservation"
Team members of the group make estimates by playing numbered cards face-down to the table without revealing their estimate (Fibonacci values: 1,2,3,5,8,13,20,40)
Cards are simultaneously displayed
The estimates are then discussed and high and low estimates are explained
Repeat as needed until estimates converge



Estimators	Round 1	Round 2
Peter	4	5
Dorothy	5	6
Derek	6	3
Tom	4	6

Scrum Planning Poker steps

By hiding the figures in this way, the group can avoid the cognitive bias of anchoring, where the first number spoken aloud sets a precedent for subsequent estimates.

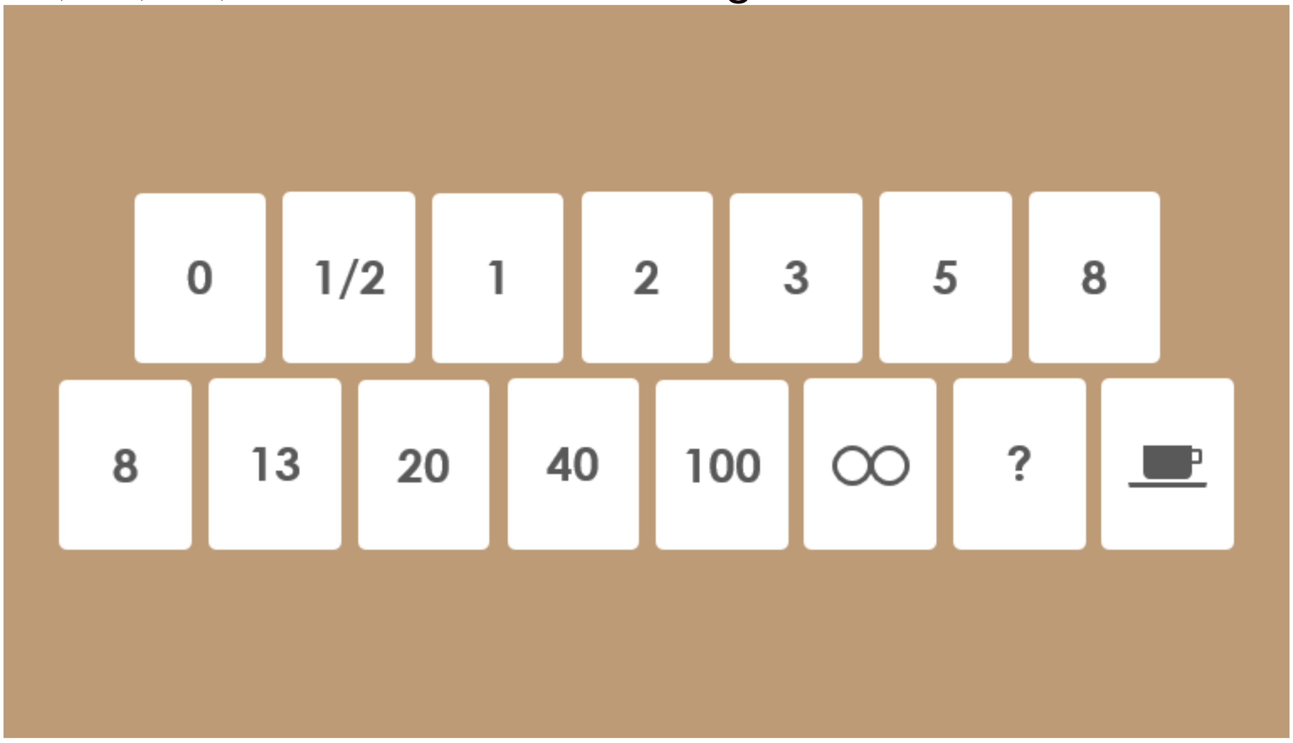
Agile Estimation – Relative vs Absolute

An estimate is nothing more than a well educated guess. We use all the knowledge and experience at hand to make a guess about the amount of time it is going to take. So instead of looking at every new work item separately, why not compare it to previously finished work items? It's easier for humans to relate to similar items than to guess the actual size of things anyway.
For example, is it closer to this really small thing? Or is it more like this normal sized item? Or is it really huge like that one piece of work we finished last month? Doing relative

estimates will not only reduce the amount of time spent on estimating work, it will also heavily increase the accuracy of the estimates.
Our brain is not capable of doing absolute estimates; we always put that new thing that we need to estimate in relationship to things we already know.

Fibonacci sequence and Planning Poker

Planning Poker uses of the Fibonacci sequence to assign a point value to a feature or user story. The Fibonacci sequence is a mathematical series of numbers that was introduced in the 13th century and used to explain certain formative aspects of nature, such as the branching of trees. The series is generated by adding the two previous numbers together to get the next value in the sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, and so on.
For agile estimation purposes, some of the numbers have been changed, resulting in the following series: 1, 2, 3, 5, 8, 13, 20, 40, 100 as shown in the Figure below:



Fibonacci Sequence and Planning Poker

The Interpretation of the point assigned to a poker card is listed in the table below:

Card(s)	Interpretation
0	Task is already completed.
1/2	The task is tiny.
1, 2, 3	These are used for small tasks.
5, 8, 13	These are used for medium sized tasks.
20, 40	These are used for large tasks.
100	These are used for very large tasks.
<infinity>	The task is huge.
?	No idea how long it takes to complete this task.
<cup of coffee>	I am hungry 🙂

Point vs Hour Value in Estimation

So why use **story points** instead of time values? Story pointing allows the team to focus on the complexity and time involved in delivering a piece of work. The team compares the new work against work they've already done. They compare the complexity of the new assignment against past challenges and rank the difficulty as well as the time required.

For example, we don't often account for "the cost of doing business." Meetings, email, code reviews, etc. with time values. But in reality, all these are necessary practices throughout in our daily life, but don't actually count as "work." Story points isolate the software development work from the associated logistic work items, so estimates using point based should more consistent than hour base approach.