Agile Estimating and Planning

Why Plans Go Wrong

Why Plans Go Wrong 5:33

Next Lesson →

2 Student syndrome

Definition

Starting a task at the last possible moment that does not preclude an on-time completion.

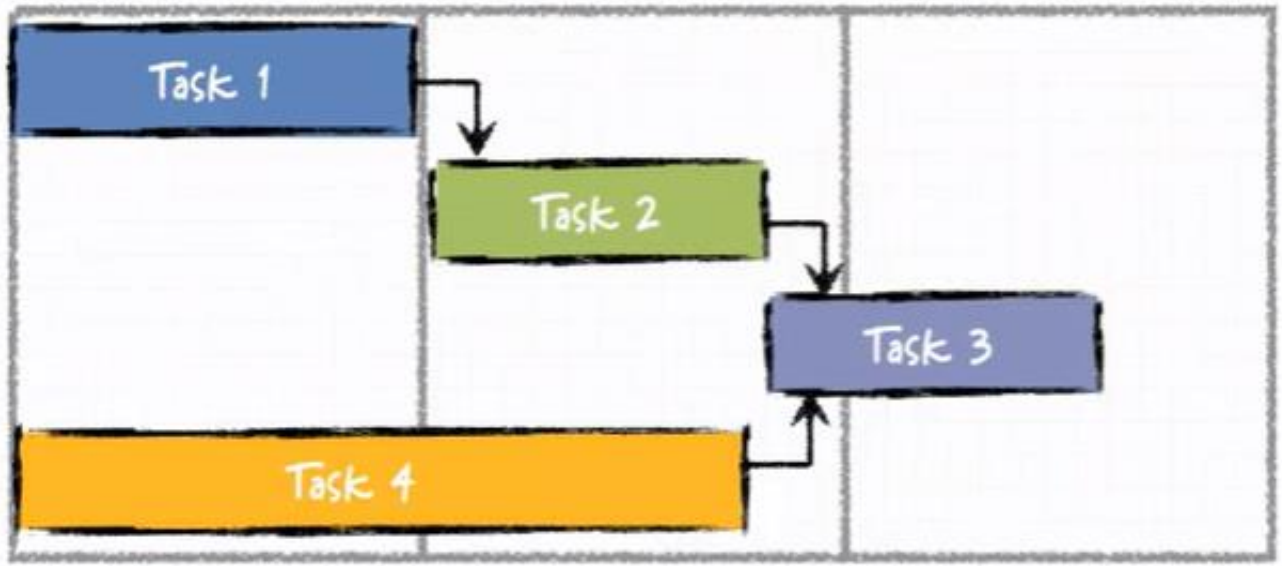
Student Syndrome Nature of Work

The estimate is based on this:

But we behave like this:

Project Delay

3 Lateness is passed down

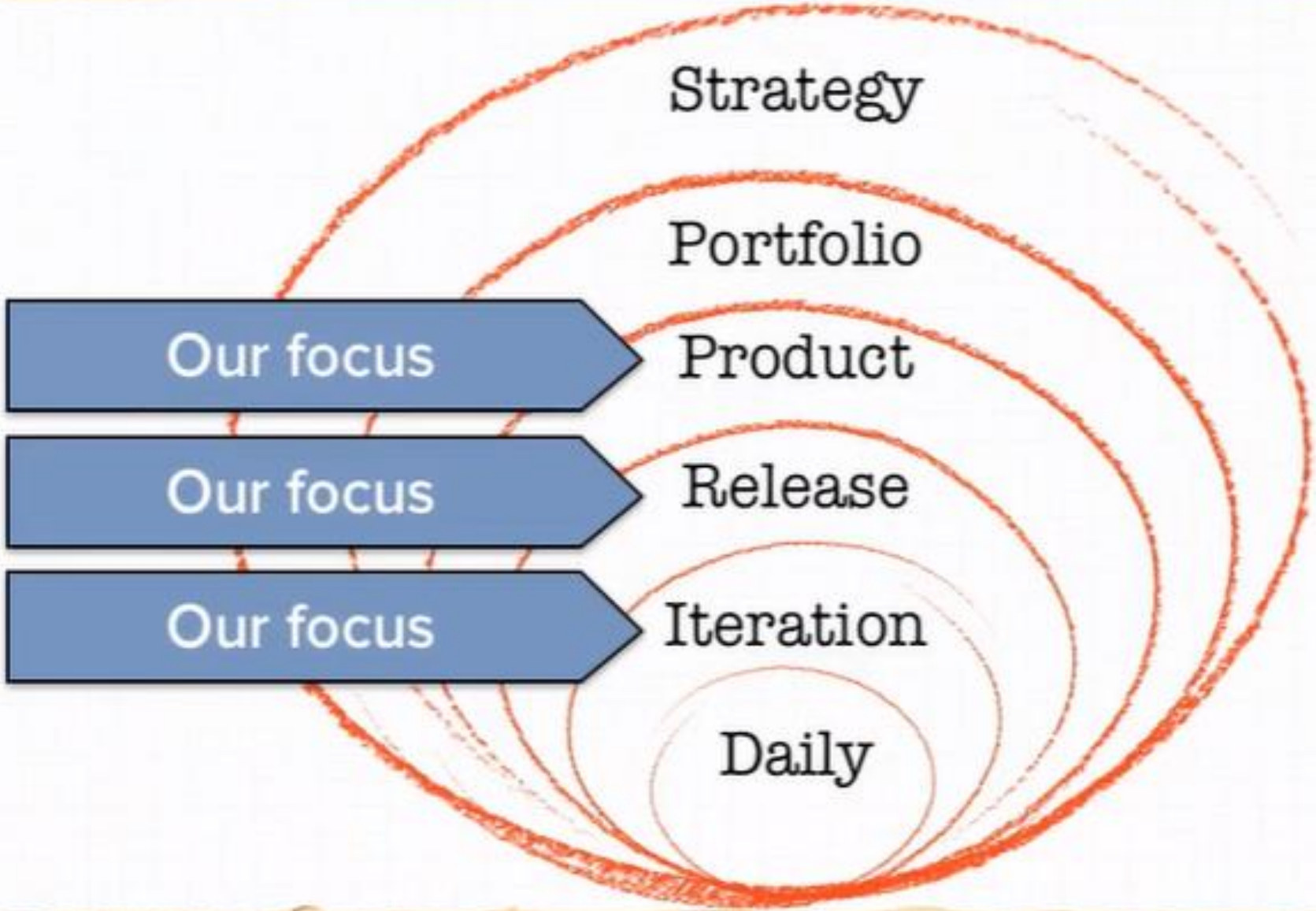


- Task 3 starts:
  - LATE if 1, 2 or 4 is late
  - EARLY only if 2 and 4 are early, and resource is available

What makes planning agile?

1. Is more focused on the planning than the plan
2. Encourages change with plans that can be easily changed
3. Is spread throughout the project

The Planning Onion



2 Different Types of Backlogs for Planning

Iteration Backlog

Code the ...	8
Test the ...	5
Confirm ...	2
Code a ...	12
Test the ...	4
Automate ...	8

Product owner prioritizes; team sequences

Story D

Story E

Story F

Iteration Plan

Story A

Story B

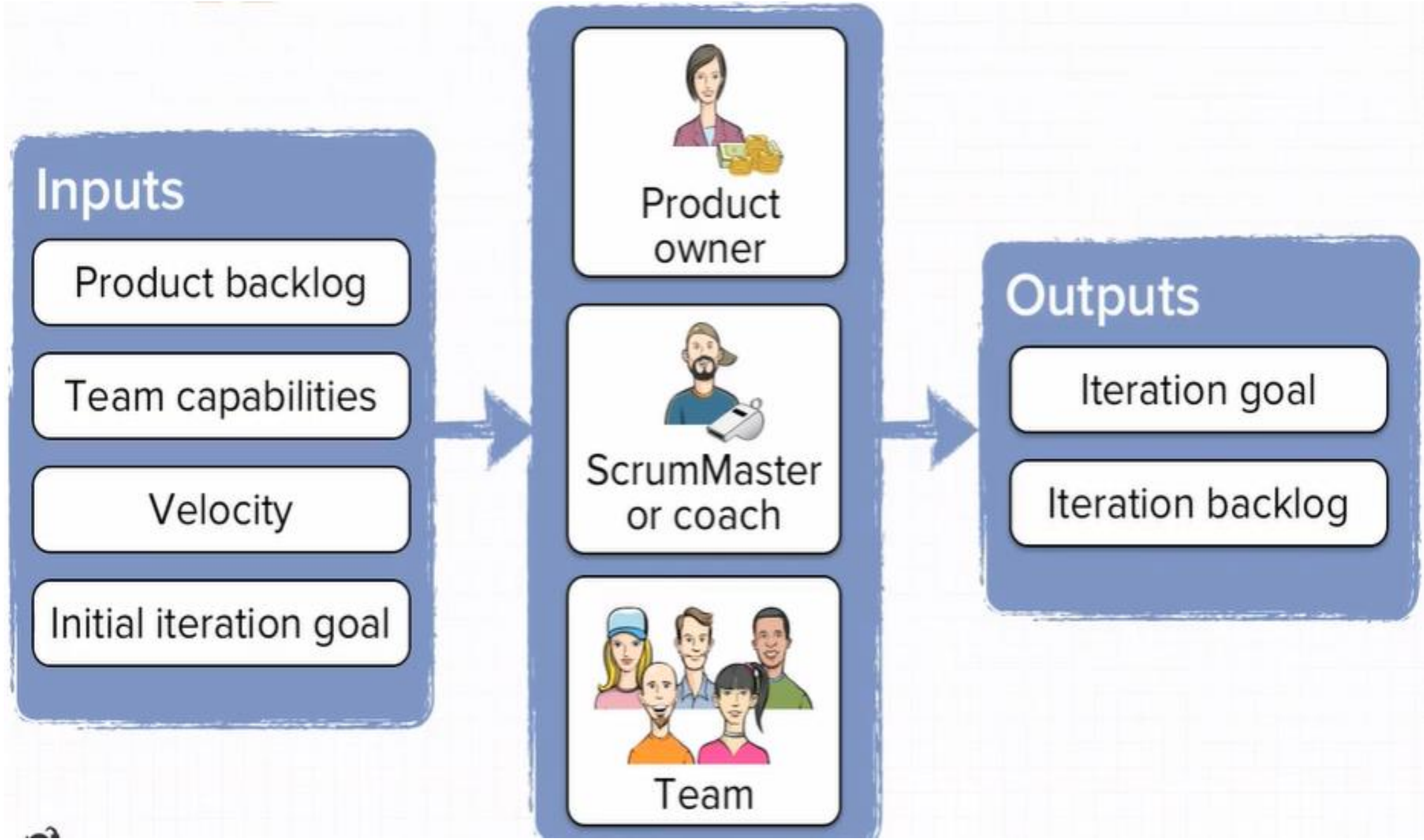
Story C

Story G

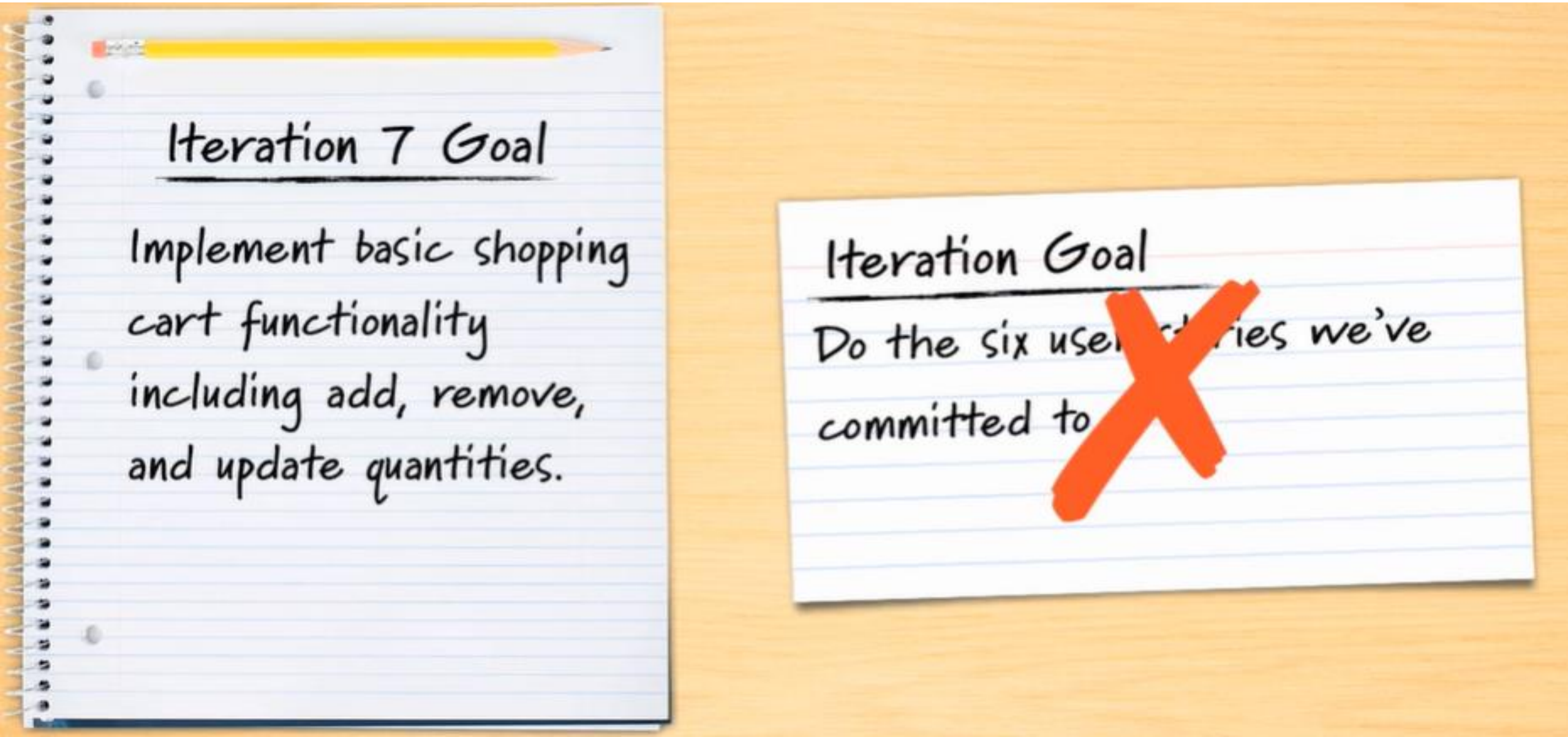
Some reasons to work slightly out of order

- Synergy between items
- Size of the work
- Dependencies
- Availability of skillsets

... and more


Inputs
Product backlog
Team capabilities
Velocity
Initial iteration goal
Product owner
ScrumMaster or coach
Team
Outputs
Iteration goal
Iteration backlog

Iteration Goal Good Example vs Bad Example



Iteration Backlog Example

An iteration backlog

	Task	Estimate
User story #1	Code the user interface	8
	Code the middle tier	12
	Test the middle tier	4
	Write online help	3
User story #2	Write the foo class	5
	Automate the ...	4
	Design a ...	1
	Test it	2

Iteration Planning Meeting – 4 Steps
(For Both Velocity Driven and Commitment Driven Planning)

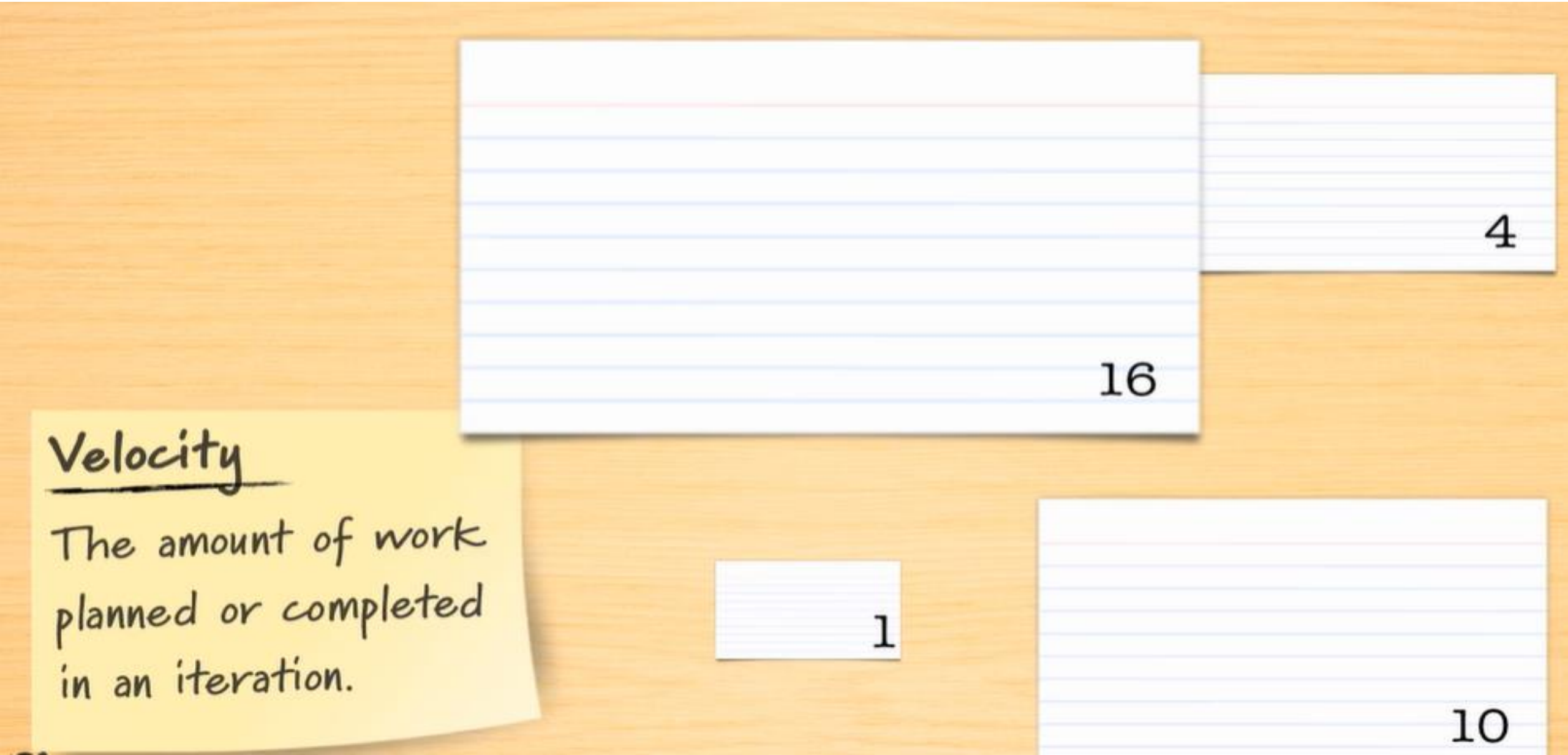
Iteration planning meeting

Four steps

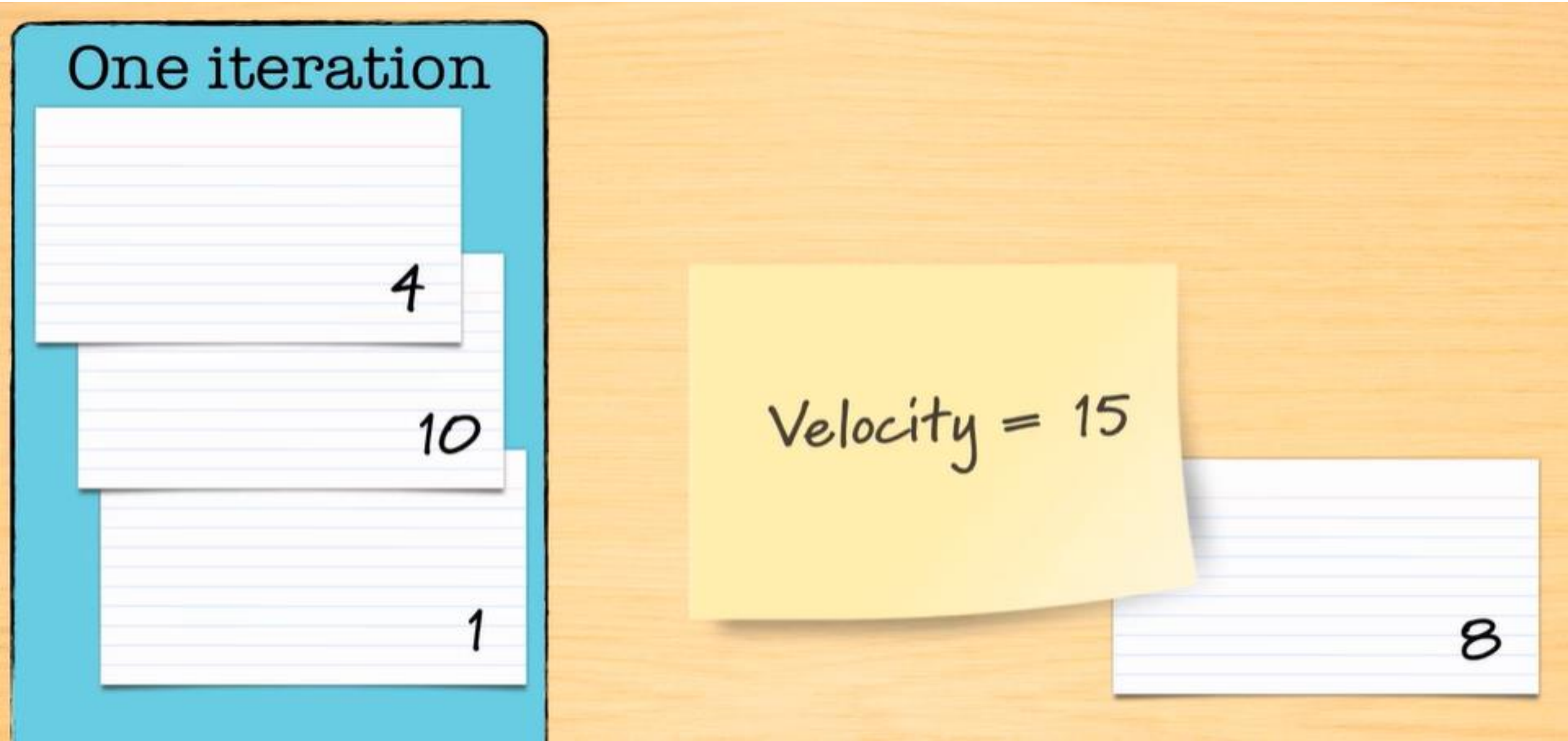
1. Determine capacity
2. Refine iteration goal
3. Define commitment
4. Acquire confidence

2 Ways to Plan an Iteration

Velocity Driven Iteration Planning



Calculating Team Velocity from most Recent Sprint



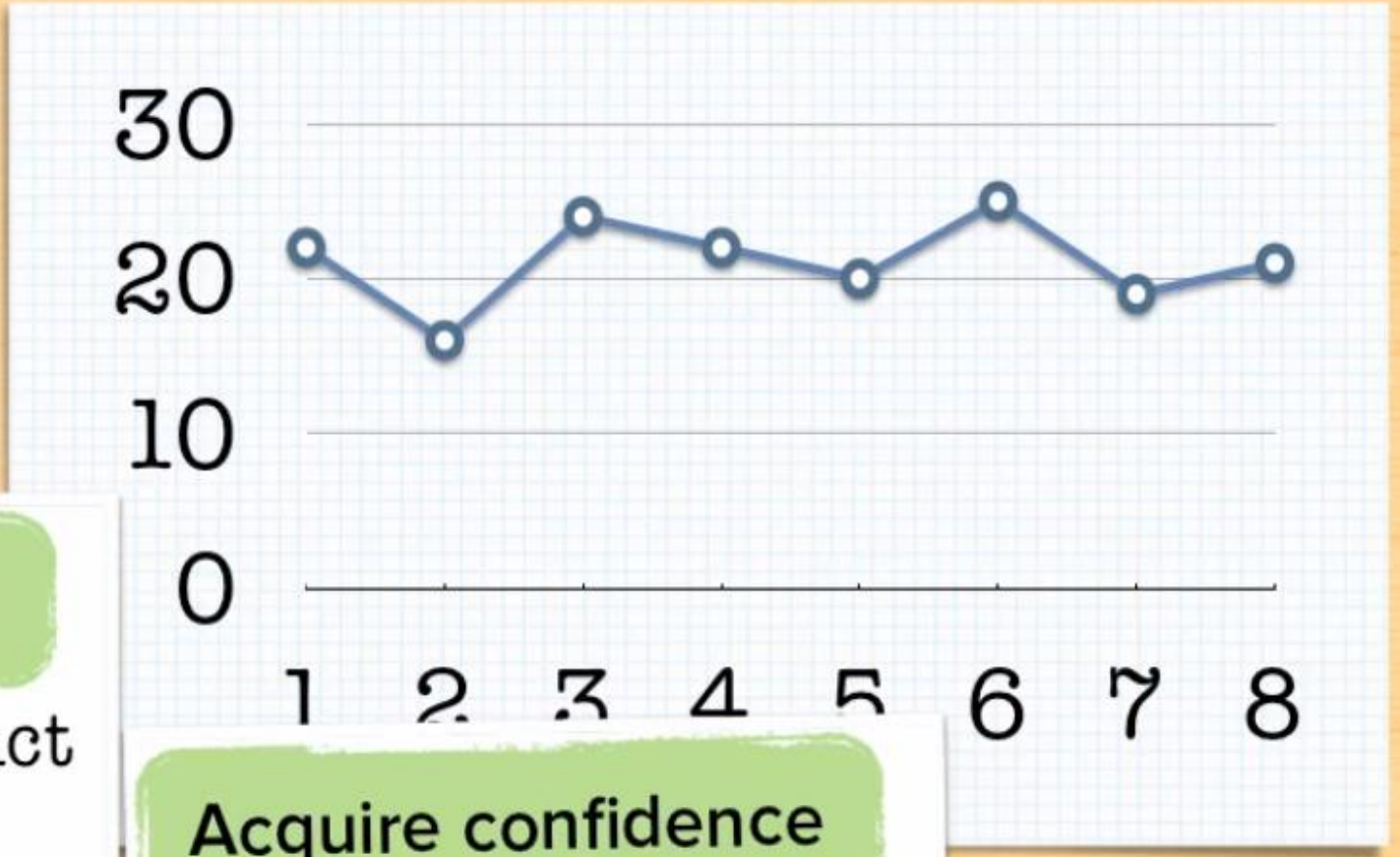
Velocity-driven iteration planning

Determine capacity

Look at historical average or most recent velocity

Define commitment

Select a set of product backlog items up to that velocity

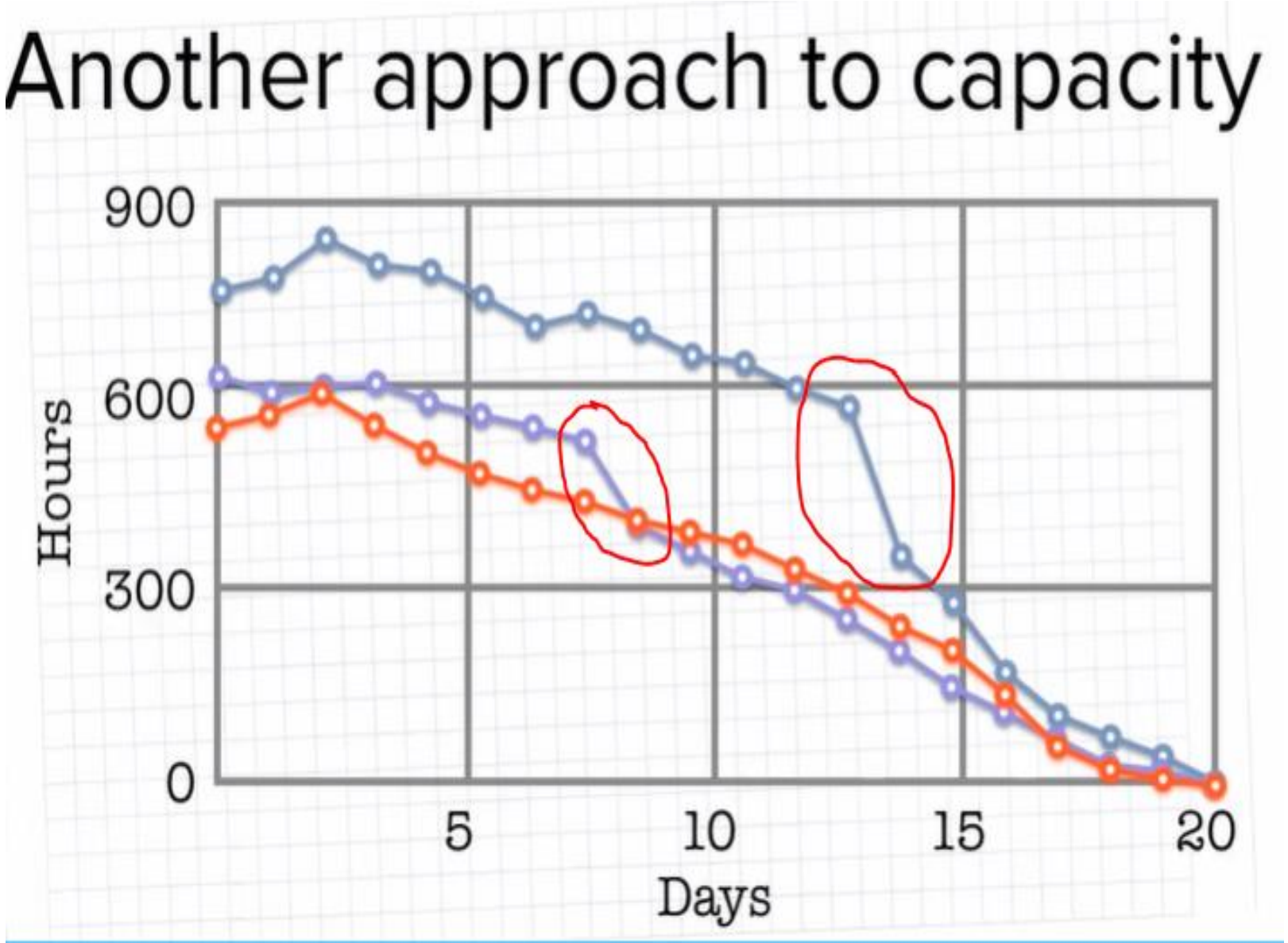


Acquire confidence

Create a list of tasks and decide if the work can be completed

Commitment Driven Iteration Planning

Observe First 3 Sprints Burndown Charts



Notice that the Team Dropped 200 hours of work in Day 13.

In Second Spring, Team decided to do 100 hours of Less work, but still dropped 100 hours of work in Day 8. **Things IMPROVED as they dropped less hours of work and earlier in the iteration.**

3RD Sprint went Ideally without big drops, Deriving from 3 Sprints that 580 hours of Work was the right amount of work for them to bring into an Iteration

The first iteration

Probably 15–20 hours

I have 26 hours available

20 wasn't bad. Maybe 22–25 this time.

That was too much. About 20 hours this time.

Subsequent iterations

Commitment-driven planning

Defining commitment & Acquiring confidence

1. Select a high-priority item on the product backlog
2. Decompose it into tasks
3. Estimate each task
4. Team members ask themselves:

   "Can we commit to this?"

   - If yes, repeat with another backlog item

A caution ...

- The purpose of iteration planning is to commit to a set of product backlog items.
- The purpose is not to come up with a list of tasks and hours.
- The tasks and estimates are a tool for determining what we can commit to.

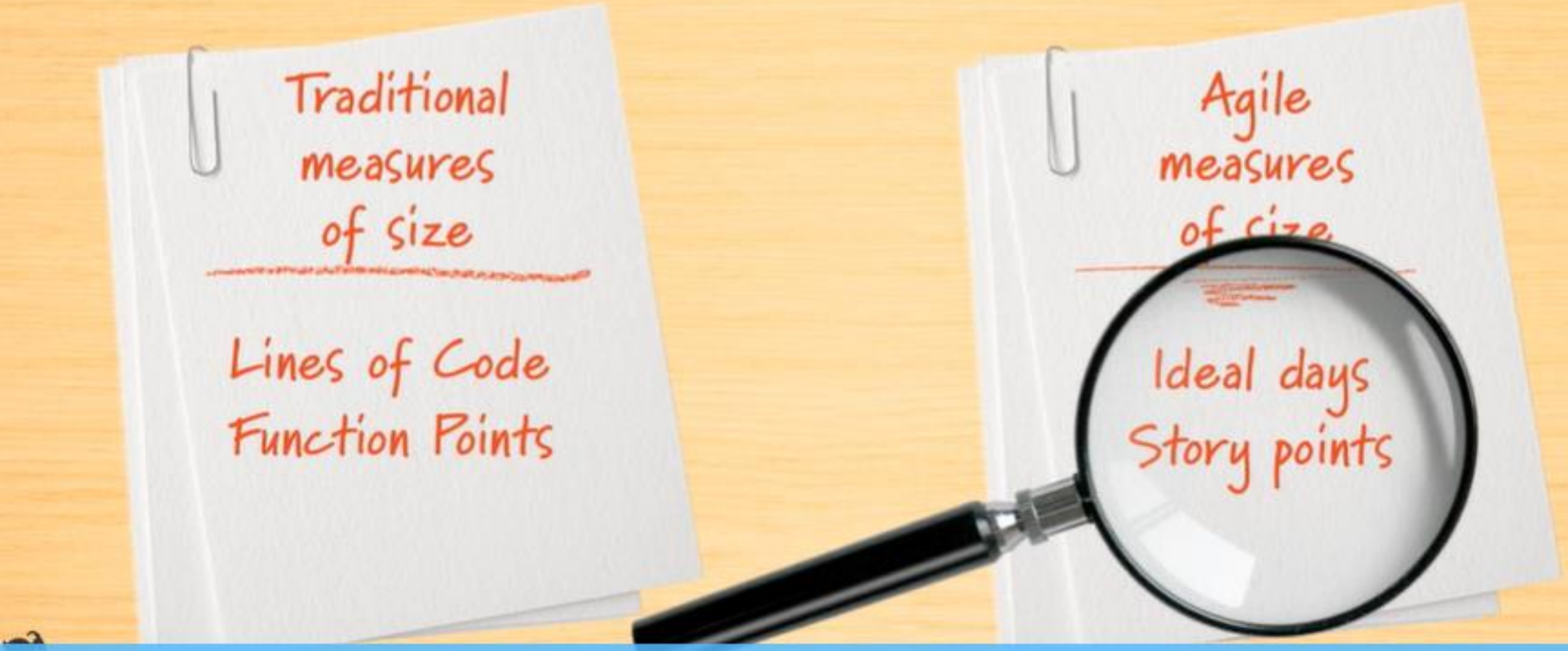
Iteration planning =
planning +
technical design +
product design

TRADITIONAL
MEASURE OF
DEVELOPMENT
WORK SIZE

→

function points =
f(external inputs,
external outputs,
external inquiries,
external logical files
external logical files)

Traditional and agile measure size differently



Ideal Time

How long something will take if:

- it's all you work on
- no one interrupts you
- and everything you need is available

Ideal time
Time spent doing the work

Elapsed time
Time from start to finish of the work

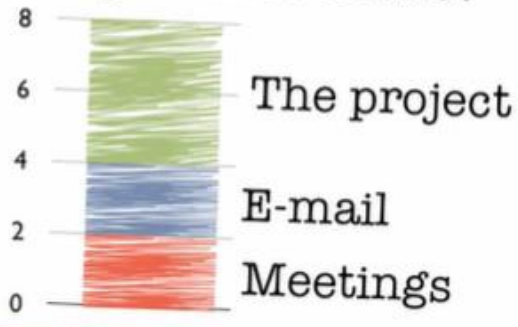
Elapsed time vs. ideal time

Ideally

- Monday has 8 hours
- Each week has 40 hours

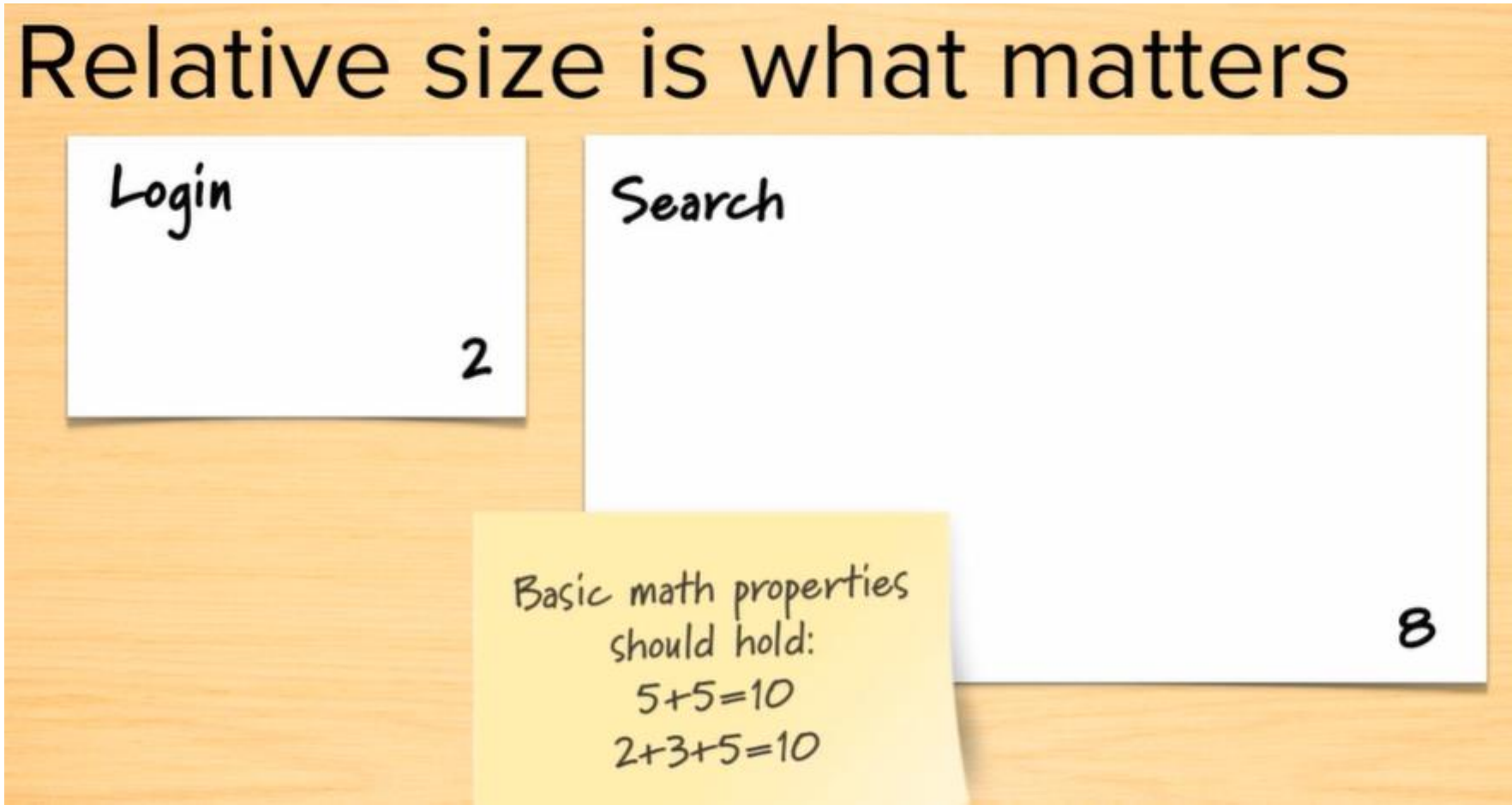
But instead...

Each day is more like:



Story points

- How long a user story will take (effort)
- Influenced by:
  - Complexity
  - Uncertainty
  - Risk
  - Etc.



Estimate size; derive duration

Backlog = 28 kilos

First iteration: We complete 7 kilos of work

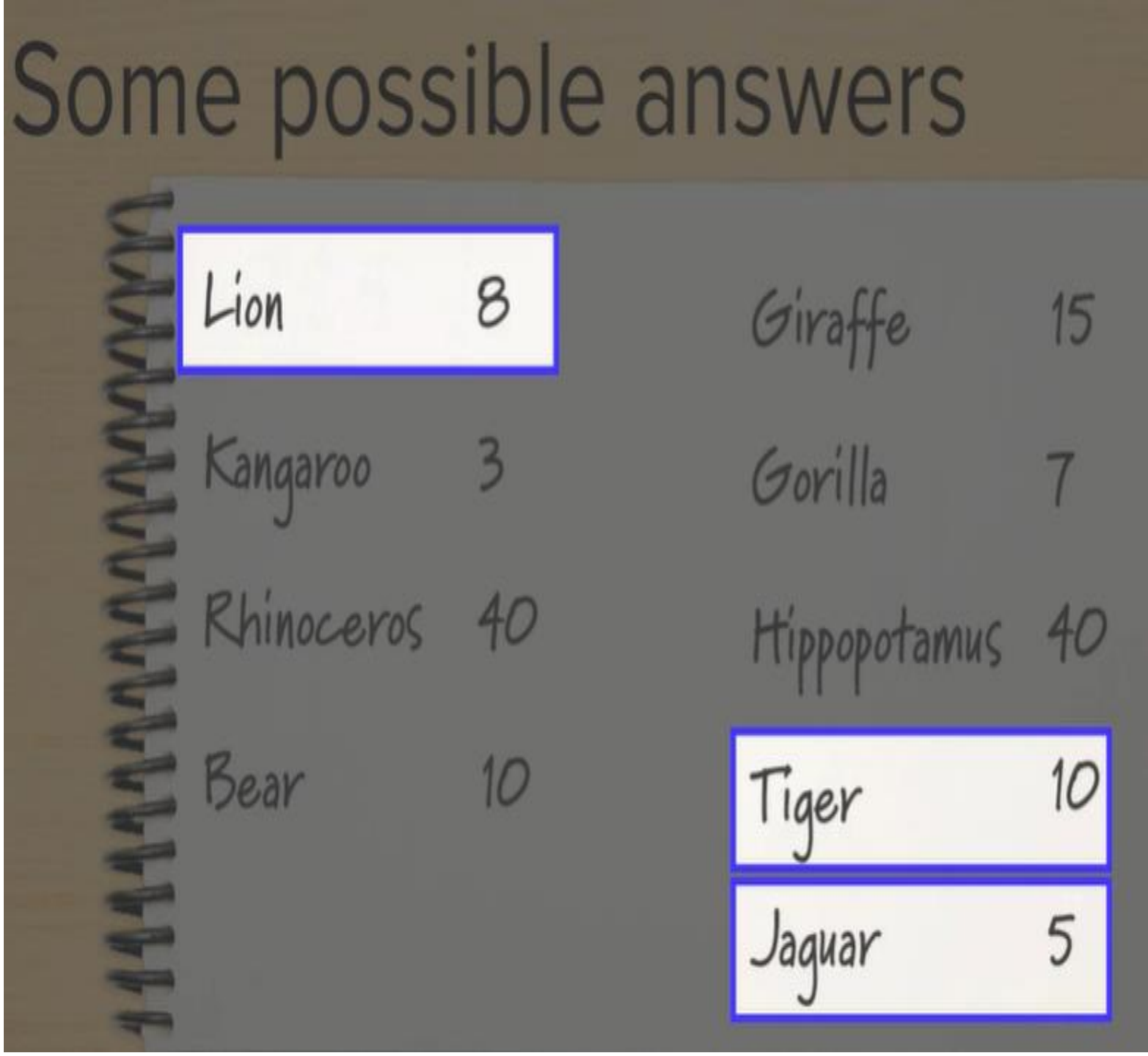
Velocity = 7

28 ÷ 7 = 4

Re-baselining

- Adjusting the estimates on the initially estimated items.
- Usually happens in the first 5-12 items

KEY POINT:
COMPARING SIZE OF TASKS THAT ARE SIMILAR



Reasons to favor:

Ideal Time

1. Easier to start with
2. Easier to explain to others
3. It's more palatable to management
4. Forces us to confront time-wasting activities

Story Points

1. Prevent incorrect conversions to calendar days
2. Promote cross-functional behavior
3. Points don't decay
4. Points are a pure measure of size
5. Time estimates are not additive
6. Points prevent unit confusion

Confusing units

Product Backlog	Iteration Backlog	
As a user...	he UI	12
As a user...	tests	8
As a user...	niddle tier	4
As a user...	e docs	6
As a user...	ate tests	5
		=35

Velocity is always the sum of units on the product backlog

IDEAL PRACTICE

What I Do

- Use story points if we can
- If not, start with ideal days
  - But emphasize relative estimating
  - Consider using a term that doesn't include "days"

Start with:

How long will this take?

But soon switch to:

How long will this take compared to _____?

Story points are an estimate of time (effort)

A team has estimated their product backlog in ideal hours, as shown at below. They have just completed the first iteration during which they completed one product backlog item, A. The iteration backlog shows the tasks that were necessary to deliver item A and the estimate for each. Which of the following statements are true?

☐ **Incorrect.** The correct answers were:
Velocity is 20.
Based on our single estimate of velocity, this project will finish in 5 more iterations.

Velocity is always based on the sum of the estimates on the items in the product backlog. In this case, the team completed only one item, A. Velocity is therefore 20. The sum of the estimates of the remaining items on the product backlog is 100, so our best estimate is that this project will take 100/20 = 5 more iterations.

Summary

Product Backlog	
Story	Ideal Hours
A	20
B	16
C	16
D	32
E	16
F	20

Iteration Backlog	
Task	Ideal Hours
A1	6
A2	5
A3	4
A4	8
A5	2

Gut feel

- A good reasonableness check
  - But not what we want to rely on all the time
- "Rapid cognition"
  - Malcolm Gladwell, *Blink*

Estimate by analogy

Analogy

A similarity between things on which you can base a comparison

Triangulate

3 points	Story A		
2 points	Story C	Story D	Story F
1 point	Story B	Story E	

Use the right units

- Use a set of numbers that make sense; I like:
  - 1, 2, 3, 5, 8, 13, 20, 40, 100

Use 0 and ½ if you'd like.

Planning Poker® steps

- Each estimator is given a set of cards containing only the numbers to be used
- Product owner reads a user story and it's discussed briefly
- Each estimator selects a card that's his or her estimate
- Cards are turned at the same time
- Discuss differences (especially outliers)
- Re-estimate until estimates converge

Two likely problems:

1. Product uncertainty
2. Technical uncertainty

Two good solutions:

1. Put the story aside until the uncertainty can be resolved
2. Use a range as the estimate

8 Reasons Why Planning Poker Works with Teams

1. Those who will do the work, estimate the work
2. Estimators are required to justify estimates
3. Most estimates are within one order of magnitude
4. Combining of individual estimates through group discussion leads to better estimates
5. Emphasizes relative estimation
6. Combining of individual estimates through group discussion leads to better estimates
7. Planning Poker is quick and fun

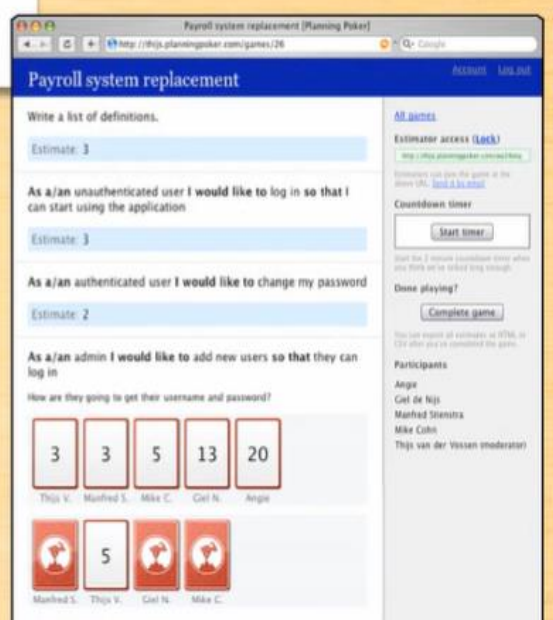
Anchoring

The tendency for estimates to cluster around an irrelevant value.

8

Planning Poker helps avoid anchoring

Planning Poker is online at www.PlanningPoker.com



New facts

- Customer thinks it will take 500 hours
- Customer isn't an expert; don't be influenced

New facts

- Customer thinks it will take 50 hours
- Customer isn't an expert; don't be influenced

https://www.mountaingoatsoftware.com/tools/velocity-range-calculator

Velocity Range Calculator

Velocities From Completed Iterations

30,31,32,33,34,35,40

velocity numbers, comma-separated

Planned Iterations *(optional)*

5

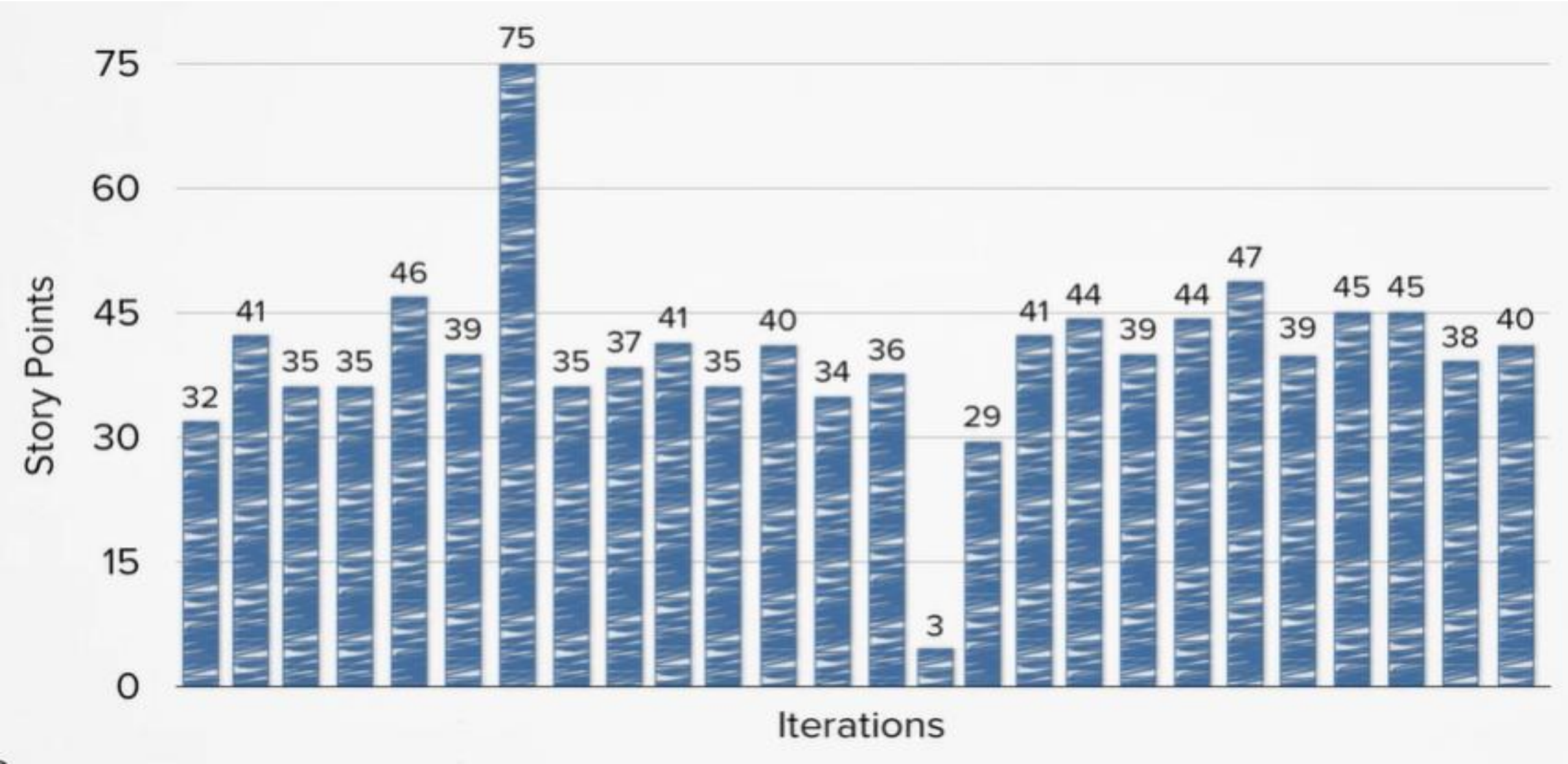
Calculate

For velocity values of **30,31,32,33,34,35,40**

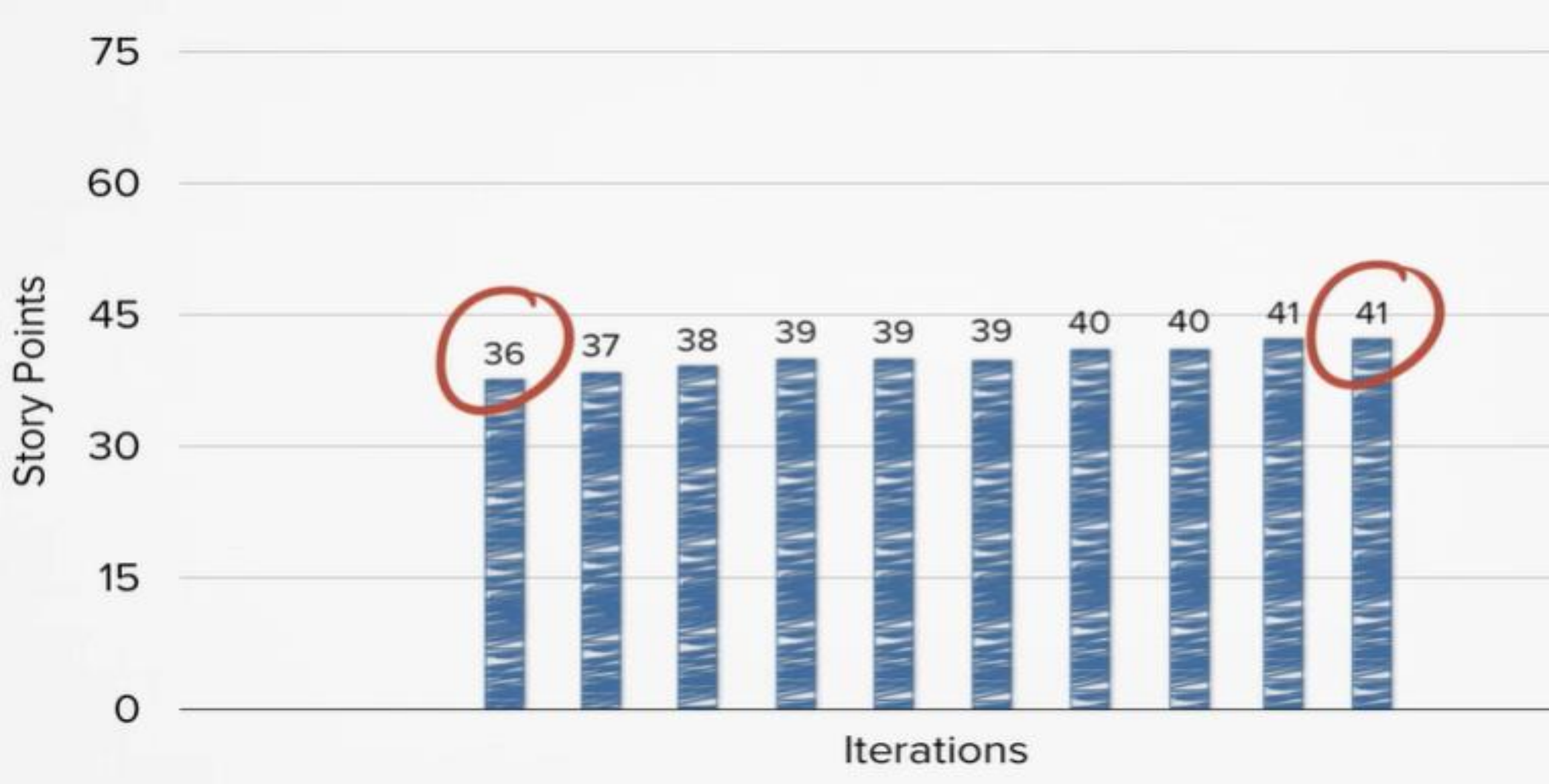
You have a median velocity of **33** and there is a 90% likelihood that your actual velocity will fall between **30** and **40**.

With 90% confidence you can expect to complete between **150** and **200** more units of work with a median value of **165**.

EXAMPLE OF VELOCITY ACROSS HISTORIAL ITERATIONS



STATISTICALLY TRUE AVERAGE – 90% CONFIDENCE INTERVAL – TRUE VELOCITY RANGE

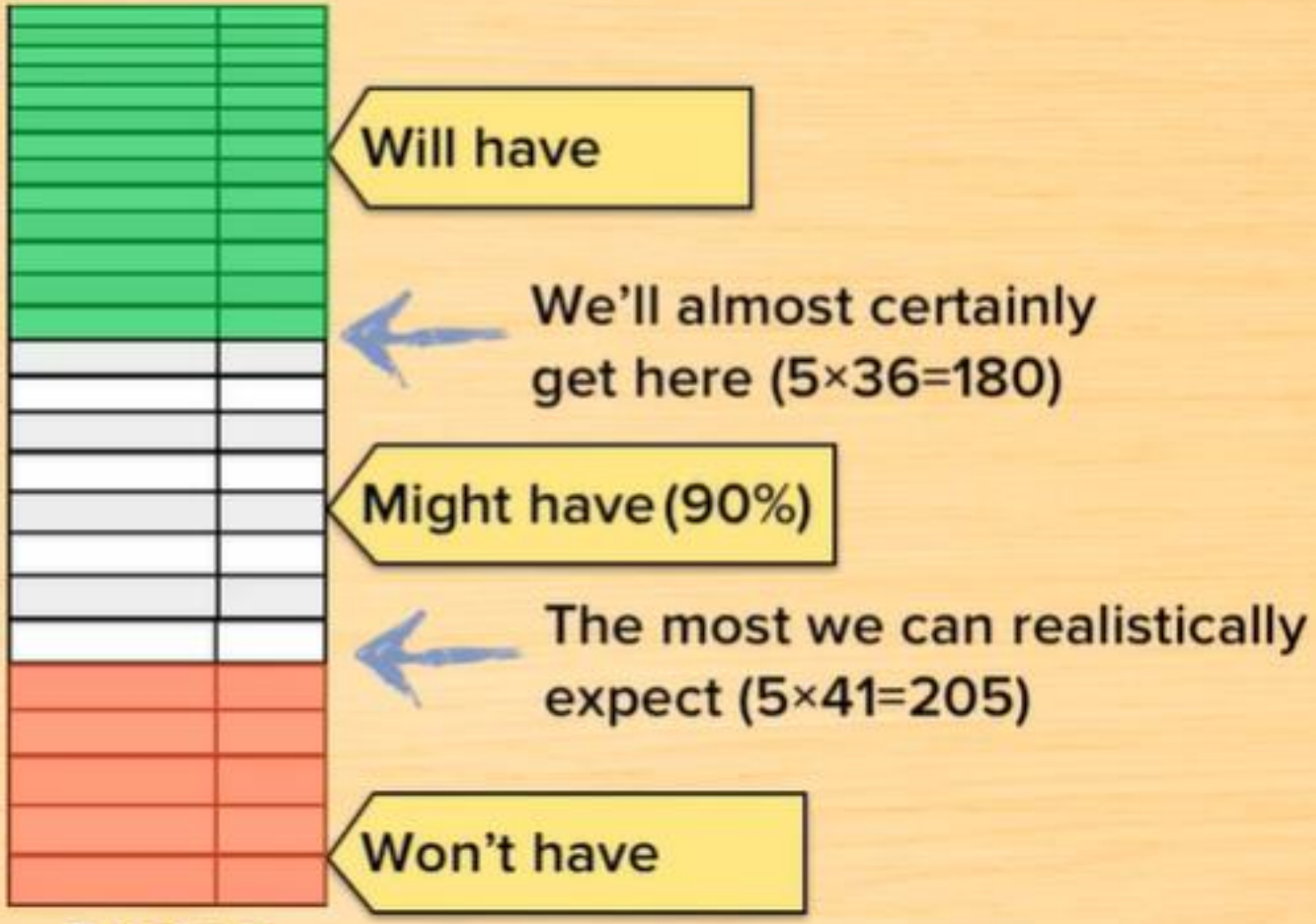


Release Planning with Historical Velocity Data 6:16

If you have this many iterations...	Then throw out this many from each end:
0–7	0
8–10	1
11–12	2
13–15	3
16–17	4
18–20	5
21–22	6
23–25	7
26+	8

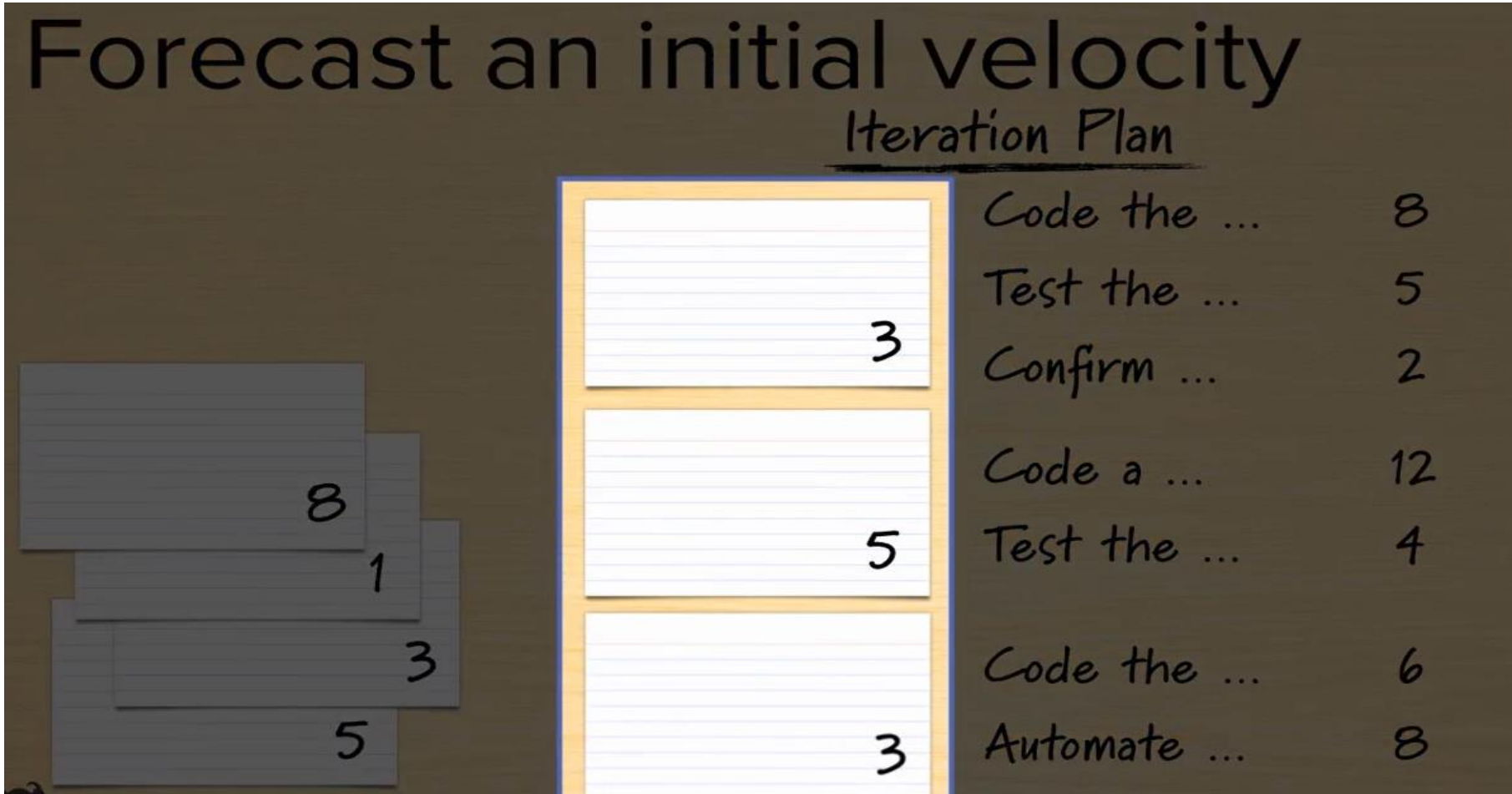
Putting the plan together

Assume there are five iterations left



Reasons for not having velocity data

1. The team is new to agile
2. The team is new

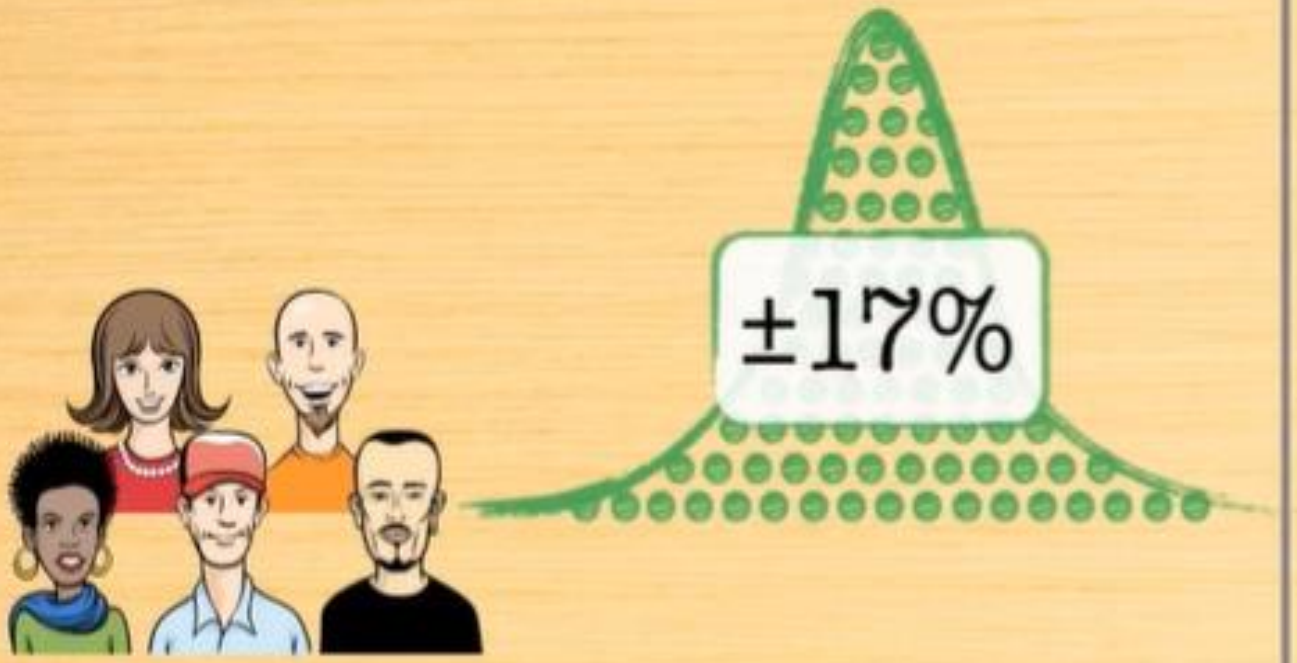


Turn the point estimate into a range

- If you don't have historical data, consider an educated guess

+/- 15% for a known team, domain, & technologies

+/- 50% if all that is unknown



? ± 18

Using relative standard deviation

Team A	
Iteration	Velocity
1	20
2	28
3	24
4	16
5	18
6	23
7	26
8	21

Mean = 22

Standard deviation = 4.0

$$\text{Relative standard deviation} = \frac{4.0}{22}$$

In Excel:

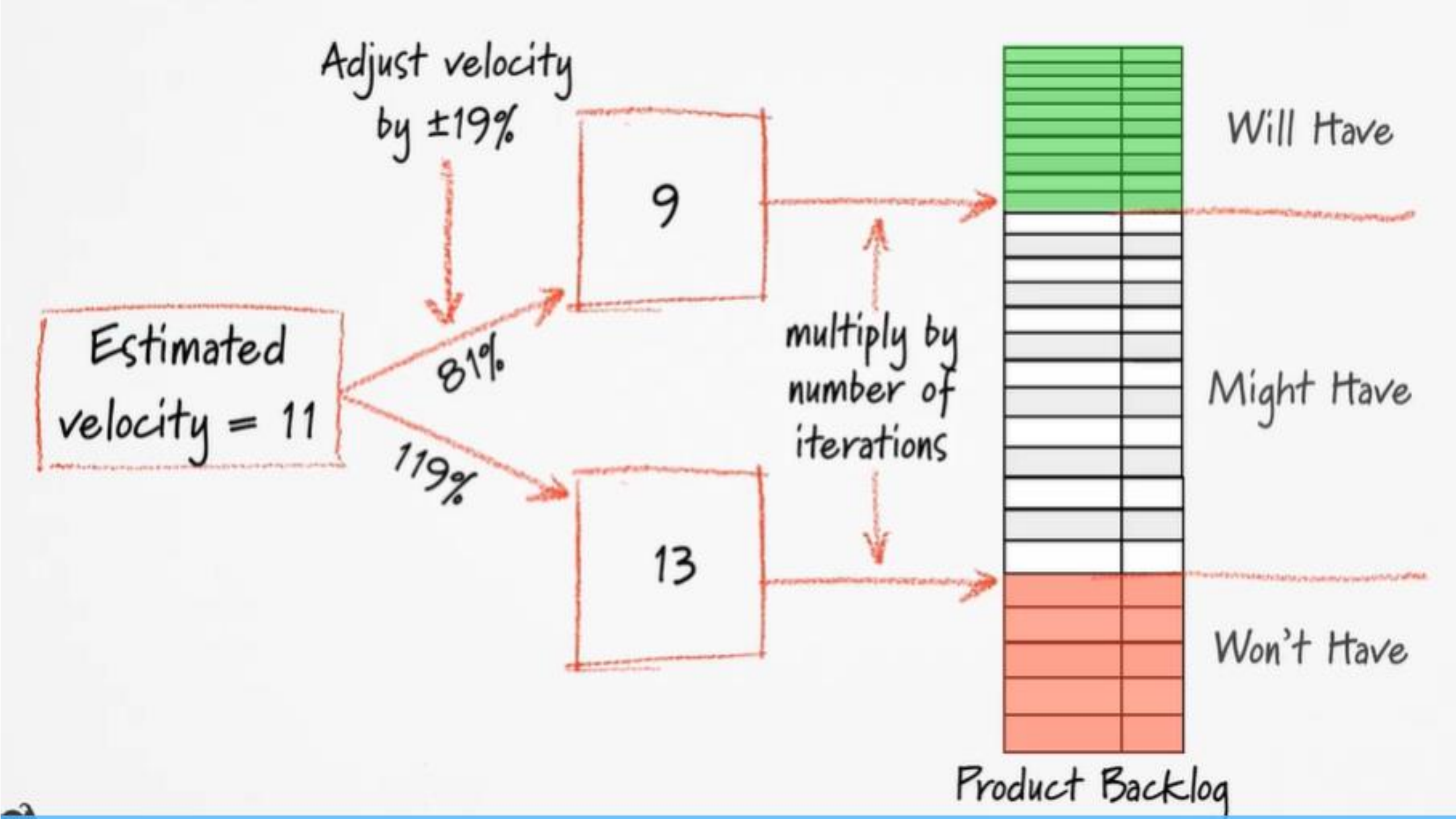
stdev( ) function

By hand:

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{N-1}}$$

Averaging across a department

	Mean	Standard Deviation	Relative Std. Dev.
Team A	22	4.0	18%
Team B	28	6.2	22%
Team C	45	9.3	20%
…	…	…	…
Average			19%


Adjust velocity by ±19%
9
Estimated velocity = 11
81%
119%
13
multiply by number of iterations
Will Have
Might Have
Won't Have
Product Backlog

Track velocity when size changes

Initial Team Size	New Team Size	Iteration +1	Iteration +2	Iteration +3
6	7	-20%	-4%	+12%
6	7	0%	-6%	+15%
7	5	-12%	-8%	+8%
8	6	-20%	-20%	+16%
7	8	-15%		

Impact of going from 6–7 people

Initial Team Size	New Team Size	Iteration +1	Iteration +2	Iteration +3
6	7	−20%	−4%	+12%
6	7	0%	−6%	+15%
6	7	−13%	−5%	
	Average	−11%	−5%	+13%
	Adjustment	0.89	0.95	1.13

Estimated work completed

$$= (V \times A_1 + V \times A_2 + V \times A_3 + V \times A_3 + V \times A_3)$$

$$= V \times (A_1 + A_2 + A_3 + A_3 + A_3)$$

$$= V \times (0.89 + 0.95 + 1.13 + 1.13 + 1.13)$$

$$= V \times (5.23)$$

about ¼ of an iteration

Estimated work completed

$= (V \times A$ ... $V \times A_3)$

$= V \times ($ ... $13)$

$= V \times (5.23)$

$$\frac{0.23}{5} = 0.046 = 5\%$$

about ¼ of an iteration

Predicting the impact

$5 \times 34 = 170$

$5.23 \times 34 = 177$

$5 \times 41 = 205$

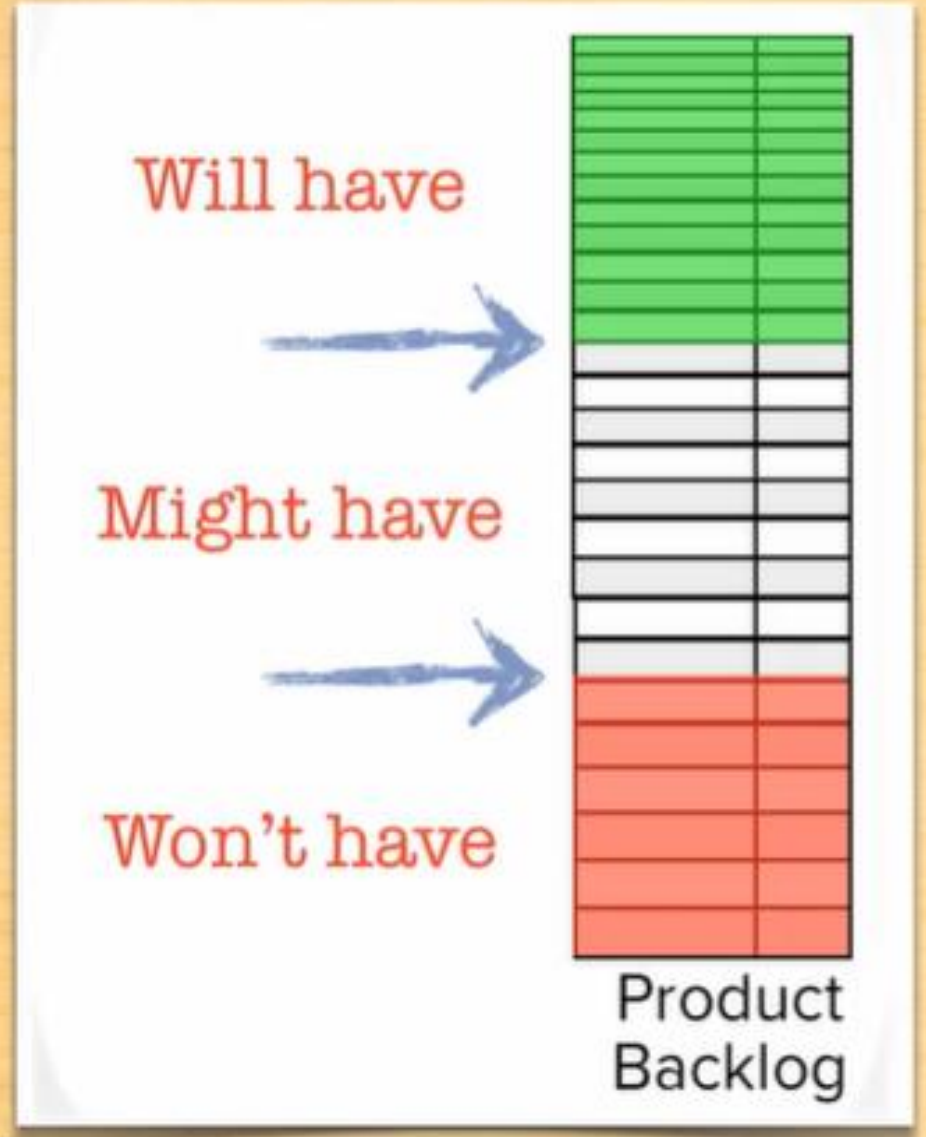
$5.23 \times 41 = 214$

Study Notes and Recap

It is possible to forecast an initial velocity by doing a commitment-driven iteration planning meeting and summing the story points or ideal days on the planned product backlog items.
When forecasting velocity, always express your forecast as a range.

Estimating velocity is definitely possible and teams should not tell their product owners to be wait for five iterations while they establish a velocity. The best way to estimate velocity is through a commitment-driven iteration planning meeting. This involves team members discussing each product backlog item, identifying its tasks, estimating those tasks in hours, committing to deliver the product backlog item, and then repeating until the iteration is full. During this meeting, the team does not discuss story points or velocity. When everyone feels the iteration is full, the team sums the number of story points (or ideal days) assigned to the selected product backlog items, giving an initial, rough forecast of velocity. But, whether forecast this way or from empirical data, velocity is always best expressed as a range. A near universal truth is that those who will do the work should estimate the work. A product owner who estimates the team's velocity without asking the team is asking for trouble. The estimate will almost certainly be overly optimistic. In some cases the product owner may need to do this (e.g., the team doesn't exist yet and no technical people are available to fill in for them), but the product owner should not estimate velocity merely to avoid bothering the team.

Fixed-date planning



Three steps

1. Determine how many iterations you have.
2. Estimate velocity as a range.
3. Use that range × the number of iterations to partition the backlog into Will Have, Might Have, and Won't Have.

An example


APRIL
SUN MON TUE WED THU FRI SAT
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
MAY
SUN MON TUE WED THU FRI SAT
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
JUNE
SUN MON TUE WED THU FRI SAT
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

By the end of June we can deliver 150–180 story points.

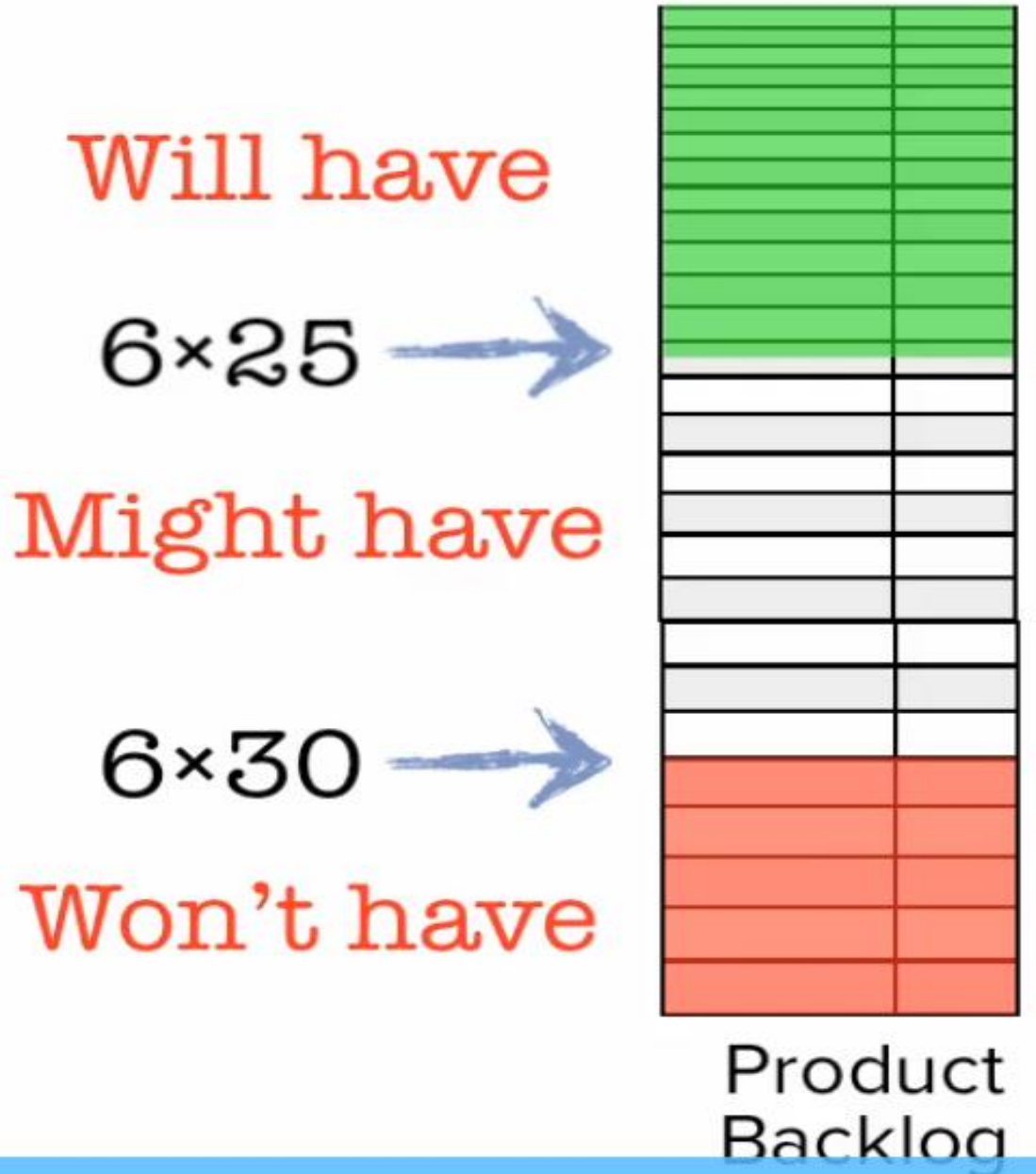
Great. I'll take the 180.

An example

Number of iterations = 6

Low velocity = 25

High velocity = 30



Fixed-date contracting

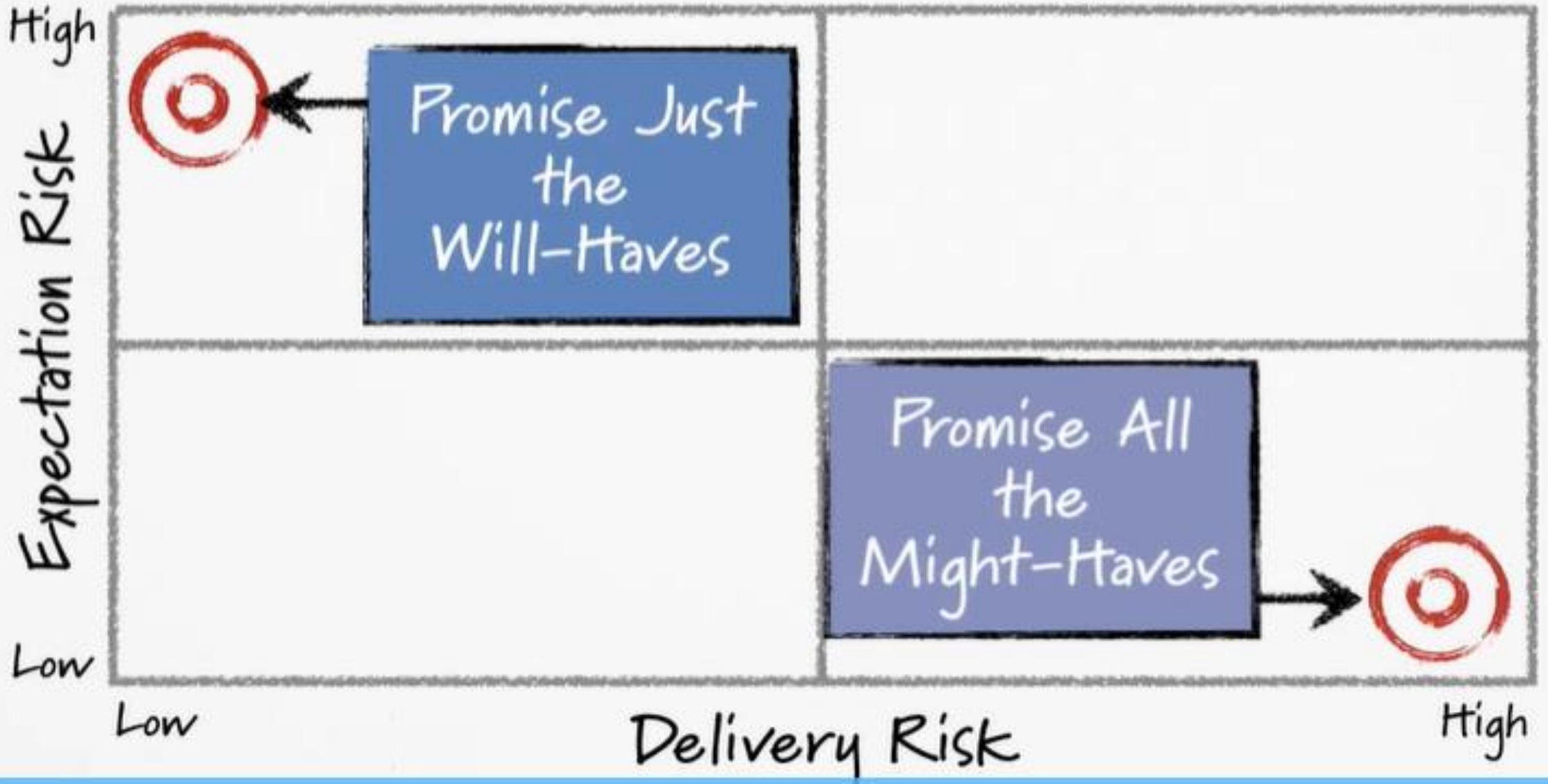
If you write the contract here

- You probably won't get the contract
- But will probably finish everything if you do

If you write it here

- You'll probably win the contract
- But will probably not finish everything in time

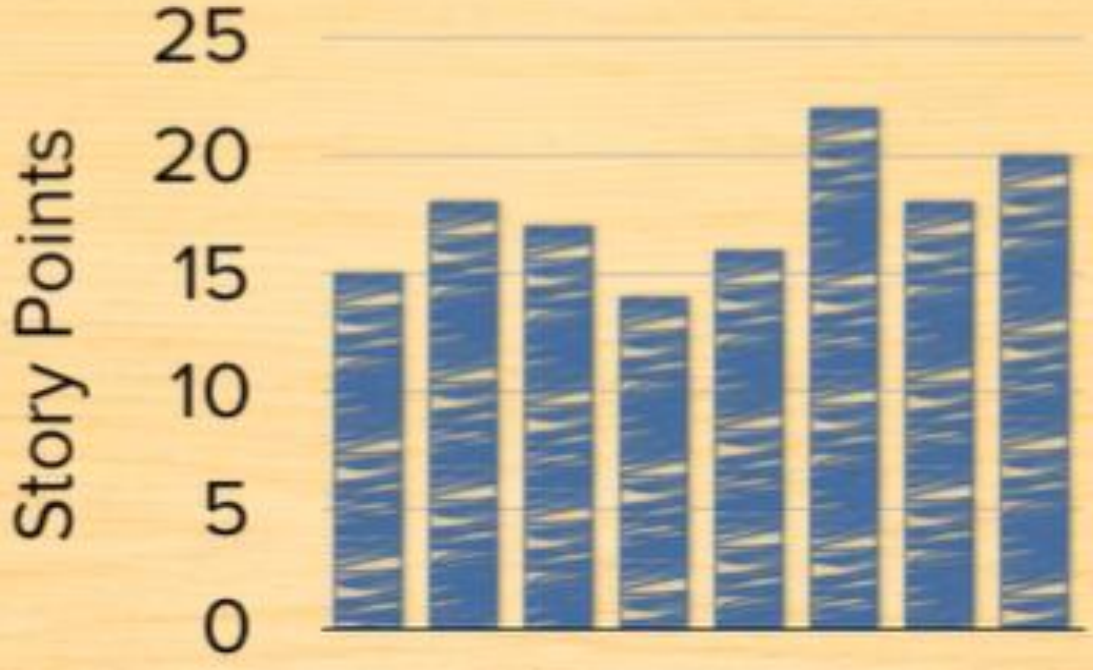
Balancing Risk



Three steps

1. Sum the product backlog items.
2. Estimate velocity as a range.
3. Use the sum of the backlog divided by the velocity range to determine a date range.

= 120 story points



}=15-20

Fixed-scope contracting

If you write the contract for the *short* duration

- You'll probably win the contract
- But it may be hard to finish everything that quickly

Lots of delivery risk
Not much expectation risk

If you write the contract for the *long* duration

- You probably won't win the contract
- But it should be easy to finish everything in time

Not much delivery risk
Lots of expectation risk

Two types of fixed-price project

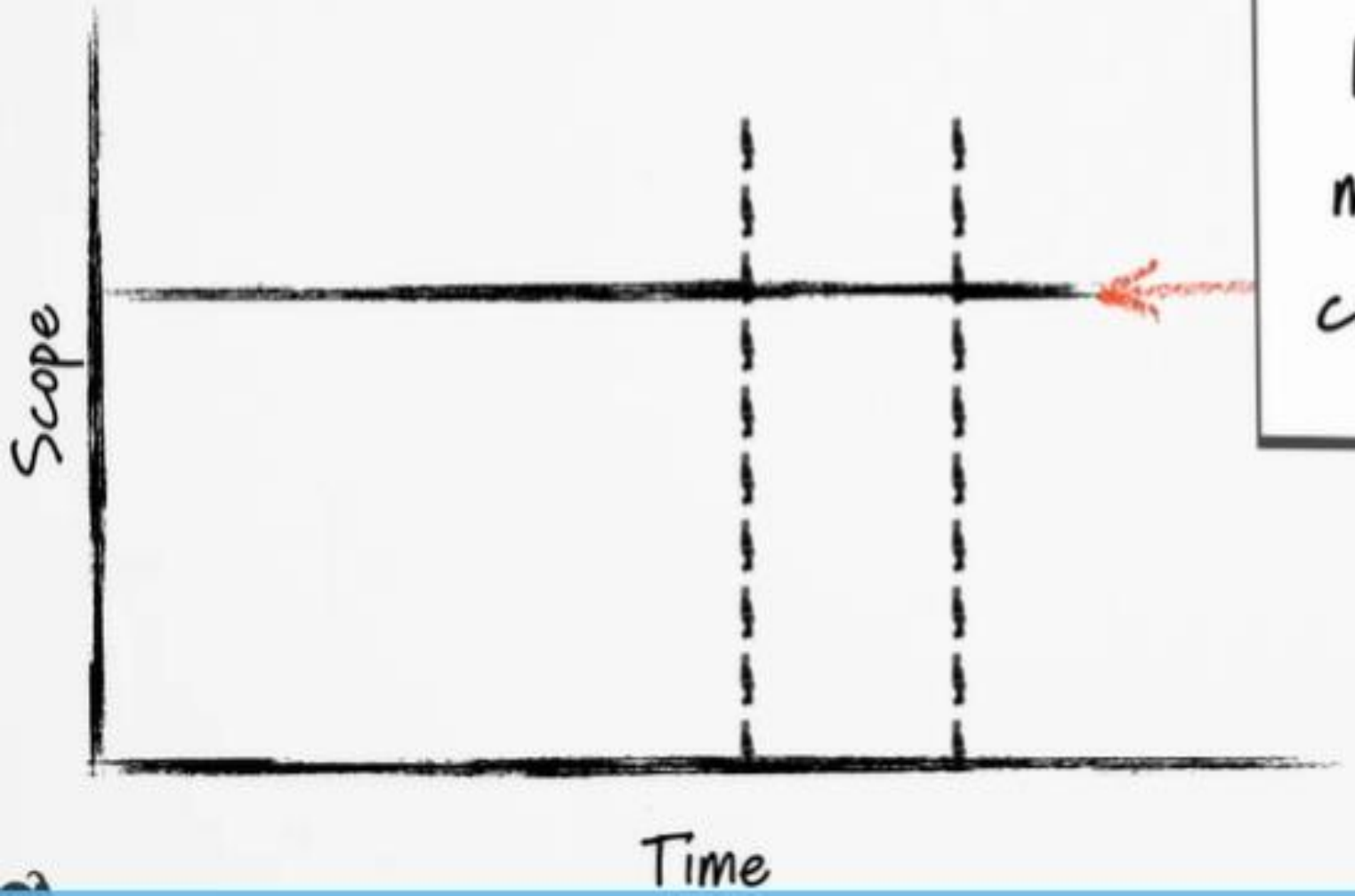
No choice

1. Assess the risk
2. Decide to do the project or not
   - Hopefully you have that choice

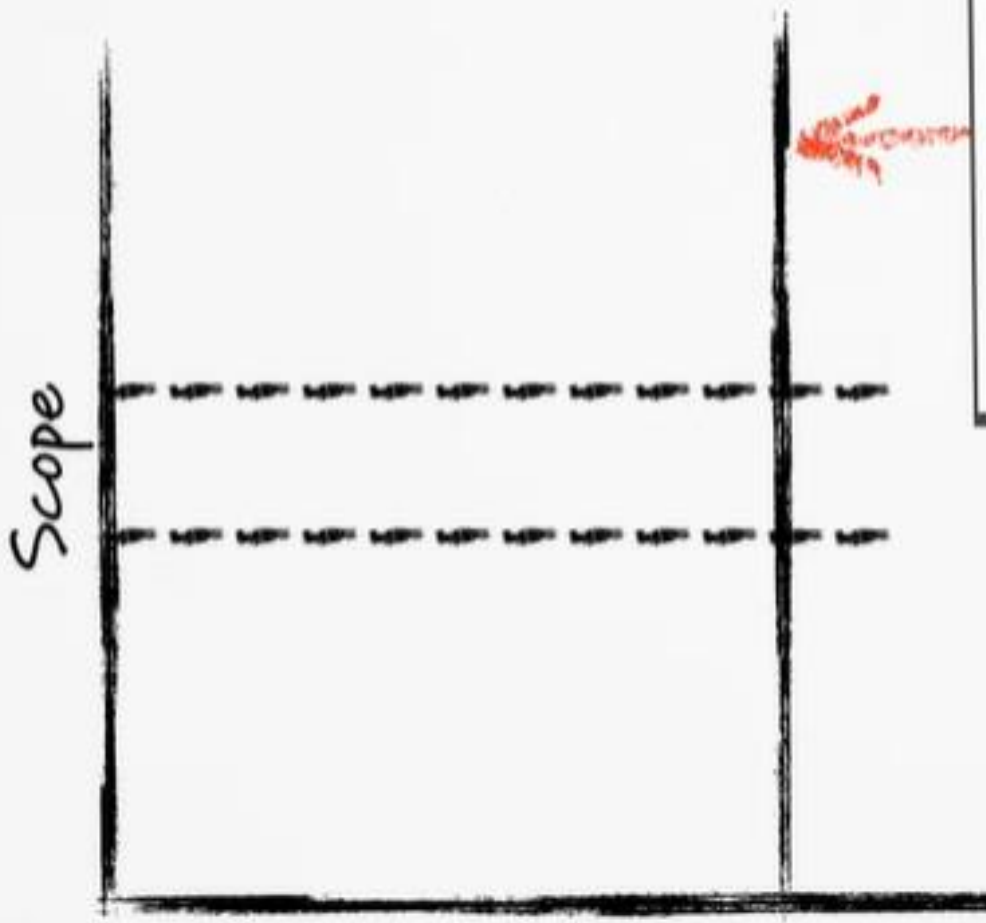
Some flexibility

1. Assess the risk
2. Try to balance the scope and schedule risk equally

A fixed-scope project



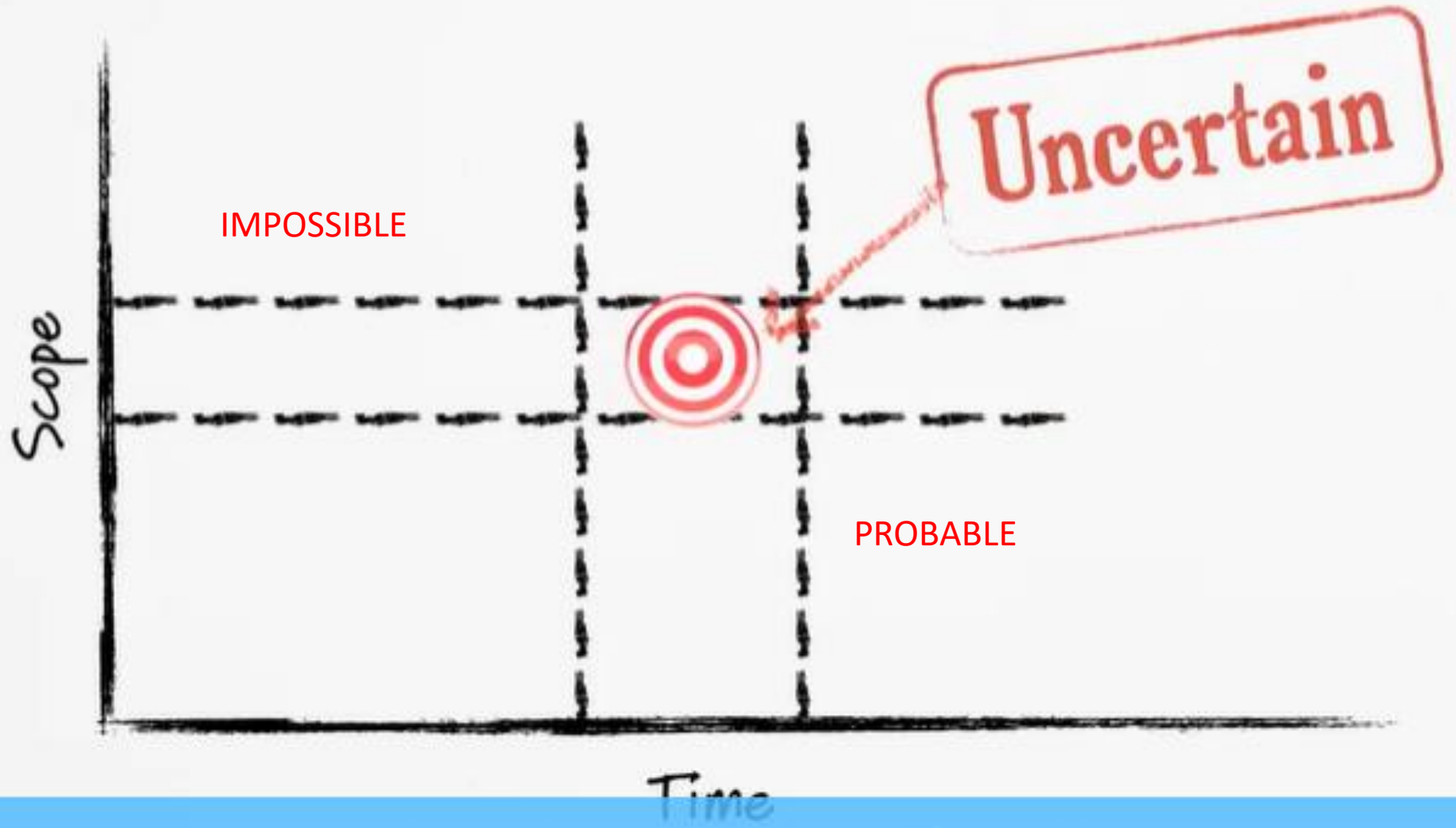
A fixed-date project



We need it on this date. How much can we have by then?

A fixed-price project

A fixed-price project



Step 1: Estimate velocity

- Use a range

Step 2: Fix the scope

- Divide fixed scope by the velocity range
- Draw the vertical lines

Step 3: Fix the schedule

- Determine the number of iterations the team has
- Multiply by the velocity range
- Draw the horizontal lines

An example

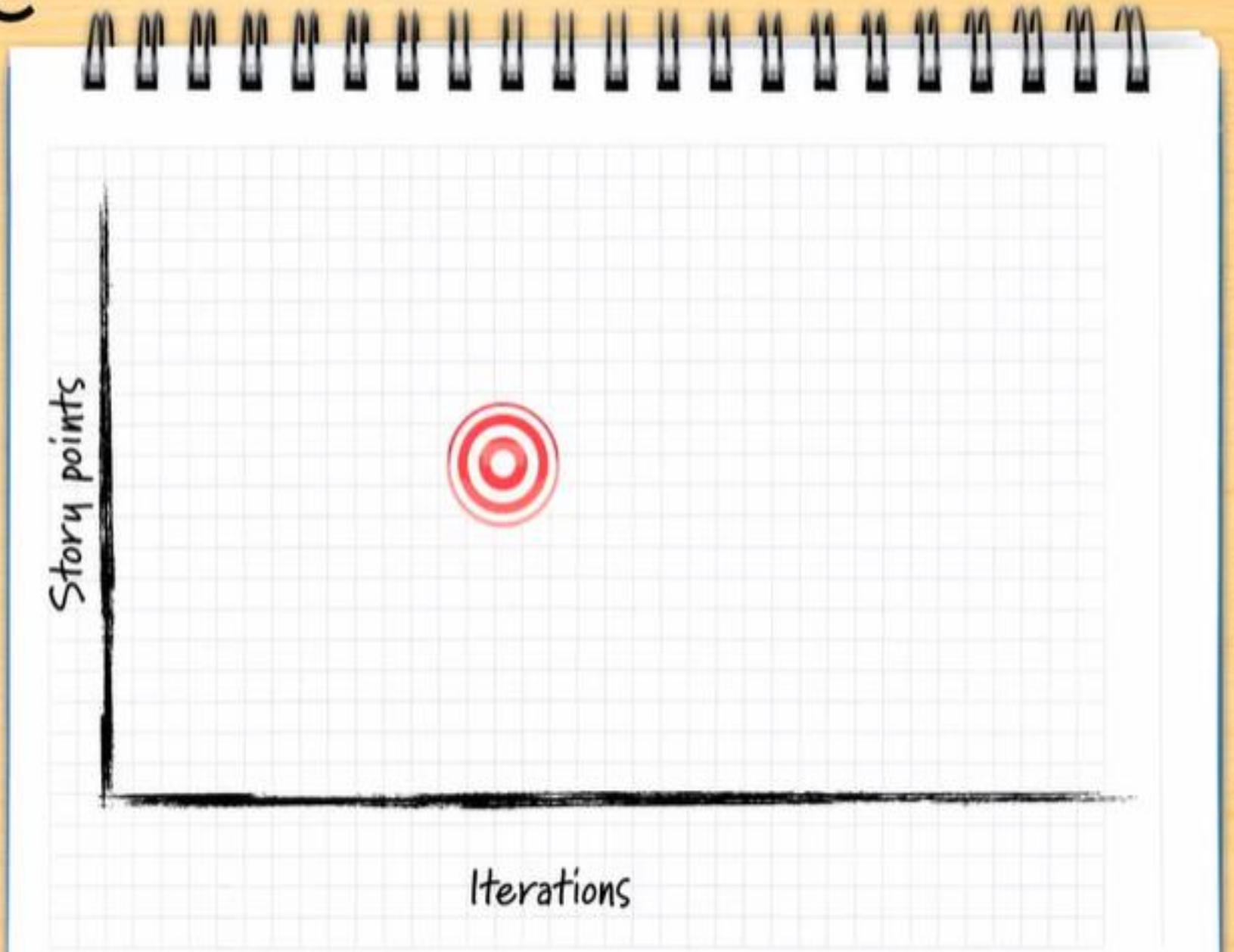
Deadline = 10 iterations

Scope = 200 points

Velocity = 16–22

200 / 22 = 9.1

200 / 16 = 12.5



An example

Deadline = 10 iterations

Scope = 200 points

Velocity = 16-22

10x16 = 160

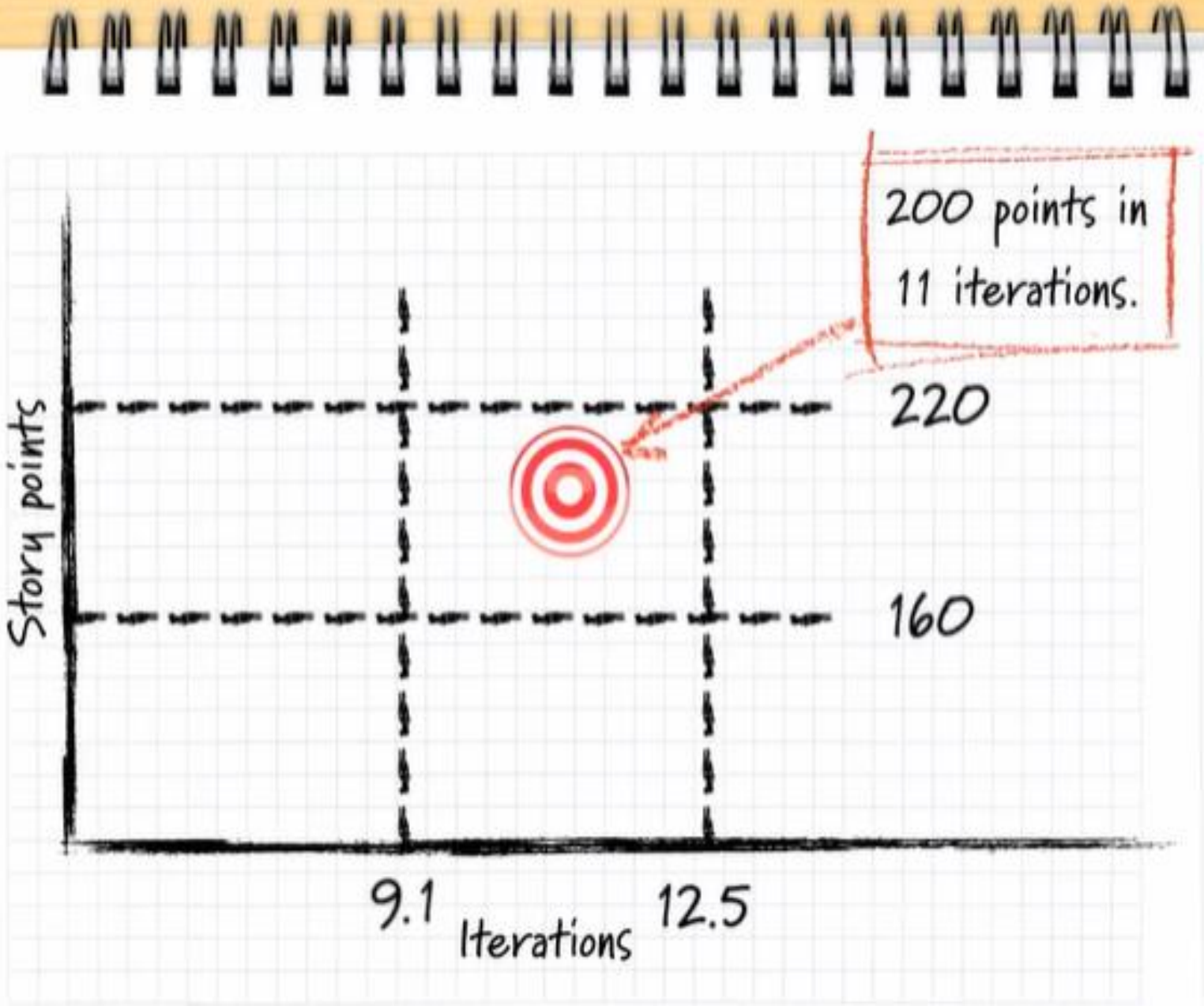
10x22 = 220

An example

Deadline = 10 iterations

Scope = 200 points

Velocity = 16–22

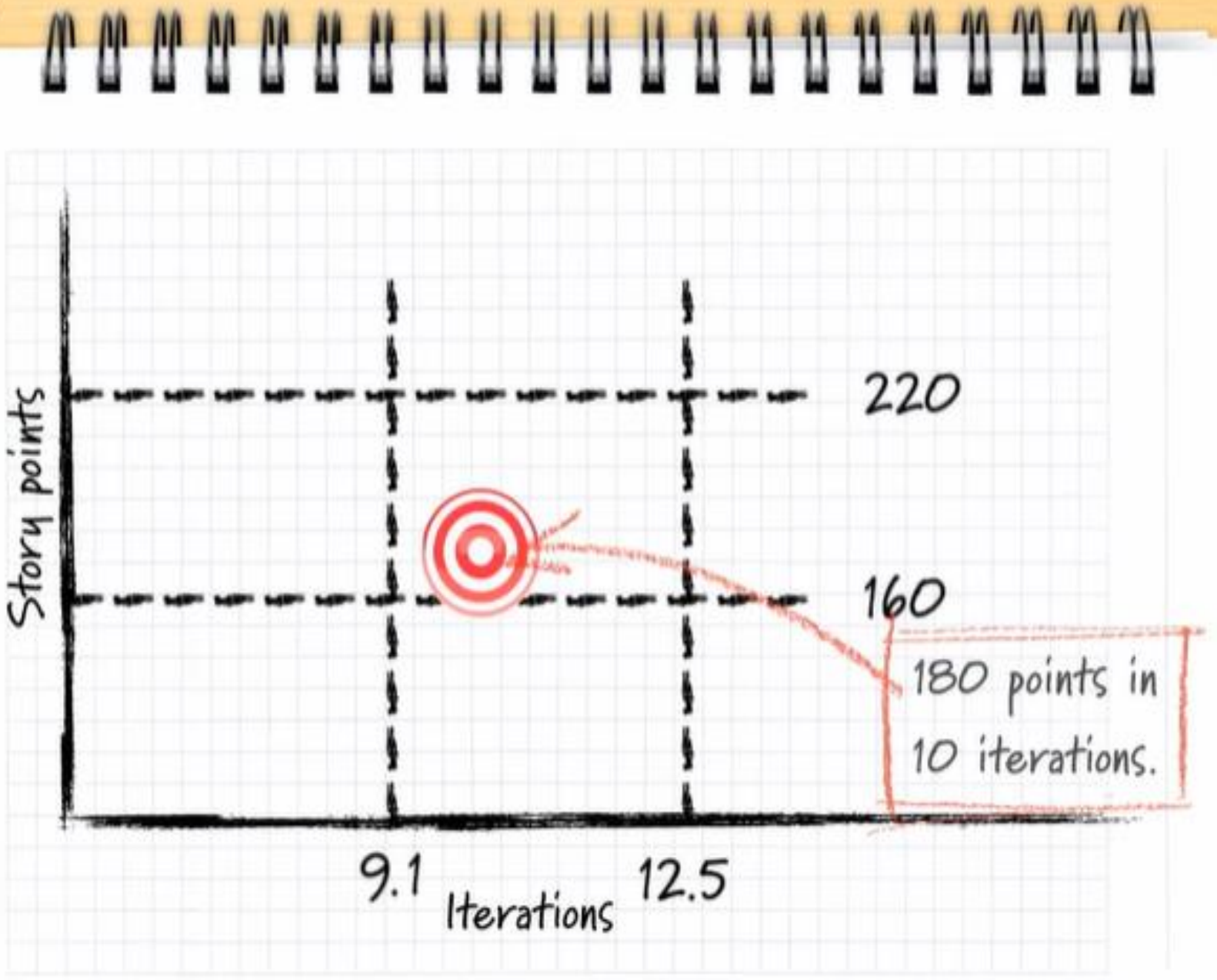


An example

Deadline = 10 iterations

Scope = 200 points

Velocity = 16-22



Study Notes and Recap

With agile, it is absolutely possible to commit to a fixed scope. A fixed-scope plan answers the question of when a set of product backlog items can be delivered. Ideally a fixed-scope plan will be given as a range: "You will get that fixed scope sometime between this and that date." Unfortunately, plans often have to be narrowed and overly specified into single-date estimates because bosses, clients, and customers prefer the false certainty of having a single date. If you promise a delivery date based on the high end of a team's predicted velocity range, that will be the earliest possible delivery date for the fixed scope. If you promise to deliver everything by the earliest possible date, you'll probably meet your customer's expectations but you run the risk of letting that customer down later when you don't deliver everything as promised.

Issues for multi-team projects

1. Estimating in a common unit
2. Iteration planning
3. Dependencies

Do create a
common baseline.

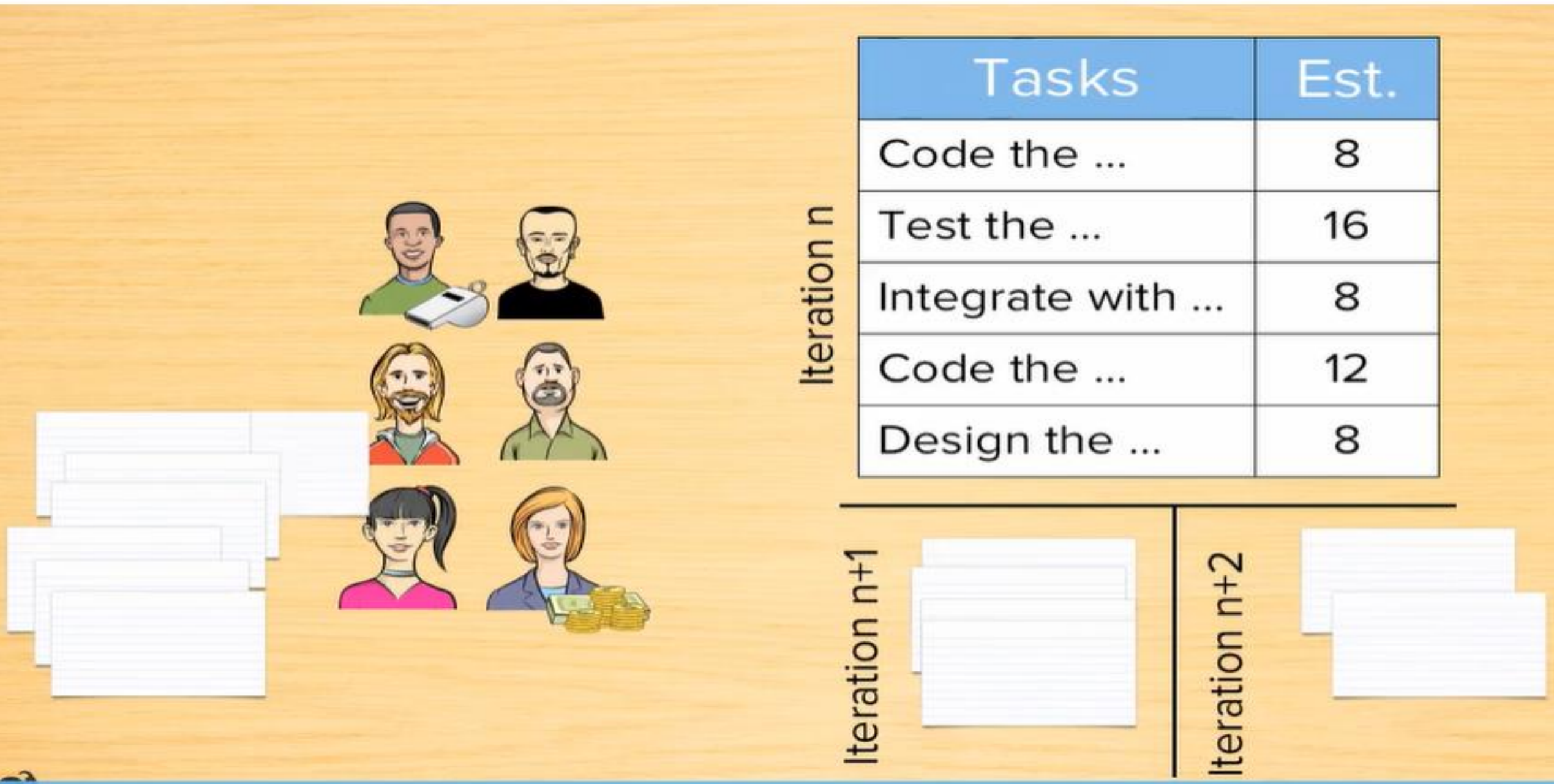
But be careful
comparing velocities.

The Big Room

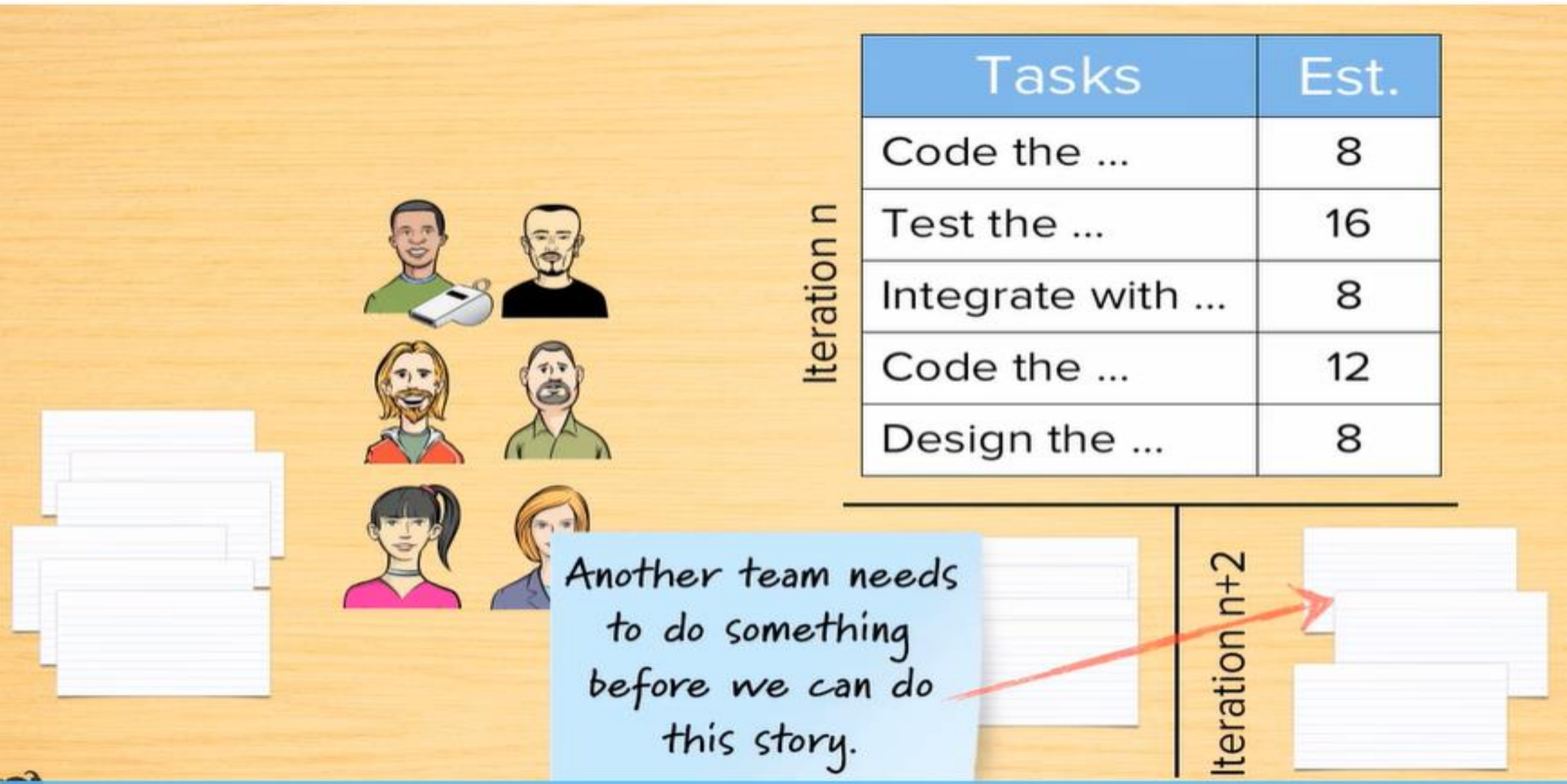
- All iterations end on same day
- All planning is on same day and in one room
- Key people move between teams as needed

Monday	Tuesday
 	

Rolling Lookahead Planning 3:35



Rolling Lookahead Planning 3:35



After planning iteration n			
n		n+1	n+2
Task A	16		
Task B	8		
Task C	4		
Task D	12		

Suppose that while planning iteration n+1, an item rolls into visibility for iteration n+3 that is dependent on work by another team

After planning iteration n+1			
n+1		n+2	n+3
Task J	6		
Task K	14		
Task K	16		
Task L	4		


Can your team do this for us?
But we could do it next iteration.

Study Notes and Recap

It is possible to establish a common baseline for story points or ideal days across a multiple-team project.
If a common baseline for story points or ideal days is established, you should still avoid comparing velocities between teams.

It is entirely possible to establish a common baseline across multiple teams for story points or ideal days. To do so, have one or more representatives of each team collaboratively estimate a set of representative stories with Planning Poker. However, even with such a baseline you should not compare velocities between teams. Comparing velocities across teams usually results in inappropriate behavior arising from the competition. One such problem is estimate inflation, which, fortunately, is fairly easy to detect.

Study Notes and Recap

Not every team needs to do rolling look-ahead planning.
Spending as little as ten minutes looking ahead at the next two iterations can be helpful.

Rolling look-ahead planning can be done in as little as ten minutes per iteration and provides a helpful peek into work that will be done two and three iterations ahead. Although not every team needs to do this, it is a helpful technique on multiple-team projects. A team looking ahead at the iterations to follow the current one does not make commitments to the work of those iterations. The team is merely making high-level predictions about which product backlog items might be done in those iterations. Doing this will not identify all dependencies between teams, but it will often find enough to justify the meager time investment required.

Summary

1. 2 Types of Iteration Planning (Commitment Driven and Velocity Driven)
2. Estimate Size, Derive Duration
3. Story Points and Ideal Days
4. Planning Poker For Estimating User Stories
5. Dangers of Human Nature to prefer precision over accuracy
6. Using Ranges to make sure plans are accurate
7. Calculating confidence interval using historical velocity Data

Agile Estimating and Planning

Goodbye