Table of Contents

Estimation & Planning Guidelines

1. Purpose

Estimation and planning ensure that IT projects are delivered on time, within scope, and with high quality. This guideline provides a structured approach to estimating effort, defining deliverables, and planning project timelines using Agile principles. Proper estimation helps teams set realistic expectations and optimize resource allocation for efficient delivery.

2. Scope

This guideline applies to all Agile teams involved in software development projects, utilizing Azure DevOps for backlog management, sprint planning, and tracking progress. It is intended for developers, testers, project managers, and product owners to ensure collaborative and effective planning across all project phases.

3. Entry Criteria

Before starting estimation and planning, the following must be in place:

- A clear product vision and business objectives to guide the development
- Defined scope and high-level requirements to ensure alignment with stakeholders
- A cross-functional team with relevant expertise to contribute to estimation accuracy
- Availability of historical data or reference points for estimation (if applicable), helping in better forecasting

4. Inputs

The estimation and planning process relies on the following inputs:

- **Product Backlog**: The product backlog is a prioritized list of all features, user stories/Tasks, enhancements, bug fixes, and technical improvements that are considered necessary for the product and need to be estimated. It acts as the sole source of requirements for any changes to be made to the product.
- **Sprint Backlog:** The sprint backlog is a subset of the product backlog that consists of items (user stories, tasks, or bugs) selected by the development team to be completed during a specific sprint (usually a time-boxed iteration of work)

- **Sprint/Iteration Goals**: Sprint or iteration goals are specific, short-term objectives that a development team aims to achieve within a sprint (or iteration), typically lasting 1–4 weeks. These goals guide the team's efforts and help maintain focus on delivering value.
- **Velocity Driven Iteration Planning:** A planning approach where the scope of an iteration (sprint) is based on the team's historical **Velocity or Past performance.**
- **Velocity:** It is the average amount of work (typically measured in story points or similar units) the team has completed in previous sprints.
- **Commitment Driven Iteration Planning**: A planning approach where the team makes a firm commitment to deliver a specific set of high priority items during the iteration. Team defines and estimates the tasks before committing. The sprint backlog is treated as a promise to complete the selected work within the sprint.
- **Historical Data**: Past project performance, **velocity**, or benchmark estimates to guide accuracy
- **Ideal Time (Used for Estimation)**: The amount of uninterrupted work time required to complete a task, assuming no distractions, interruptions, or dependencies.
- **Elapsed Time (Not used for Estimation)**: Time from Start to finish of the work, including all interruptions, breaks, and any context switching.
- **Team Capacity**: Availability and skills of the team members to determine workload feasibility
- **Constraints**: Budget, timeline, resources, and dependencies that may impact the project scope
- **Acceptance Criteria (or Conditions of satisfaction)**: Definition of when work is considered complete, ensuring clear deliverable expectations

5. Activity List

A structured approach to estimation and planning includes:

- **Backlog grooming and refinement**: Ensuring backlog items are well-defined and ready for estimation
- **Sprint planning and task breakdown**: Breaking down work into smaller, manageable tasks
- **Capacity assessment and workload distribution**: Ensuring tasks are assigned based on team availability
- **Risk identification and mitigation**: Recognizing potential challenges early and planning contingencies

6. Estimation Techniques

To improve accuracy, estimation follows these techniques:

I. Relative Estimation (Story Points, T-shirt Sizing)

Rather than estimating in absolute time, teams assign a relative effort value:

- **Story Points**: Compares tasks to a baseline task and assigns points based on complexity, effort, and uncertainty. Using the right units for estimating ensures accurate projections. Such as, using a set of numbers like 1, 2, 3, 5, 8, 13, 20, 40, 100 etc. that make sense.

  Reasons to Favor Story Points:

  - **Focus on Complexity, Not Time:** Story points emphasize the relative effort, complexity, and uncertainty of a task rather than the specific time it will take, which can be more useful in agile environments.
  - **Better for Agile Teams:** They help sprint planning by allowing teams to estimate without being constrained by the assumption that every task will take the same amount of time.
  - **Reduces Time Pressure:** Story points prevent teams from feeling pressured to meet specific time estimates, promoting a more flexible and adaptive approach to task completion.
  - **Helps with Velocity Tracking:** It allows teams to track their velocity (how much work they complete per sprint), providing a more consistent measure of performance over time

- **T-shirt Sizing**: Categorizes tasks into XS, S, M, L, XL for quick assessment.

II. Planning Poker

A consensus-based approach to estimation:

- Team members choose an estimate (typically Fibonacci numbers) and reveal their choice simultaneously.
- If estimates differ significantly, discussion follows until consensus is reached.
- **Benefits**: Encourages discussion, reduces anchoring bias, and balances perspectives.
- Step by step guide to **Planning Poker**:

  1. **Prepare Cards**: Each team member gets a set of cards with numbers (usually Fibonacci numbers: 1, 2, 3, 5, 8, etc.).
  2. **Present the Task**: The product owner explains the task or user story to estimate.
  3. **Discuss**: The team talks about the task to make sure everyone understands it.
  4. **Choose Cards**: Everyone secretly picks a card to estimate how hard the task is.

5. **Reveal Estimates**: All cards are revealed at the same time.
6. **Discuss Differences**: If estimates vary a lot, the team discusses why and what they think the effort should be.
7. **Re-estimate**: After the discussion, the team votes again if needed.
8. **Record Estimate**: Once the team agrees, the estimate is noted down.
9. **Repeat**: Repeat for each task or user story.

The main goal is to agree on an estimate through discussion and collective understanding.

Planning Poker works well with teams because it encourages collaboration, ensures everyone's opinion is heard, and helps the team reach a consensus on estimates. It eliminates bias by having everyone choose their estimate in secret, promotes more accurate estimates by combining team knowledge, uncovers any uncertainties or misunderstandings and helps avoid anchoring. The process engages all team members, fosters healthy debate, and ensures collective ownership of the estimates.

III. Expert Judgment

Relies on experienced team members to estimate based on experience and technical knowledge.

- Works best when combined with other techniques like relative estimation.
- **Benefits**: Useful when historical data is lacking or for complex tasks.

IV. Affinity Estimation

A quick categorization technique:

- Items are grouped based on similarity in effort.
- The team places them into predefined categories (small, medium, large).
- **Benefits**: Fast, avoids overthinking, and ideal for large backlogs.

V. Ideal Days & Velocity-Based Estimation

Uses past performance to forecast effort:

- **Ideal Times**: The time required to complete a task without interruptions. It represents the best-case scenario where the task is completed in an uninterrupted flow, focusing only on the work itself.

Reasons to Favor Ideal Time:

- **Clear Time Expectations:** Ideal time is more straightforward as it relates to how long a task will take, which makes it easier for team members and stakeholders to understand.
- **Realistic Planning:** It is based on actual time, helping teams plan more accurately when managing their schedules and deadlines.
- **Less Ambiguity:** There is no need to convert between different scales or assume relative difficulty, as the time required for each task is explicitly specified.

- **Velocity-Based Estimation**: Uses past sprint performance to predict work capacity.

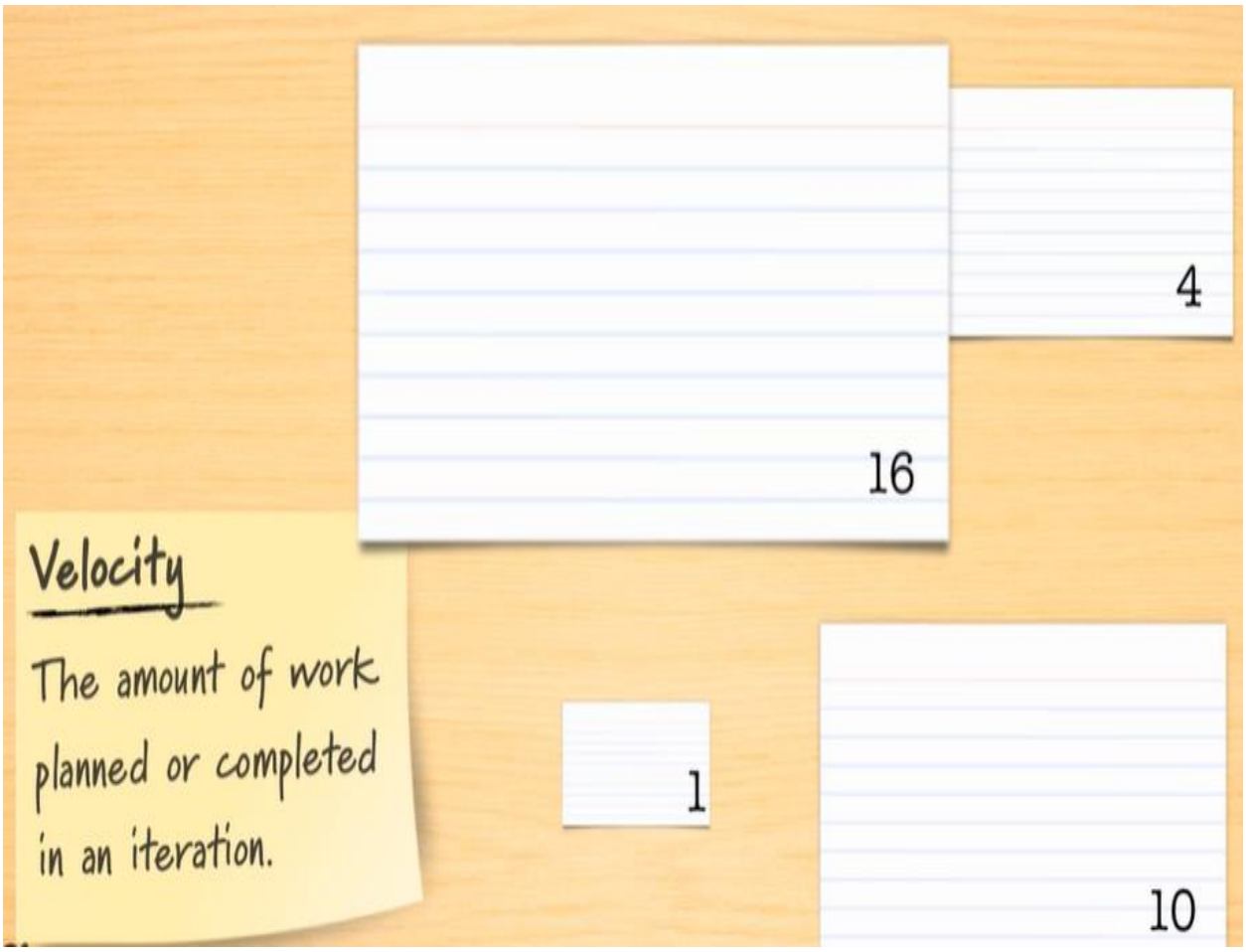
Reasons to Favor Velocity Based Estimation:

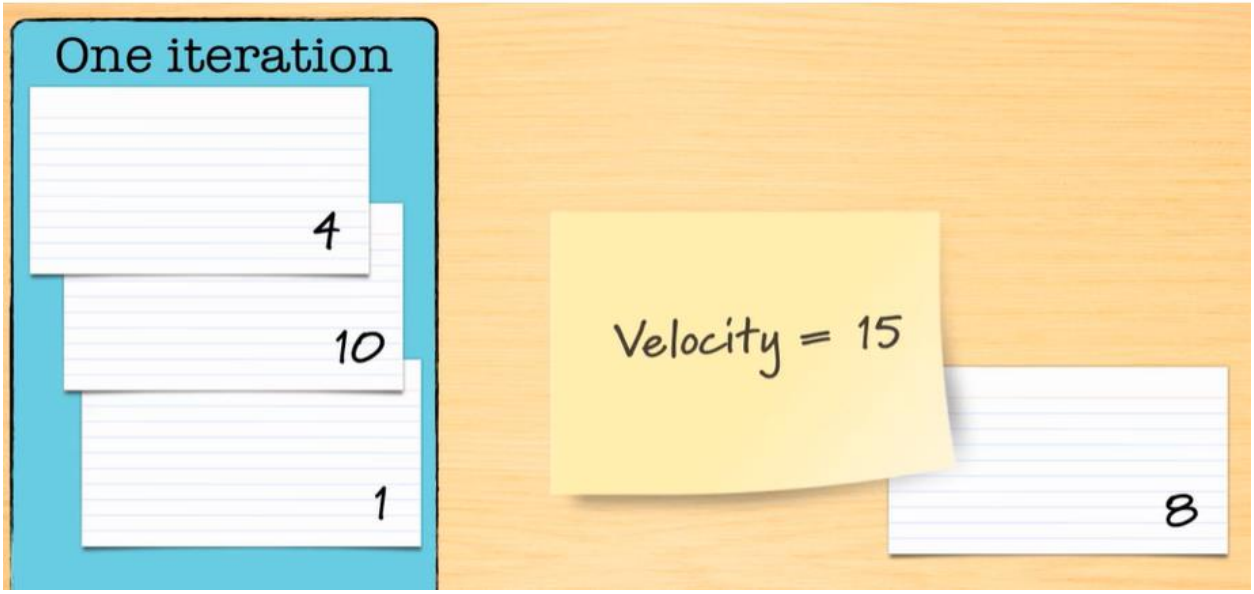
- **Data-Driven Forecasting:** Uses empirical data from past iterations to predict how much work can be accomplished in the upcoming sprint.
- **Flexible Scope:** The team selects backlog items up to a target that aligns with their average velocity rather than making a rigid promise to complete a fixed amount of work.
- **Continuous Improvement:** The historical velocity can be refined over time as the team becomes more predictable, helping in future planning.

Example of Estimate size/Derive Duration:
Let us consider,
Backlog = 28 Kilos
In the first iteration, we completed 7 kilos of work.
Here, Velocity = 7
So, Duration = 28 / 7 = 4

There will be 3 Scenarios for Velocity Based Estimation,
(A) one where there is a lack of historical data for previous sprints (For all teams)
(B) Another situation will be where historical data is available for previous sprints.
(C) Lastly, we may have Historical Velocity Data from Other teams, but not the team that needs it. **Relative Standard deviation calculation** will be used here.

For Scenario (A), consider taking Velocity from the most recent sprint performed by the team. Like Below, where we assume the team Velocity to be 15 Story Points from their previous sprint performance. The previous sprint acts as the baseline for velocity estimation.


4
16
Velocity
The amount of work planned or completed in an iteration.
1
10



If you do not have historical Data and must rely on experienced team members with **Product Knowledge, Domain expertise and technical knowledge to determine velocity like above.**

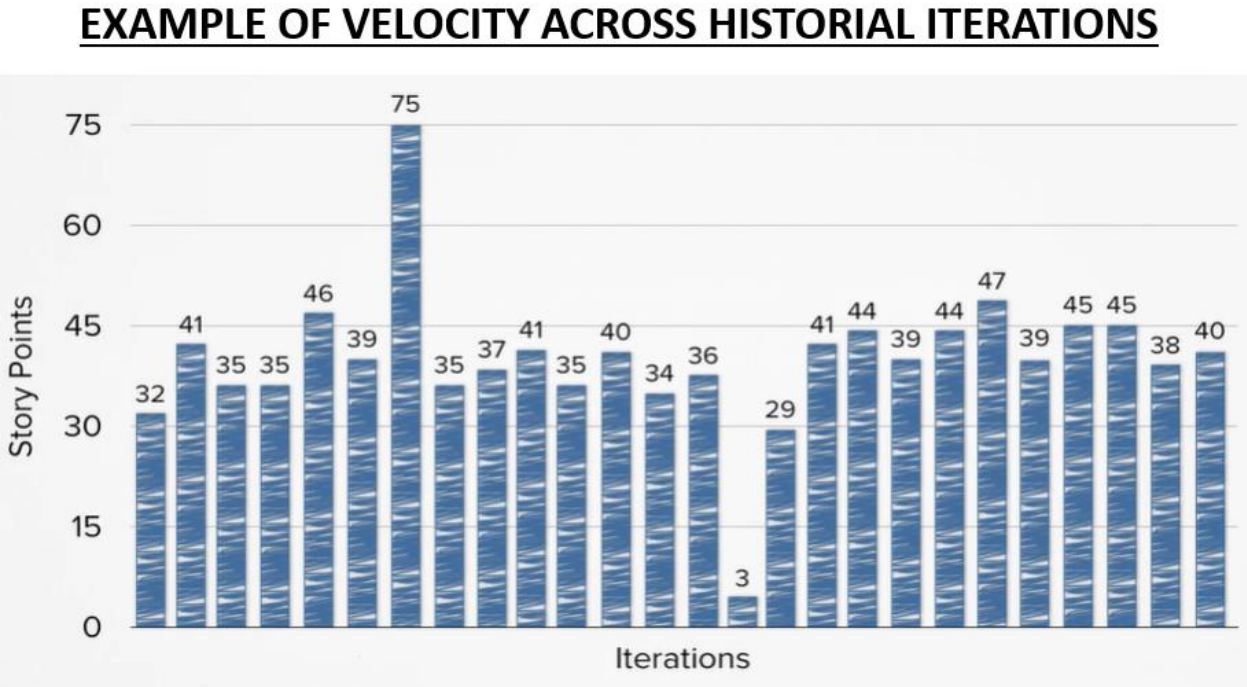
Use a buffer of +-15 % on velocity estimate. So, for this example the estimated velocity was 15 from the previous sprint, add and subtract 15% from 15.

15x1.15=17.25=17

15x0.85=12.75=13

Giving us 13 and 17 as Velocity Value range (Rounded up to nearest whole number).

B) Scenario (B) is where there is historical sprint Velocity data available for the team



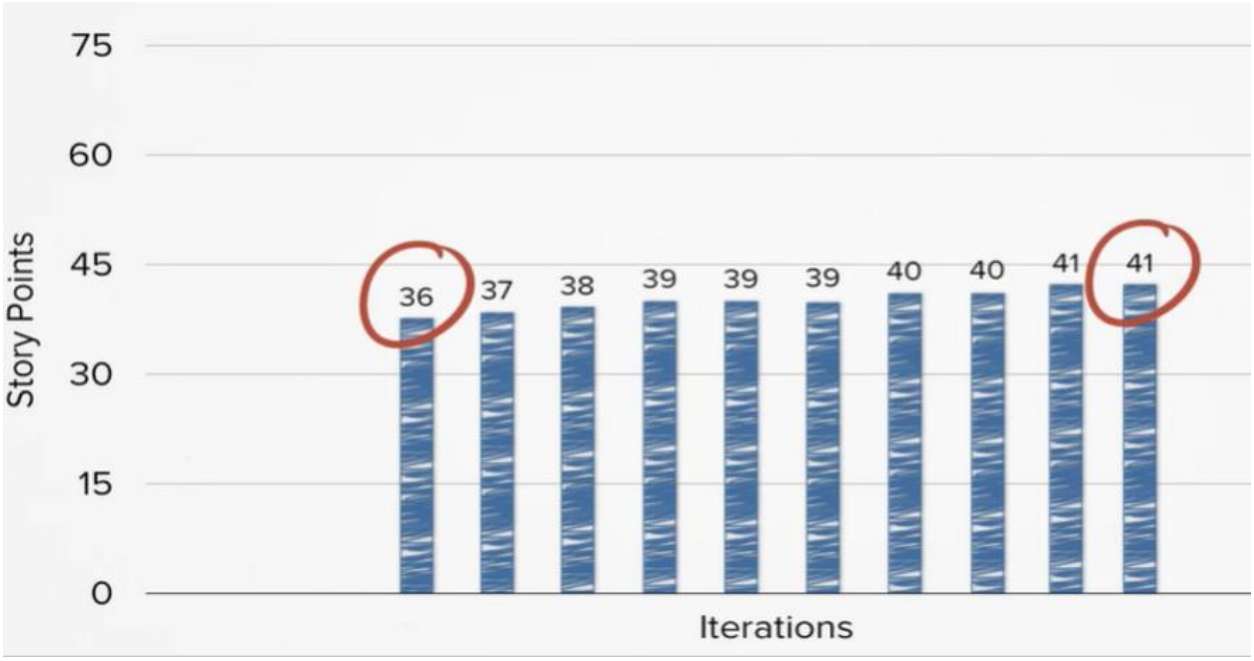
32,41,35,35,46,39,75,35,37,41,35,40,34,36,3,29,41,44,39,44,47,39,45,45,38,40 = **26 Values in total**

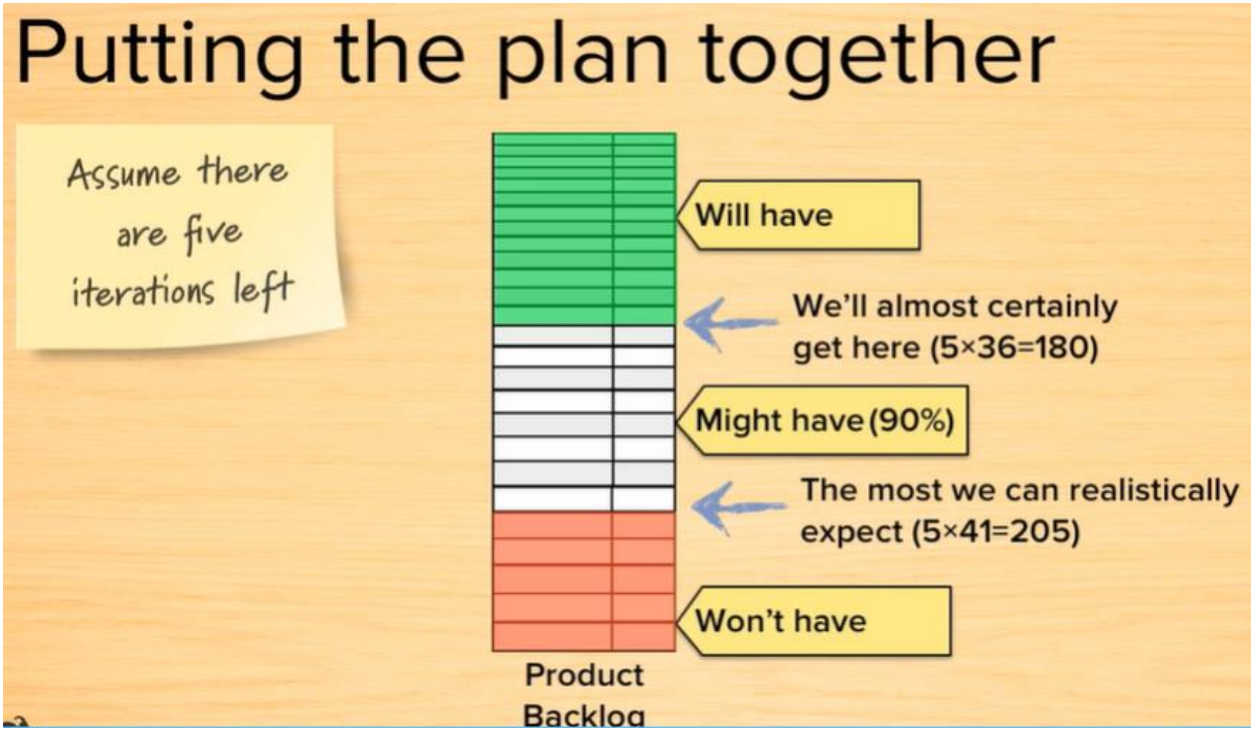
Remove the lowest and highest outliers. In this case 75 and 3. Leaving 24 Total Values.

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

We have **24 Sprint/Iteration Velocity Data**, following the table above, we will **remove 7 Values** from each end of the values. The result gives us the statistically true average Value Range shown below:

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





Therefore, if we have 5 sprints/Iterations left, we can multiply that with the Velocity Range Values. So we can separate the product backlog into Will have, might have and Won't Have sections.

Alternatively, use past velocity data into the calculator below to determine team capacity.

Velocity Range Calculator is an Agile tool that predicts a team's work capacity based on past sprint velocities.

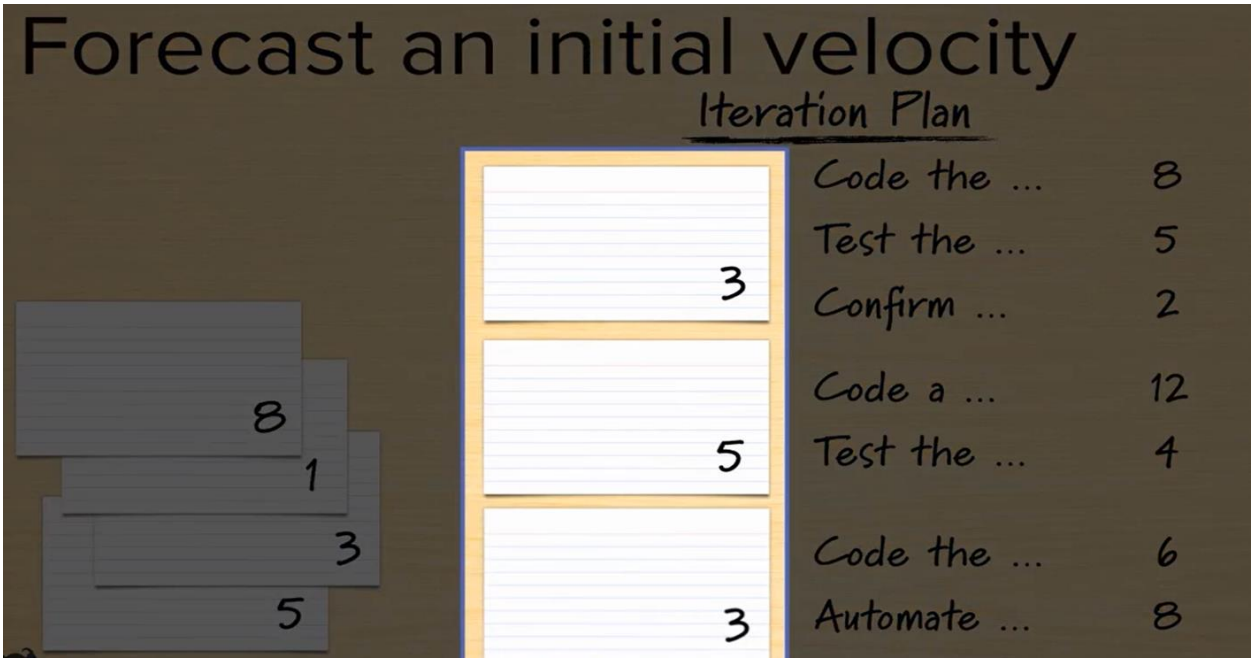
How It Works:

1. Collect past sprint velocities (e.g., 18, 22, 20, 25 story points).
2. Identify the **minimum, maximum, and average** velocity.
3. Use this range to estimate future sprint capacity realistically.

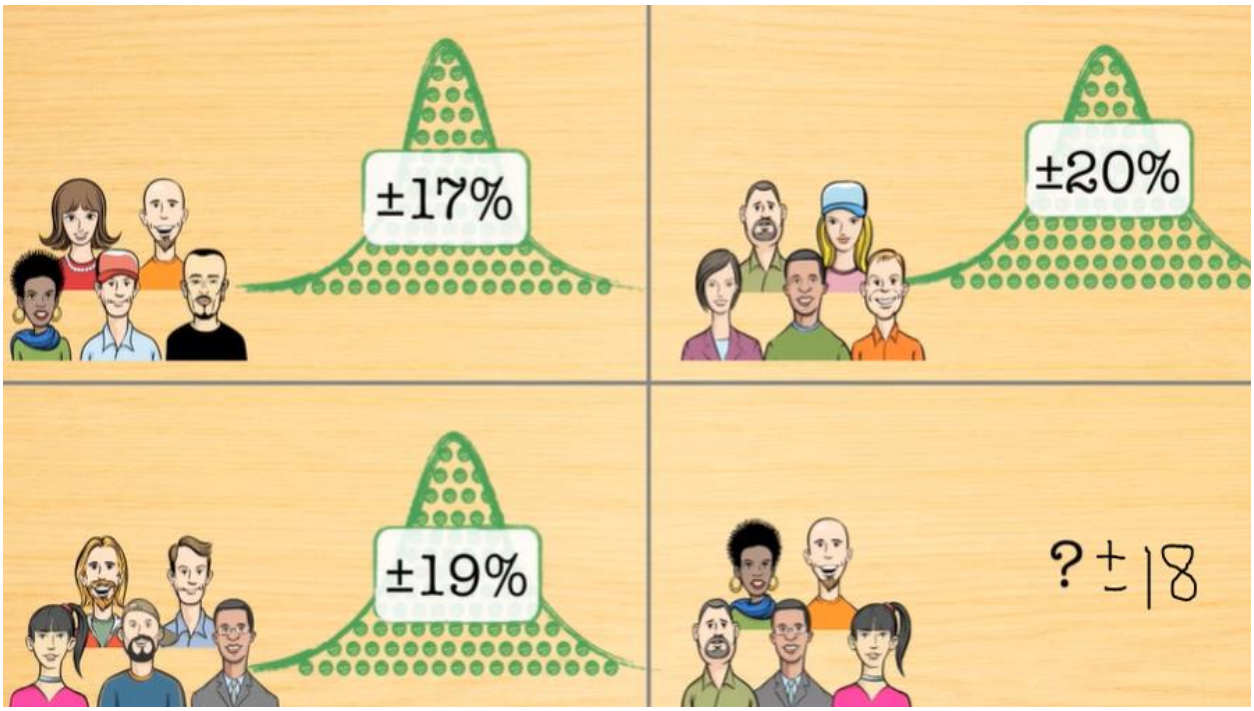
Online Calculator Link: https://www.mountaingoatsoftware.com/tools/velocity-range-calculator

(C) For **Scenario (C)** we may have Historical Velocity Data from Other teams, but not the team that needs it. **Relative Standard deviation calculation** will be used here.

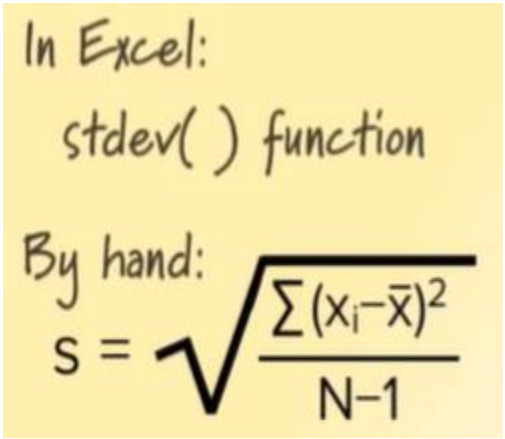
Firstly, we do commitment driven iteration planning for **forecasting an initial Team Velocity** (From Latest Sprint for example). Here 3+5+3 = **11** is the **Initial Team Velocity** based on story points of work taken into sprint.



Then, we find out how variable the other team's velocity is (In percentage terms) as shown below.



Let's see how we start doing this. Take an example of **Team-A** below with historical velocity data. By using the formula below we will determine the Relative Standard Deviation.



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

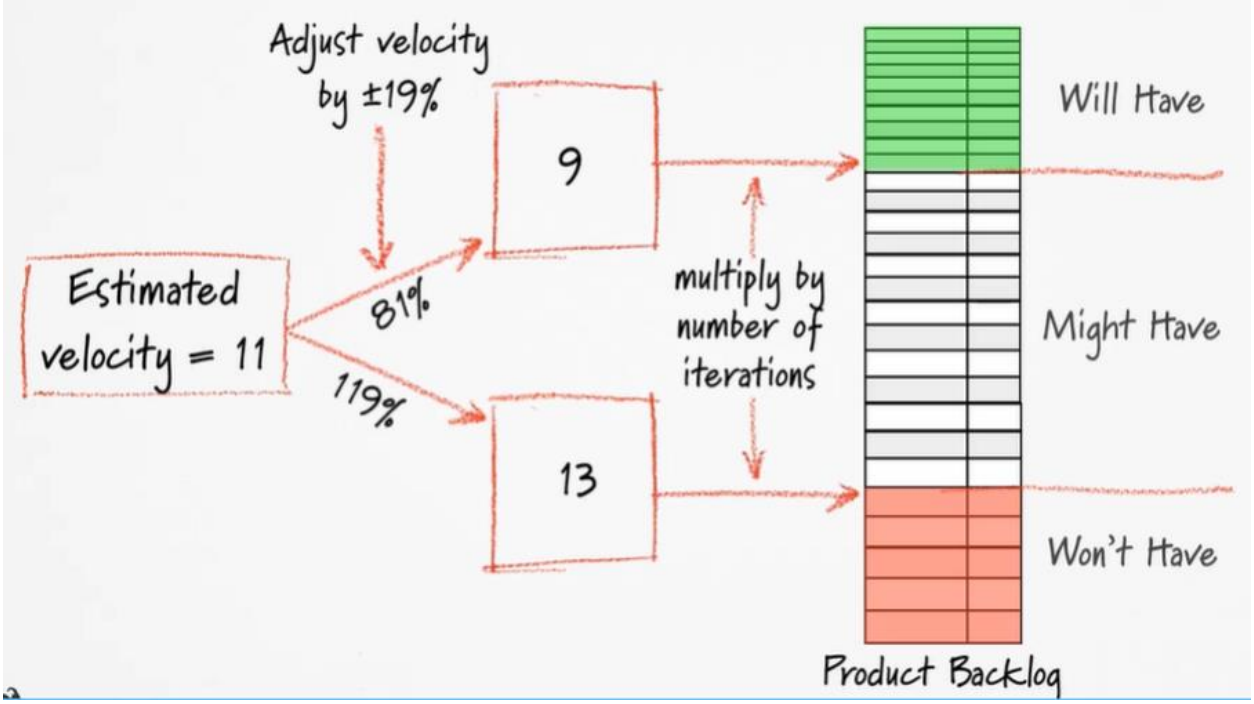
$$\text{Relative standard deviation} = \frac{4.0}{22} = 18\%$$

We will then repeat this step for the OTHER TWO TEAMs with Historical Velocity Data, like above.

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%

We then average the Relative Standard Deviation for all 3 teams, giving us +-19% of the initial estimated velocity (11 Story points) from the Commitment driven sprint planning (Provided below).

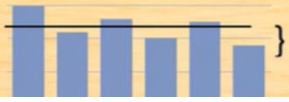


We can then separate the product backlog into Will have, Might Have and Won't have sections successfully.

Predicting Velocity when Team Size Changes (Track Velocity for 3 sprints for percentage changes in Velocity for Sprint 1, 2 and 3)

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%		

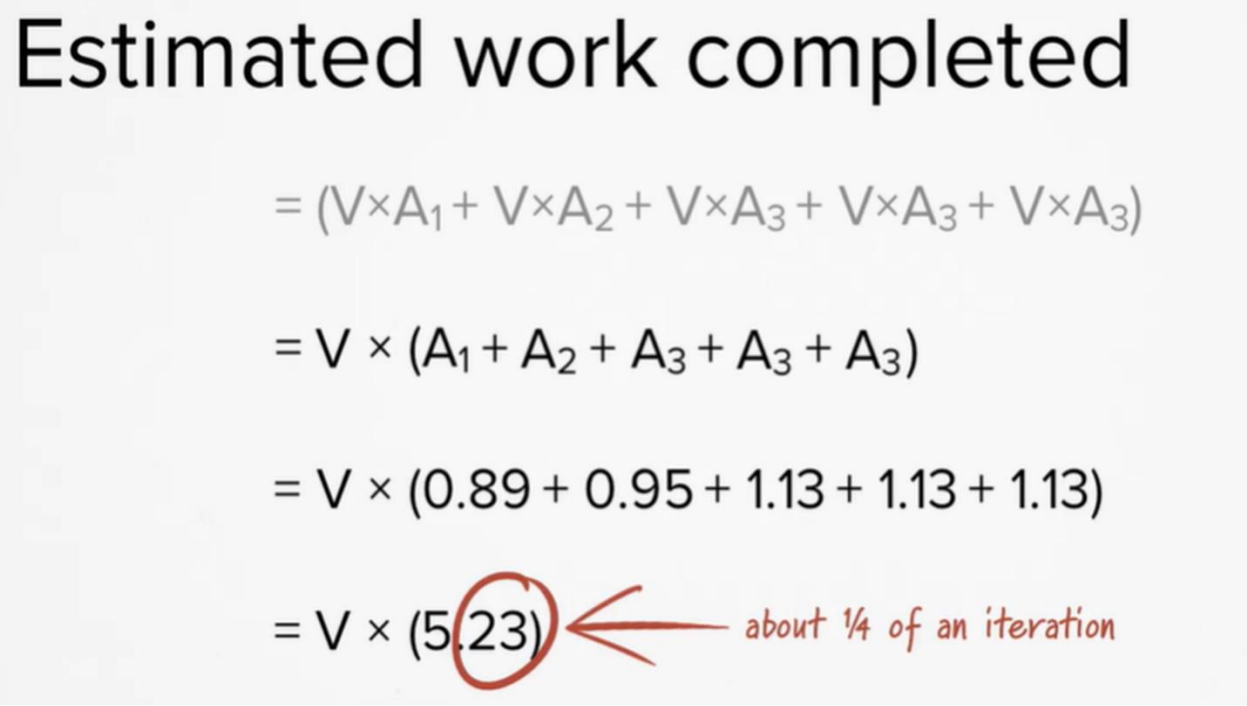


Suppose we are considering adding a 7th person to a 6 people project team about to undertake a 5 sprint/iteration project. We will average the velocities in each sprint/iteration and convert them to decimal values relative to 1. (-11% means 0.89, -5% means 0.95 and +13% means 1.13).

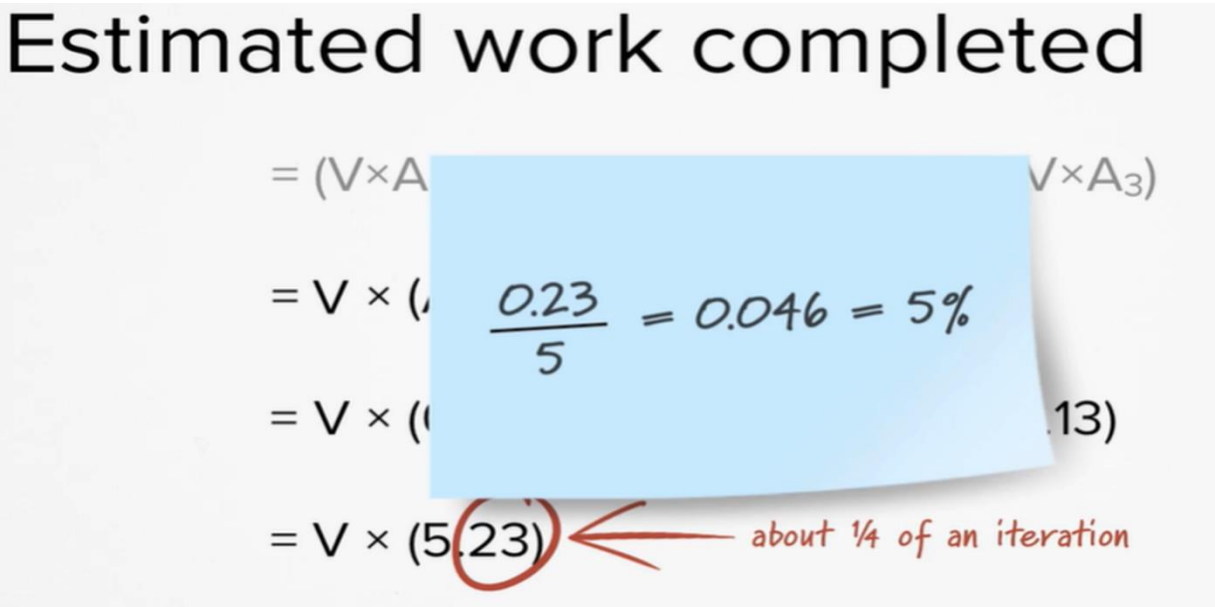
Impact of going from 6–7 people

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

Now let's use these 3 values to estimate the impact on the total amount of work the team will complete with the new 7th team member. Assuming that the project will be 5 sprints/iterations long.

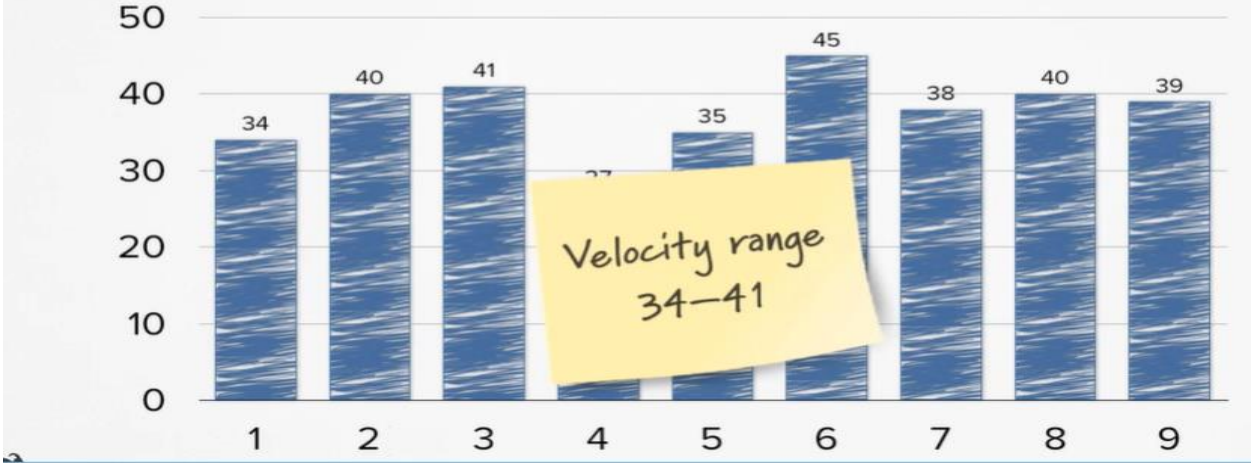


Therefore, adding an extra team member gives us 5.23 times velocity or extra ¼ of an iteration or 5% extra work done in 5 iterations. If we kept the team at 6 people, our work would be done at 5.0 times velocity.

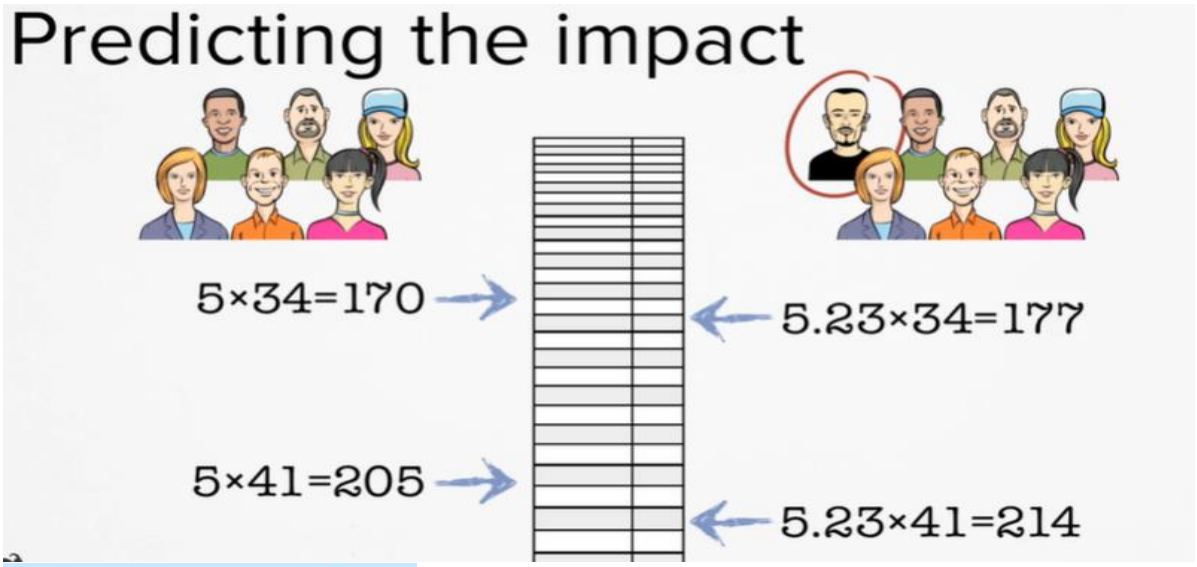


Now let's see how this impacts the product backlog, let's see the example from earlier of a team with Historical Velocity Data available.

Historical velocity



Predicting the impact



Is adding an extra person worth 7–9 points over five iterations?

7. Activities with Azure DevOps

Planning and estimation are crucial for smooth project execution, and Azure DevOps helps keep everything structured and transparent. Here is how the process works:

1. Creating and Maintaining the Product Backlog

- Think of the product backlog as the heart of your project. It's a dynamic list that evolves over time, ensuring teams stay aligned with business goals.
- Start by gathering requirements from stakeholders and converting them into **user stories**—clear, bite-sized descriptions of features or tasks.
- Regularly refine these backlog items to keep things **organized and prioritized**. This means breaking down vague ideas into actionable tasks.
- Remove outdated or unnecessary items to maintain a **lean and relevant backlog**. Add important details like **acceptance criteria** to define when a user story is considered complete.

2. Assigning User Stories and Tasks to Iterations

- Once the backlog is in good shape, it’s time to plan how work gets done within sprints (iterations).
- Prioritize items based on **business value** and the team’s **capacity** for the sprint.
- Assign user stories to upcoming **sprints**, ensuring that the team’s workload is **realistic and achievable**.
- Break down larger stories into **smaller, manageable tasks** so that they are easier to track and complete.
- Allocate tasks to team members based on their **skills, expertise, and availability**, ensuring a balanced workload.

3. Logging Estimation Points and Tracking Velocity

- Good estimation is key to realistic planning and avoiding overcommitment.
- Use **estimation techniques** to assign **story points** based on effort and complexity.
- At the end of each sprint, track how many **story points were completed**—this is the **velocity** (a measure of how much work the team can complete in a sprint).
- Use velocity trends to **predict future timelines** and improve estimation accuracy over time.
- Make sure every team member regularly updates their progress in Azure DevOps to keep tracking **accurate and up to date**.

4. Monitoring Sprint Progress Using Dashboards

- A well-monitored sprint helps teams stay on track and adapt quickly if things go off course.
- Set up **custom dashboards** in Azure DevOps to get a **real-time overview** of sprint progress.
- Keep an eye on **burndown charts, work items, and velocity reports** to understand how the sprint is progressing.

- If bottlenecks appear (e.g., too many tasks stuck in progress), address them quickly to **prevent delays**.
- Share **reports and insights** with stakeholders so they have clear visibility into progress without needing frequent updates.

5. Reviewing and Adjusting Sprint Commitments

- At the end of each sprint, take time to reflect and improve.
- Conduct a **Sprint Review** to highlight completed work and gather **feedback from stakeholders**.
- Hold a **Retrospective** to discuss what went well, what did not, and how to improve future sprints.
- Adjust **future sprint plans** based on real-world performance and insights from past sprints.
- If priorities shift, update the **backlog** accordingly to align with evolving **business needs**.

Please follow the document **"Standard Operating Procedure (SOP): Project Management in Azure DevOps"**, specifically **Section 4: Project Planning** and **Section 5: Project Planning**, for a detailed procedure.
Document Link:

8. Developer Contributions

- **Participate in backlog refinement and estimation sessions**: Provide accurate estimates and insights
- **Provide insights into technical feasibility and effort**: Ensure realistic planning
- **Break down tasks into manageable units**: Improve tracking and execution efficiency
- **Commit to deliverables within the sprint**: Ensure accountability and timely completion
- **Update task progress in Azure DevOps**: Maintain visibility of ongoing work

9. Tester Contributions

- **Define testable acceptance criteria**: Ensure completeness and quality
- **Identify dependencies and risks related to testing**: Highlight potential blockers
- **Estimate effort required for test case creation and execution**: Ensure accurate workload planning
- **Validate completed work against acceptance criteria**: Confirm alignment with requirements
- **Report defects and collaborate on resolutions**: Improve software quality

10. Planning Approach

VI. Sprint Planning (Short-Term Planning)

- Defines work to be completed in the next sprint (2-4 weeks).
- Ensures tasks are well-defined, estimated, and prioritized.
- Balances workload based on team capacity.
- **Key Benefit**: Keeps work structured and manageable.

VII. Release Planning (Mid-Term Planning)

- Covers multiple sprints and defines deliverables for a product release.
- Aligns development goals with business needs.
- Allows for adjustments based on feedback and market conditions.
- **Key Benefit**: Ensures a predictable and well-planned product rollout.

VIII. Roadmap Planning (Long-Term Planning)

- Provides a high-level vision of the project over several quarters or years.
- Defines major milestones, strategic goals, and dependencies.
- **Key Benefit**: Aligns team efforts with business strategy and long-term goals.

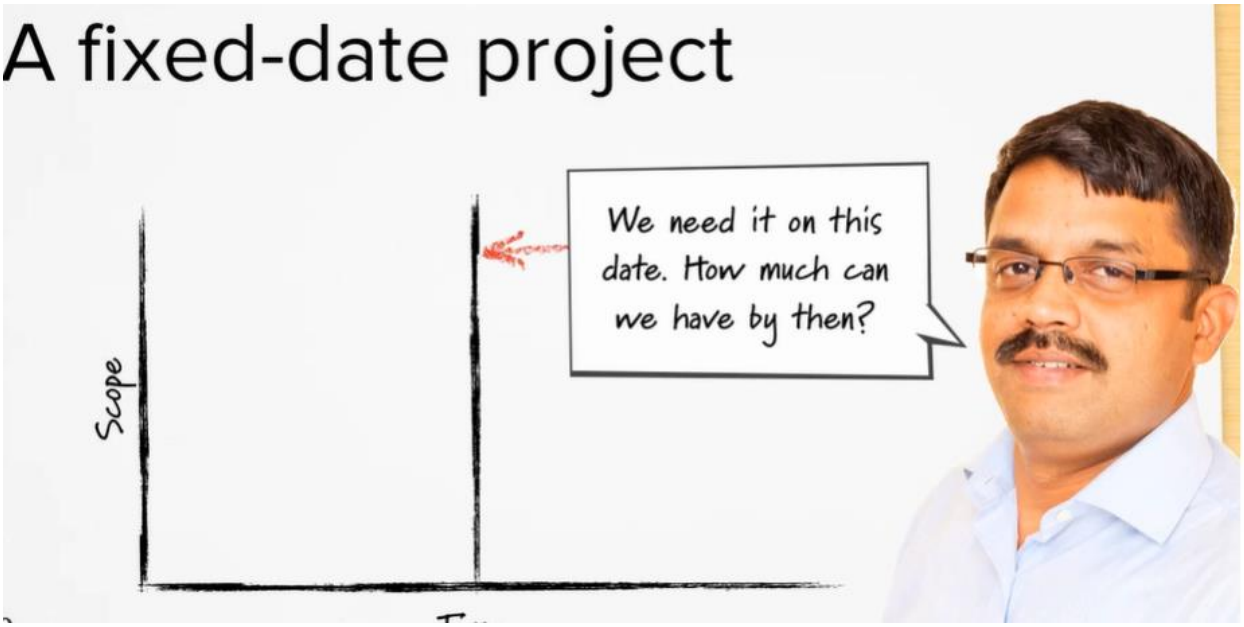
IX. Continuous Adaptation

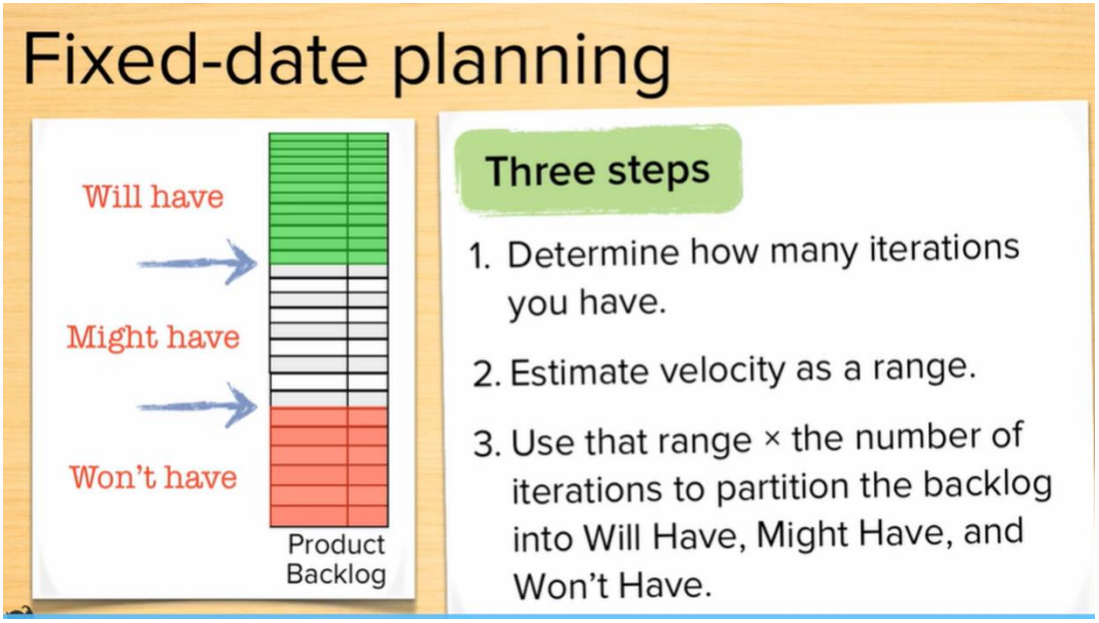
- Plans are regularly revisited and refined based on feedback and performance.
- Adjustments are made to accommodate changing priorities and risks.
- **Key Benefit**: Ensures agility and responsiveness to change.

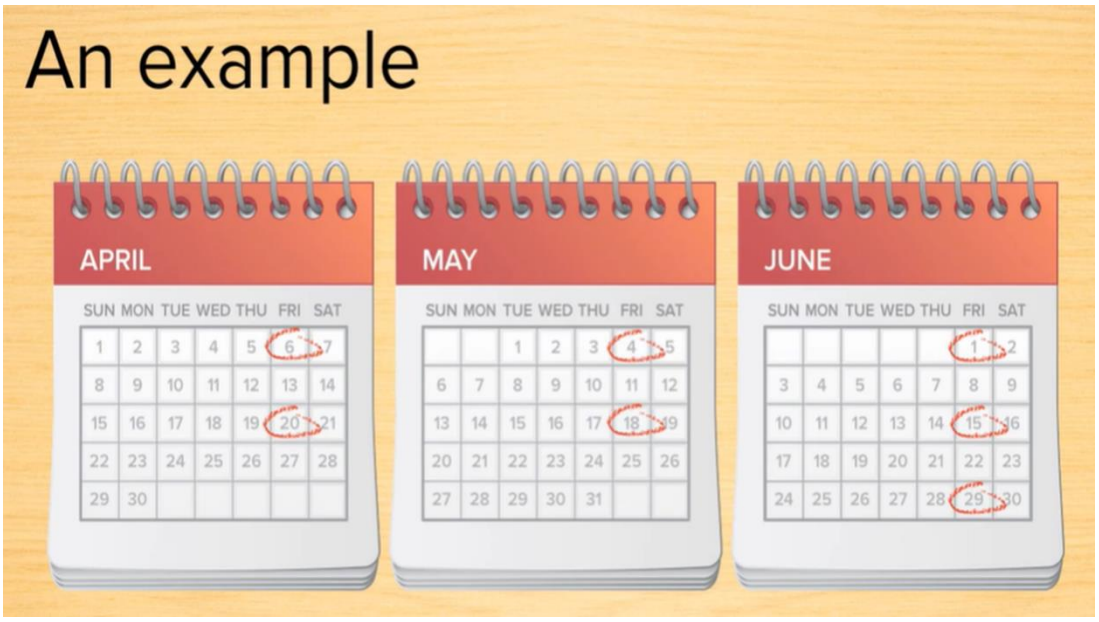
X. Re-Baselining

- It is the process of updating a project's original plan (scope, schedule, budget) when things change.
- Changes might happen due to scope changes, unexpected risks, or resource issues.
- The project's baseline (original reference point) is reviewed and reset.
- The new baseline is approved and used for tracking progress moving forward.
- **Key Benefit:** It helps keep the project on track by ensuring expectations are realistic and performance is measured against the most up-to-date plan.

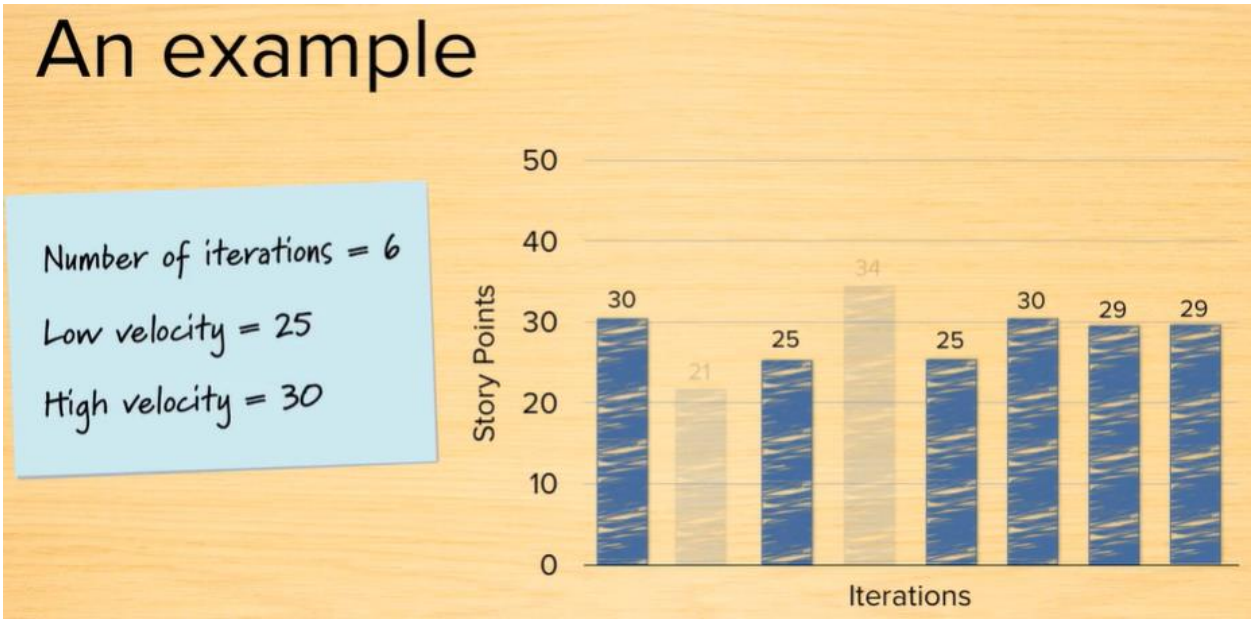
XI. Fixed Date Planning (Management and Client seeking estimations from team on how much of project Scope can be delivered within Fixed Date)



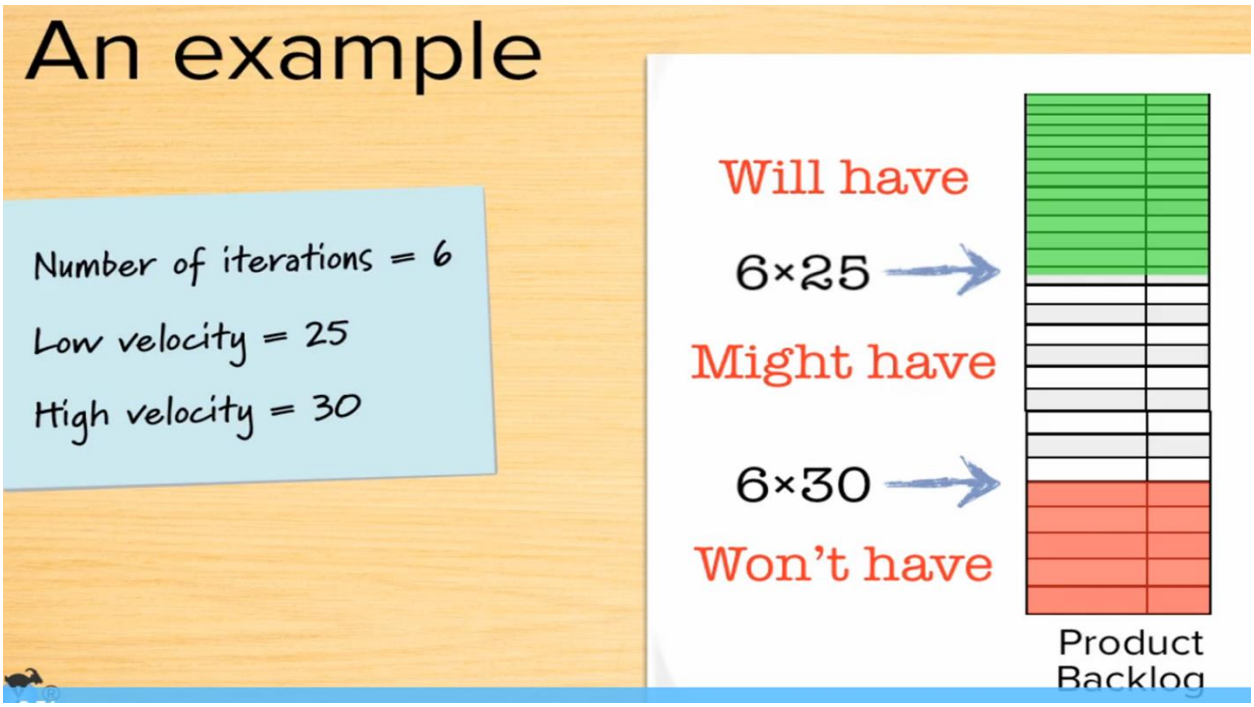




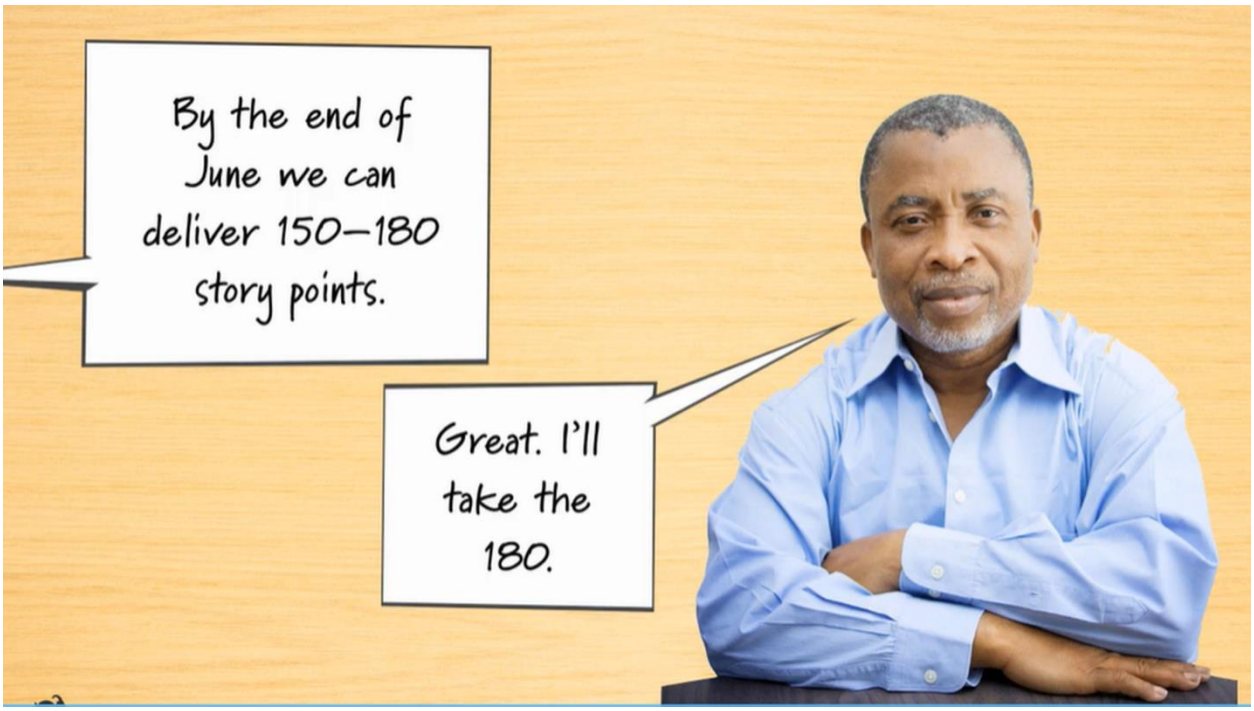
In this example Scenario, Client has given us till End of June to deliver as much of the scope as possible.



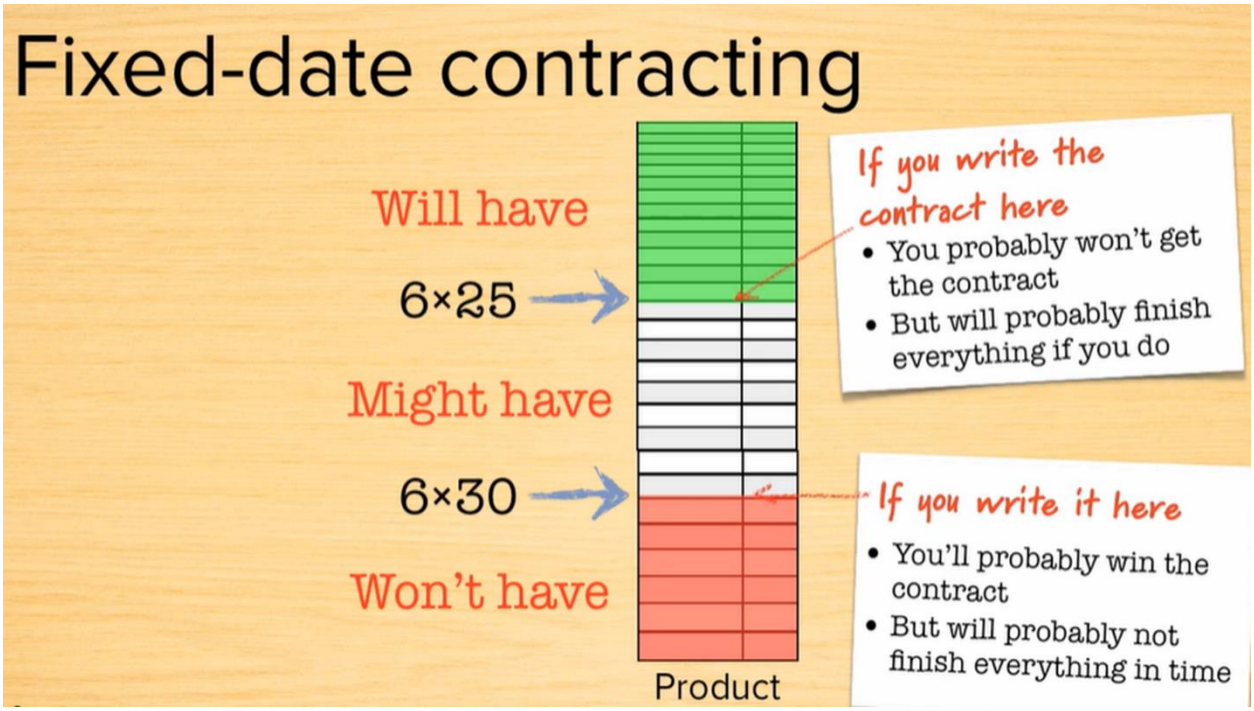
Therefore, we have 6 Iterations to deliver and a range of historical Velocity data. We have velocity data from 8 sprints so we will remove 1 highest and 1 lowest value. Giving us a Velocity range of 25 to 30 story points. Now we can proceed with segregating the product backlog into Will have, might have and Won't Have sections.



Historically, Internal Management and clients always expect commitment for more work. The product backlog gives us a range of **150 to 180 story points** to be completed within End of June or the next 6 sprints.



Fixed-date contracting

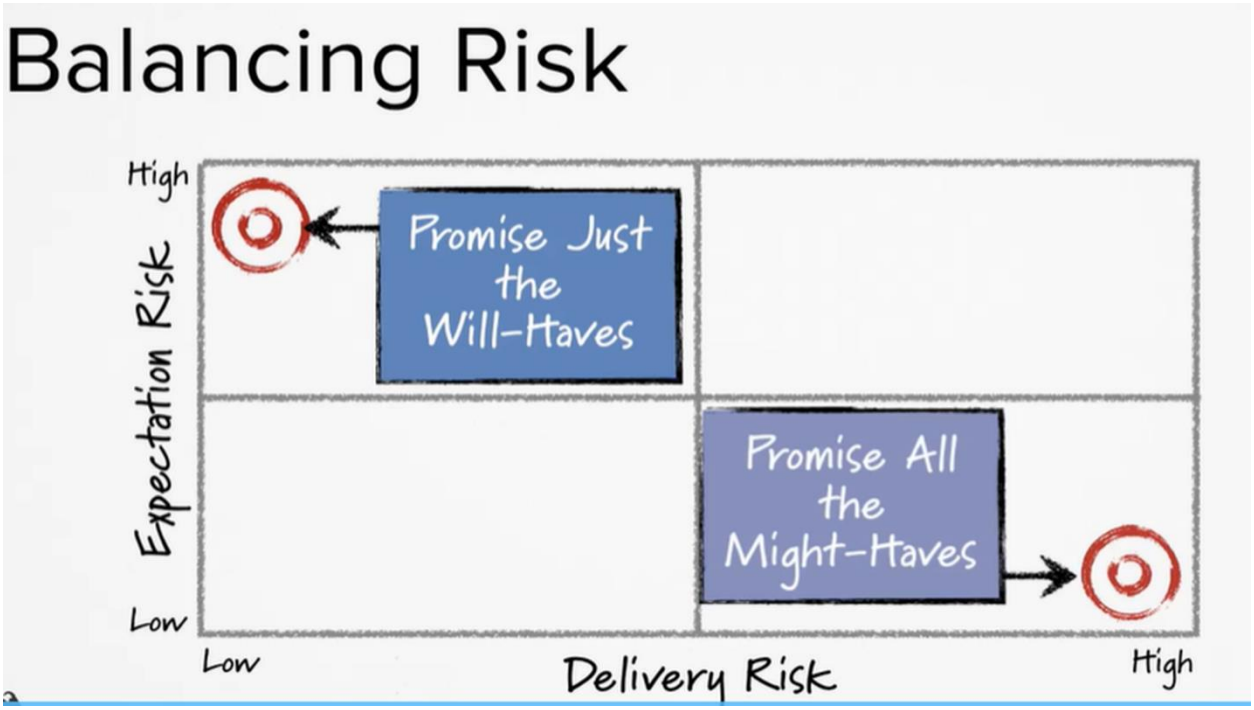


Based on this Range of Story Points, we have to balance the 2 types of risk.

Delivery Risk (Risk of Delivering what is promised)

VS

Expectation Risk (Not promising enough to meet Internal Management/Client expectation)

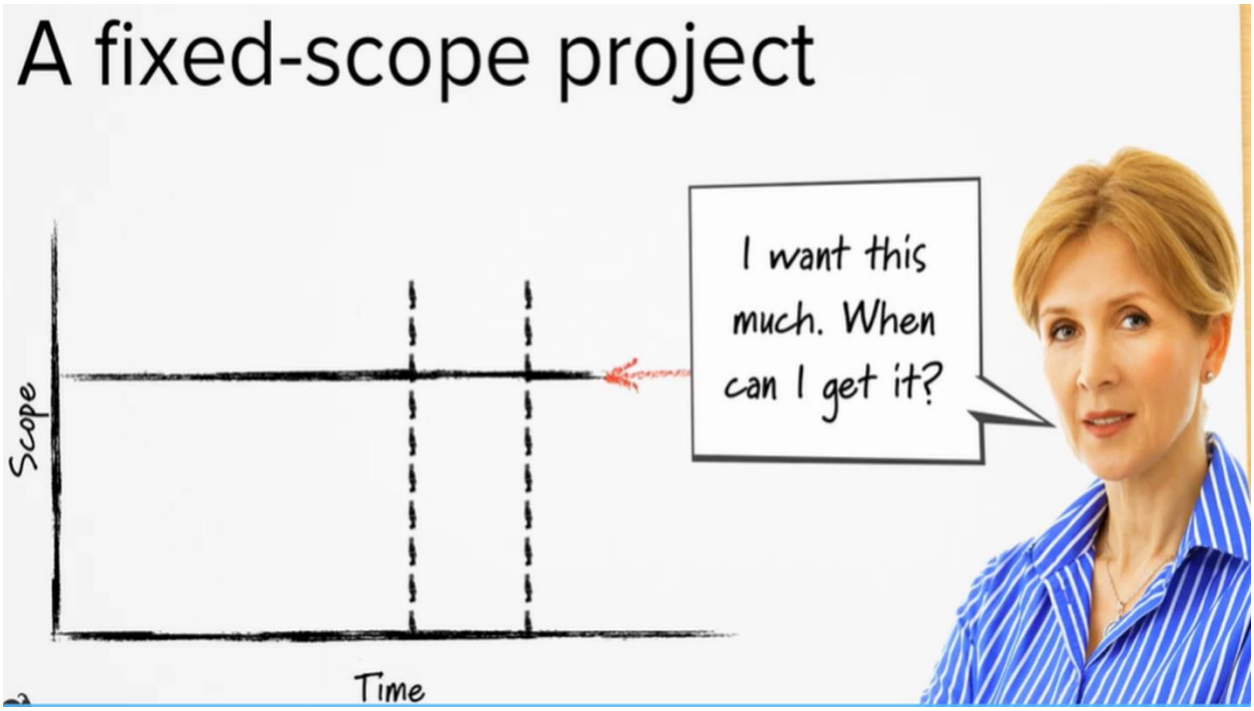


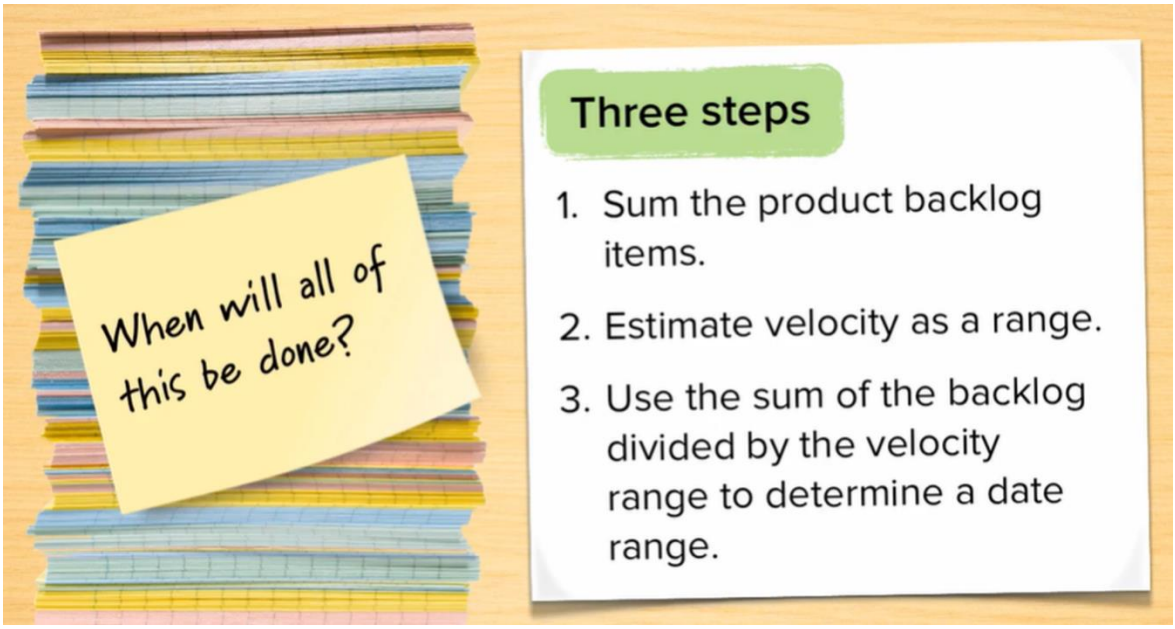
If we promise just the 'Will Have' part of the backlog, we are not taking much delivery risk, and we will face high expectation risk as team can almost certainly finish that much work from total scope. Internal management and clients will not be happy.

If we promise all the 'Might have' part of the backlog, we are not taking much expectation risk, but alot of Delivery Risk. Since this is being really optimistic about team performance, Internal management and Clients will be very happy initially, but risk of delivery will be very high.

With the math above, Project Managers and teams can know the truth about the project. Then we must find a balance between Delivery risk and Expectation risk and communicate something between 150 to 180 story points as your point estimate when forced to do so.

XII. Fixed Scope Planning (Management and Client seeking time estimation/Deadline from team for completing Fixed Scope)





To create our Fixed Scope Plan, we check our **Project Scope/product backlog** and add up all the estimates on the individual user stories present in the Project Scope/product backlog. We then take the velocity range for the team and divide the sum of backlog with the range.

What we know

- Project size = 120 points
- Velocity = 15–20

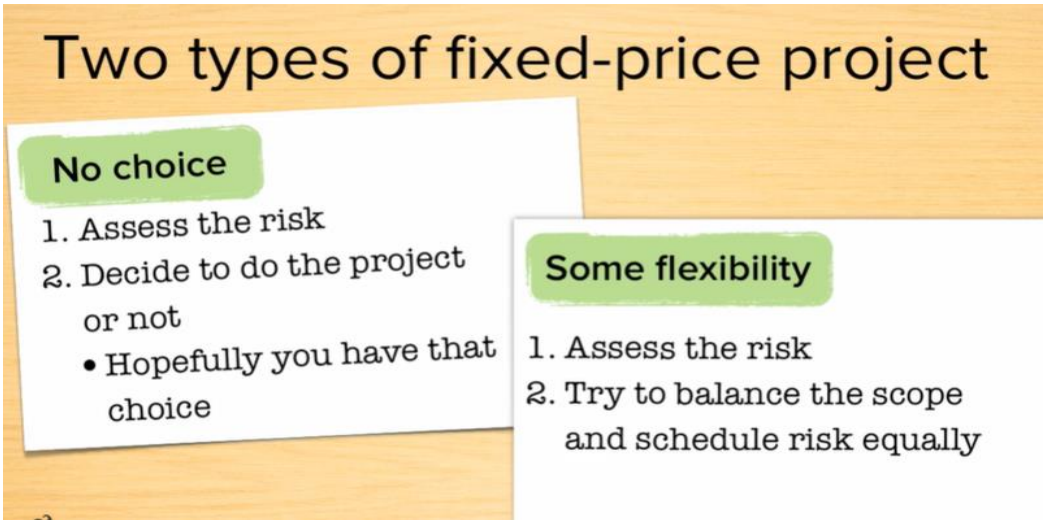
To Delivery the entire fixed scope, this project will take between 6 to 8 Sprints/Iterations.

120 ÷ 20 =

120 ÷ 15 =

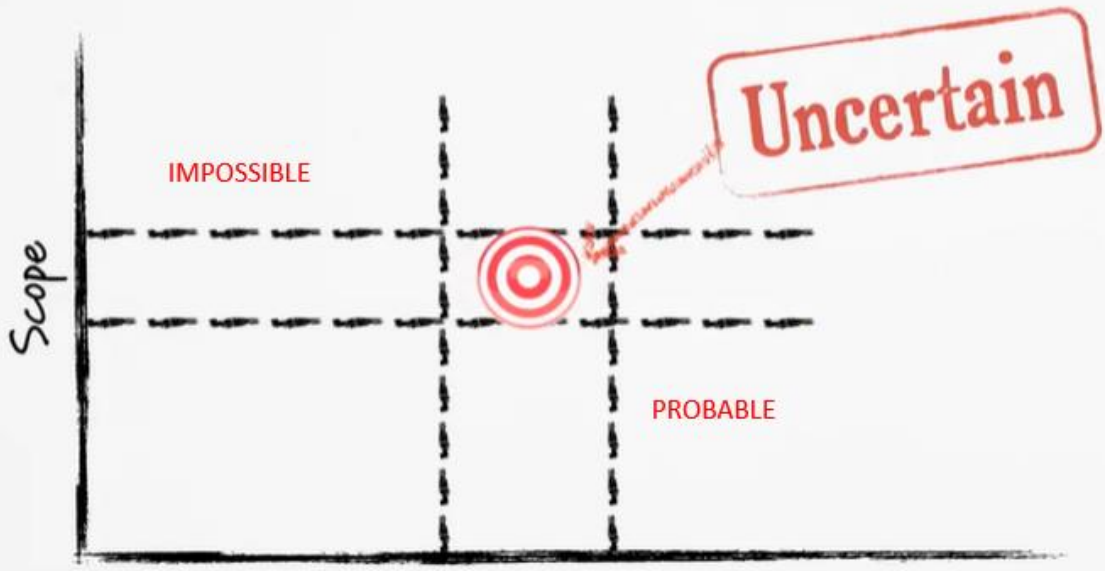
XIII. Fixed Price Planning (Management and Client Demand Fixed Scope to be Delivered by Fixed Date)

There are **2 types** of Fixed Price Projects.



Now we must do the Backlog Segregation to know how much work we can safely commit.

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

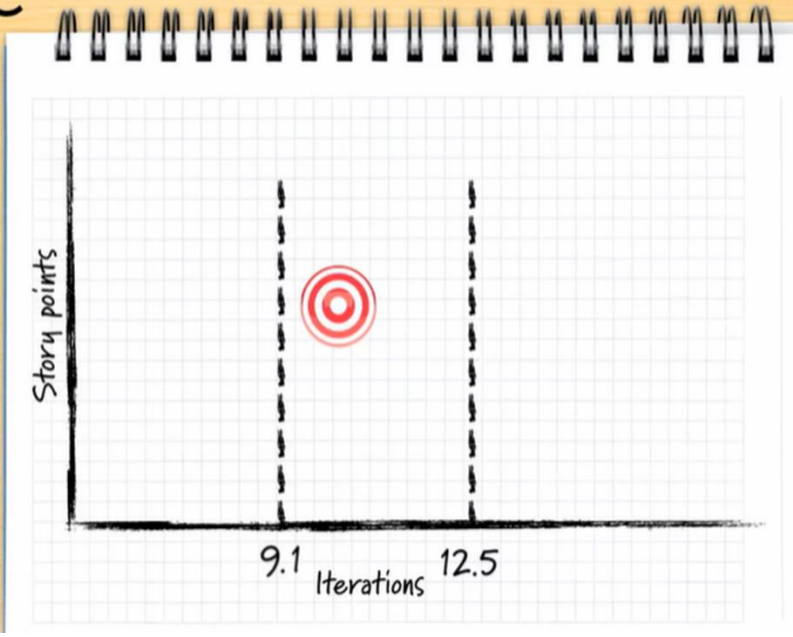
Deadline = 10 iterations

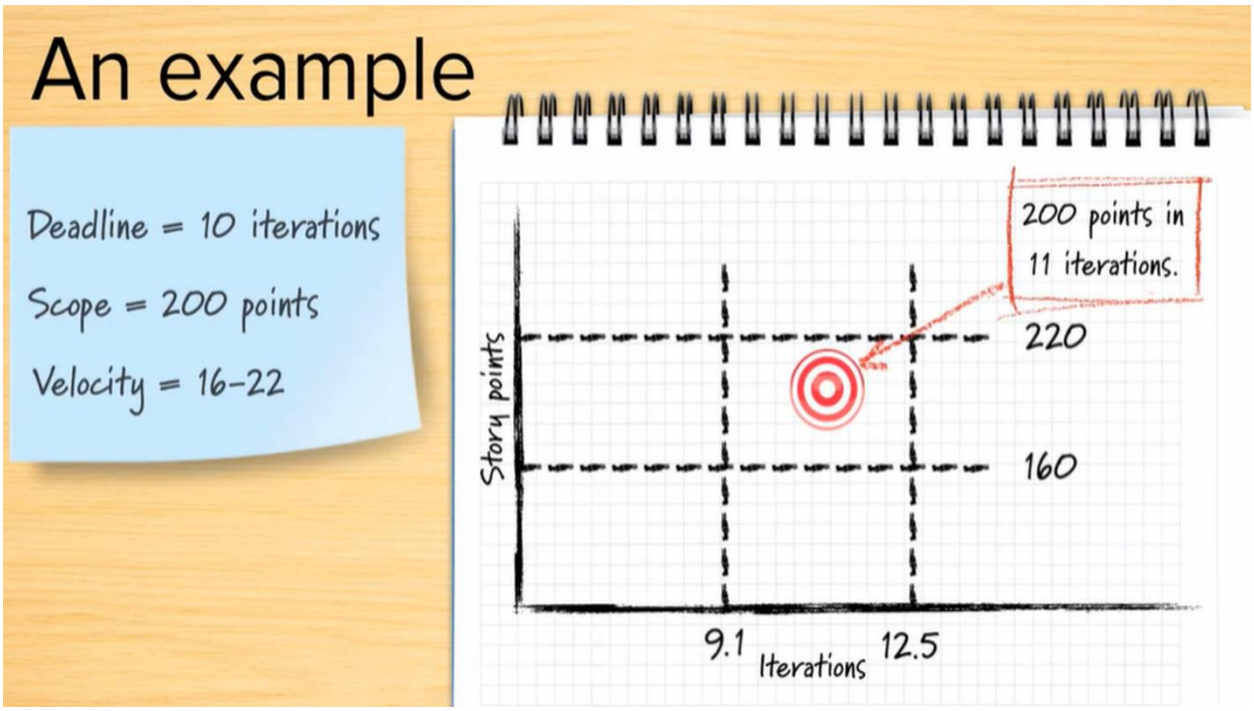
Scope = 200 points

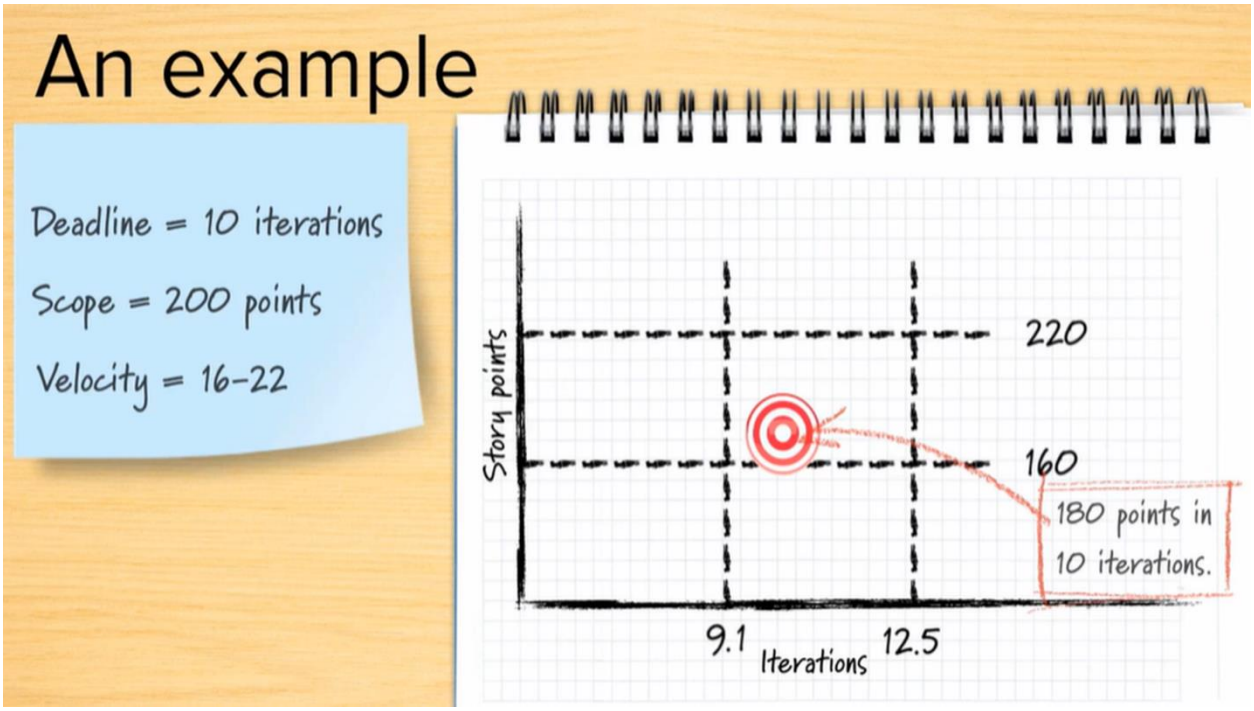
Velocity = 16–22

10x16 = 160

10x22 = 220







Therefore, we will commit to an iteration and story point number that is closer to the centre of the marked area. **Example- 200 points in 11 Sprints/Iterations or 180 points in 10 Iterations.**

11. Exit Criteria

A planning and estimation session is considered complete when:

- **User stories/tasks are estimated and prioritized**: Ensuring structured execution
- **The team agrees on the planned work for the sprint/release**: Confirming alignment on commitments
- **Risks and dependencies are identified and addressed**: Mitigating potential blockers
- **The plan is reviewed and approved by relevant stakeholders**: Ensuring buy-in and feasibility

12. Outputs / Work Products

- **Estimated Backlog**: User stories/tasks with assigned estimates to provide transparency
- **Sprint Plan**: Commitment on what will be delivered in the sprint to ensure focus
- **Release Plan**: Roadmap of deliverables across multiple sprints to maintain clarity
- **Capacity Plan**: Mapping of team availability and workload distribution for better planning
- **Risk & Assumption Log**: Documented potential risks and mitigations to improve decision-making

13. Acceptance Criteria or Conditions of Satisfaction

- **Clearly defined and measurable conditions for work completion**: Ensuring quality standards
- **Align with business requirements and user expectations**: Maintaining relevance and value
- **Reviewed and approved by stakeholders before development**: Confirming clarity
- **Validated by both developers and testers before closure**: Ensuring thorough testing and verification

By following these guidelines, teams can create realistic and reliable project plans, improving predictability, resource efficiency, and successful project delivery.

14. Relevant Stakeholders

- **Product Owner**: Defines and prioritizes the backlog for alignment with business needs
- **Scrum Master/Project Manager**: Facilitates the estimation and planning process for efficiency

- **Development Team**: Provides estimates and commits to the plan for execution
- **Business Stakeholders**: Aligns planning with business goals for strategic success
- **QA Team**: Ensures testability of estimated work to maintain software quality

15. References

- *Mike Cohn, Estimation & Planning* – Foundational principles of Agile estimation
- Agile Manifesto and Scrum Guide – Core guidelines for Agile methodology
- Azure DevOps Documentation – Technical documentation for implementation

16. Glossary

- **Velocity**: The amount of work a team can complete per sprint, based on historical performance
- **Story Points**: A unit of measure for estimating effort, rather than time
- **Sprint/Iteration**: A time-boxed iteration in Agile development for structured execution
- **Backlog Grooming**: The process of refining and updating the product backlog for clarity
- **Acceptance Criteria Or Conditions of Satisfaction**: Conditions that must be met for a story to be considered complete, ensuring quality

By following these guidelines, teams can create realistic and reliable project plans, improving predictability, resource efficiency, and successful project delivery.

- **Anchoring:** The tendency for estimates to luster around an irrelevant value.