Uniwersytet Wrocławski

Estimation Techniques

Presented by: Fady Yacoub
M.Sc. Data Science Student, University of Wrocław

Instructor: Prof. Kamil Matuszewski
University of Wrocław

November 2025

Agenda

1. Introduction
2. Core Concepts
3. Agile Estimation Techniques
4. From Estimation to Forecasting
5. Common Pitfalls and Biases
6. Best Practices
7. Summary

Introduction

- Purpose of Estimation in Agile
- Challenges of Traditional Estimation
- Agile Perspective on Estimation
- Key Principles

Purpose of Estimation in Agile

- Helps the team and stakeholders **forecast delivery timelines and plan releases.**
- Supports Sprint Planning by **estimating how much work can fit into a sprint**.
- Provides **a basis for roadmaps**.
- Helps **balance scope, time, and resources** for optimal outcomes.
- **Tracking actual effort vs. estimates helps improve team predictability.**

Purpose of Estimation in Agile

- **Reveals process inefficiencies.**
- Reveals areas where **understanding or accuracy can improve over time.**
- Provides **transparency to stakeholders** on how long features may take.
- Sets **realistic goals** to avoid overpromising and under-delivering.

Challenges of Traditional Estimation

- Estimates are often made **before full requirements are known**.
- **Changes in scope** make early estimates quickly **outdated**.
  - **Leads to unrealistic deadlines** and rigid commitments.
- Traditional methods (like hours or Gantt charts) **assume predictability**.
- Focus on exact numbers **creates a false sense of accuracy.**
- **Deviations from estimates are seen as failures instead of learning opportunities.**

Challenges of Traditional Estimation

- Poor adaptability to change as it **assumes stable scope** and requirements.
- Struggles with **changing priorities** common in Agile environments.
- Leads to delays and **rework when plans must be updated.**
- Hidden Costs and Waste:
  - **Time-consuming estimation meetings** and detailed documentation.
  - **Delays starting actual development** and feedback loops.

Agile Perspective on Estimation

- Estimation is **team-driven**, not management-imposed.
- Builds a **shared understanding** of the user story and its complexity.
- Agile favors **task-relative estimation** rather than trying to predict exact durations.
- Prioritize **high-value, low-effort stories**.
- Team works on **what delivers maximum customer impact first.**
- Retrospectives help compare estimates to outcomes for future improvement.
- A **feedback and learning loop,** not a rigid contract.

Key Principles Behind Agile Estimation

- Relative Estimation:
  - Compare tasks **against one another** instead of assigning exact hours.
- Estimation ≠ Commitment:
  - Estimates are forecasts, **not promises**.
  - Teams commit to **delivering value within a sprint**, not to perfect predictions.
  - **Encourages flexibility** and realistic expectation-setting.

Key Principles Behind Agile Estimation

- Progressive Elaboration
  - **Estimations are refined over time** as more information becomes available.
  - Early estimates are rough, details **increases closer to implementation.**
- Simplicity and Transparency
  - Keep estimation **lightweight and easy to explain.**
  - **Avoid complex formulas.**
- Estimates should consider **business value and technical risk**, not just size.

Core Concepts

- What is Effort Estimation?
- Relative vs. Absolute Estimation
- Story Points
- Ideal Time vs. Actual Time
- Velocity

What is Effort Estimation?

- The process of **predicting how much work is required to complete** a specific task.
- It’s about understanding the **relative effort compared to other tasks.**
- Helps the team plan and **prioritize work realistically.**
- Agile estimation uses **relative sizing.**
- Focuses on **complexity, uncertainty, and risk, rather than just time.**
- **Done collaboratively** by the whole development team to ensure shared understanding.

Relative vs. Absolute Estimation

Absolute Estimation	Relative Estimation
• Estimating **exact time or effort** needed to complete a task (8 hours, 3 days). • Common in **traditional project management**. • Focuses on **precision**. • Assumes **stable scope** and predictable conditions.	• Compares tasks **to one another** instead of predicting exact durations. • Uses scales like **Story Points, Fibonacci or T-shirt sizes**. [Discussed later] • Focuses on **complexity, effort, and uncertainty** rather than hours.
• **Hard to predict early** in the project. • **Encourages overconfidence and micromanagement.** • **Doesn't handle uncertainty or change well.** • Asks **"How long will it take?"**	• Easier for teams to agree on relative sizes. • Adapts well to evolving understanding and scope changes. • Asks **"How big is it compared to what we've done before?"**

Relative vs. Absolute Estimation: Example

Task	Absolute Estimation	Relative Estimation
Simple Login Page	6 hours	3 Points
Login + Social Media Integration	12 Hours	6 Points
Login + MFA + Role-based Access	24 Hours	12 Points

Story Points

- A way for teams to **compare one story's difficulty to another.**
- Capture three dimensions of effort:
  - **Complexity**: how difficult the work is.
  - **Risk/Uncertainty**: how much is unknown or likely to change.
  - **Effort**: how much work is required (coding, testing, reviewing, etc.).
- Common Scales Used
  - Fibonacci sequence (1, 2, 3, 5, 8, 13, 21...).
  - Modified Fibonacci (1, 2, 3, 5, 8, 13, 20, 40, 100).
  - T-shirt sizes (XS, S, M, L, XL)

Story Points: Example

Task	Story Points	Reason
Simple Login Page	3 Points	Simple feature Low risk
Login + Social Media Integration	6 Points	Moderate complexity API Integration
Login + MFA + Role-based Access	12 Points	High complexity Multiple Flows

Ideal Time vs. Actual Time

- Ideal Time: Represent the amount of **uninterrupted time it would take to complete a task** if nothing else interfered (meetings, context switching, or delays).
- Actual Time: **Real-world time** including meetings, breaks, interruptions, and context switching.
- **Easy to understand for teams used to traditional estimation.**
- Limitations:
  - Can **reintroduce time pressure** if misunderstood as commitments.
  - **Reduces focus on complexity and uncertainty**.

Velocity

- **The amount of work** (usually in story points) **a team completes in a single sprint.**
- Predicts **how much work can be delivered next sprint** (planning).
- Tracks **progress and stability over time.**
- Helps identify trends such as overcommitment.
- Velocity is team-specific and **should not be compared between teams.**
- **It’s not a performance metric but a planning tool.**

Velocity: Example

- Sprint 1: 30 story points completed
- Sprint 2: 28 story points completed
- Sprint 3: 32 story points completed

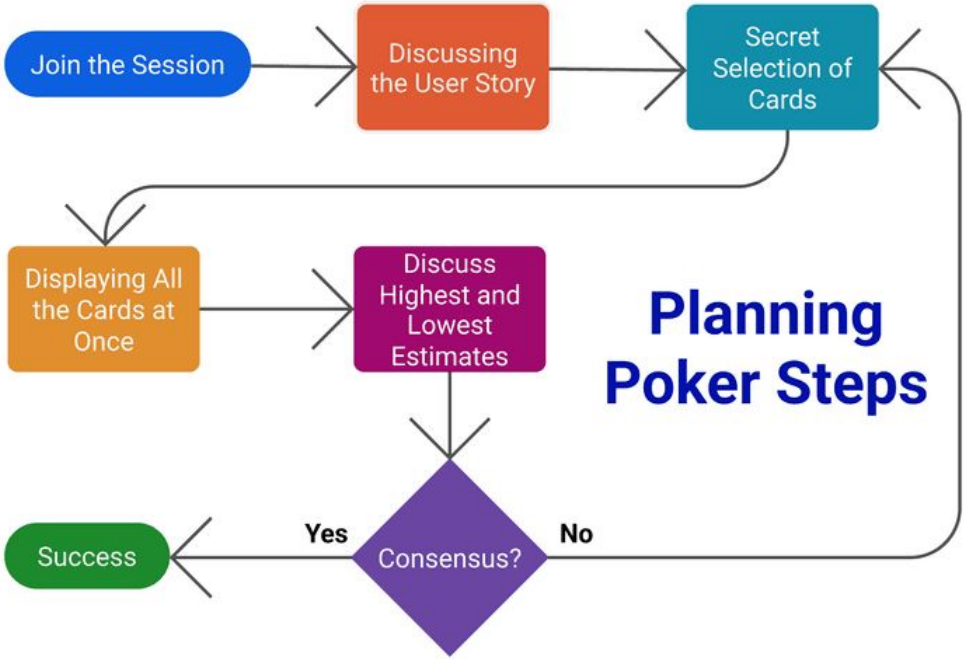
- **Average velocity = 30 story points per sprint**

Agile Estimation Techniques

- Planning Poker
- T-Shirt Sizing
- Bucket System
- Affinity Estimation
- Dot Voting
- Three-Point Estimation

Planning Poker

- **A consensus-based estimation technique**.
- **Discussion between highest and lowest estimate.**
- Used during s**print planning** or backlog refinement.
- Best suited for small-to-medium stories.



Planning Poker

Pros	Drawbacks/Limitations
Encourages **team discussion and knowledge sharing.**	Time-consuming for large backlogs.
Fast and simple to facilitate (requires only cards or a digital tool).	May lead to pressure for consensus instead of honest agreement.
	Quality depends on team maturity and story clarity. **i.e. too ideal!**

T-Shirt Sizes

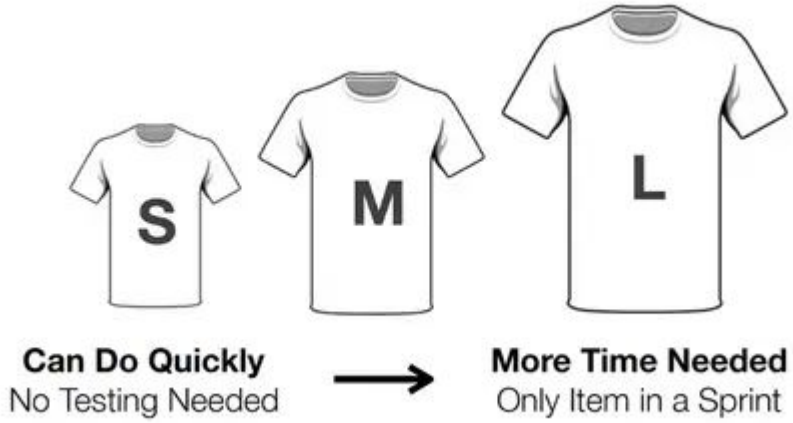
- A simple **relative estimation technique** used to express the size or effort of a user story without assigning numbers.
- **Each story is assigned a size label instead of numeric values.**
- Teams compare new stories with known examples
  - This feels like an M, similar to the login page feature.
- Can later be mapped to Story Points (S = 3 pts, M = 5 pts…).

T-Shirt Sizing

Pros	Drawbacks/Limitations
Fast and intuitive	Too coarse for sprint-level planning or velocity tracking.
Useful for non-technical stakeholders (managers, product owners).	**Inconsistent across teams** unless calibration stories are defined.
Can be easily converted into Story Points later.	**Must later be converted to numerical values for forecasting.**

T-Shirt Sizing: Example

- XS: Fixing a typo
- S: Adding a validation rule
- M: Implementing a new form
- L: Integrating third-party API
- XL: Refactoring an entire module

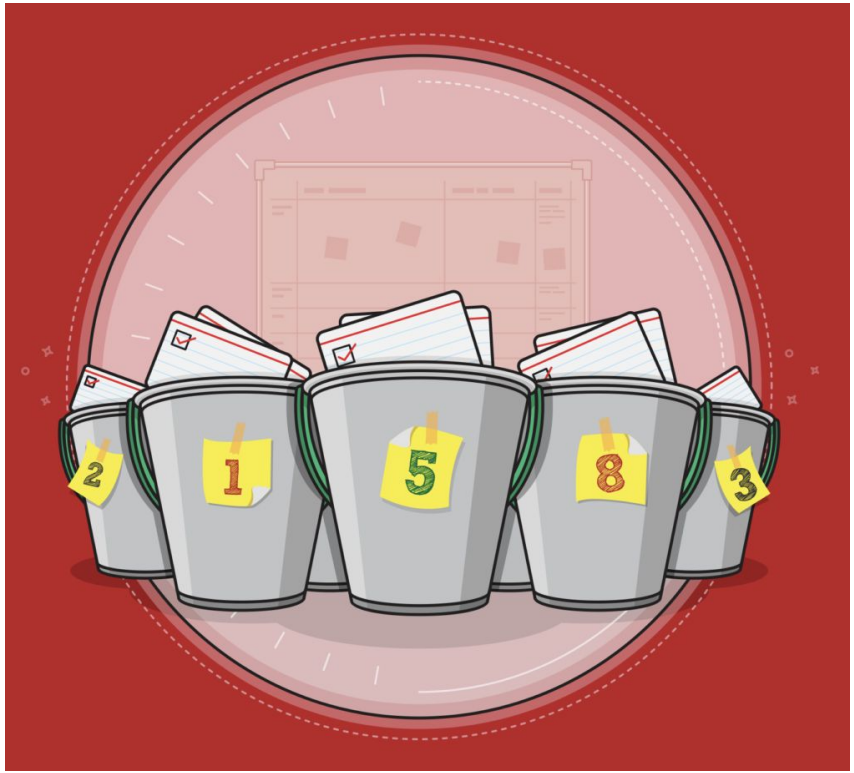


Bucket System

- A **group-based estimation technique** designed for rapid estimation of large backlogs.
- User stories are sorted into predefined “buckets” representing relative effort or complexity.
- **Achieve fast consensus across many items (50–200+) without individual discussion for each.**
- Used when relative comparison matters more than numeric precision.

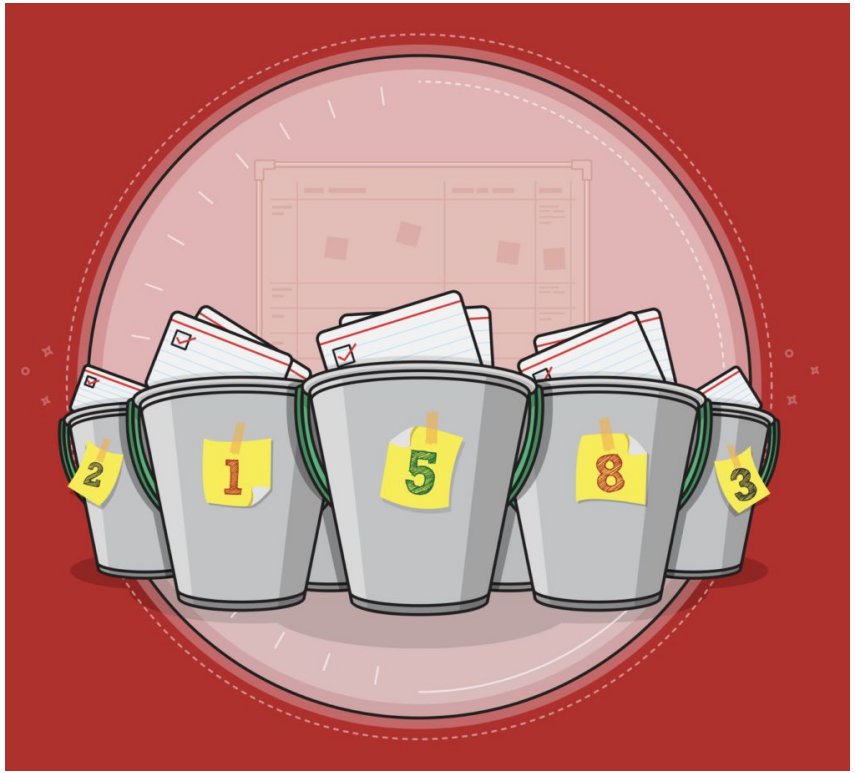
Bucket System

- Create buckets (1, 2, 3, 5, 8 Fibonacci-like).
- **Select a few reference stories and assign them to appropriate buckets.**
- Place remaining stories into **buckets based on similarity in effort/complexity**.
- **Team reviews outliers** and reassigns if necessary.
- We could convert buckets to story points later.



Bucket System: Example

- Bucket 1: Very small (typo fix)
- Bucket 5: Moderate feature (CRUD form)
- Bucket 13: Large story (new module or screen)
- Bucket 40: Epic-level (requires refactoring)



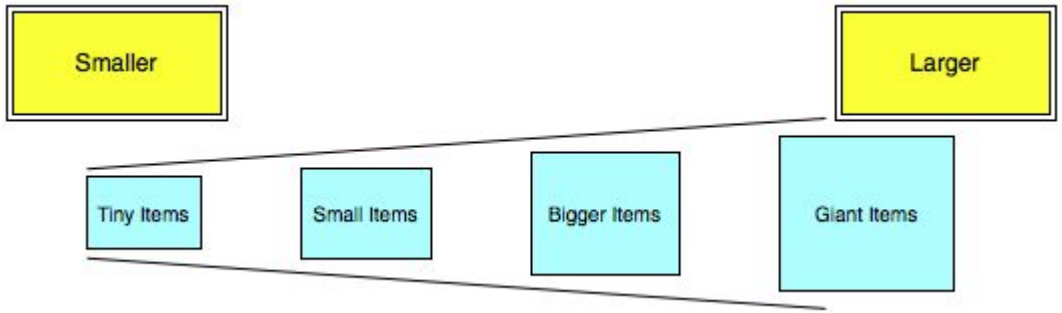
Bucket System

Pros	Drawbacks/Limitations
Very fast.	Less precise than Planning Poker.
Encourages collaboration and broad perspective.	**May overlook dependencies.**
Suitable for large-scale planning sessions.	Requires strong facilitator to maintain pace and focus.

Affinity Estimation

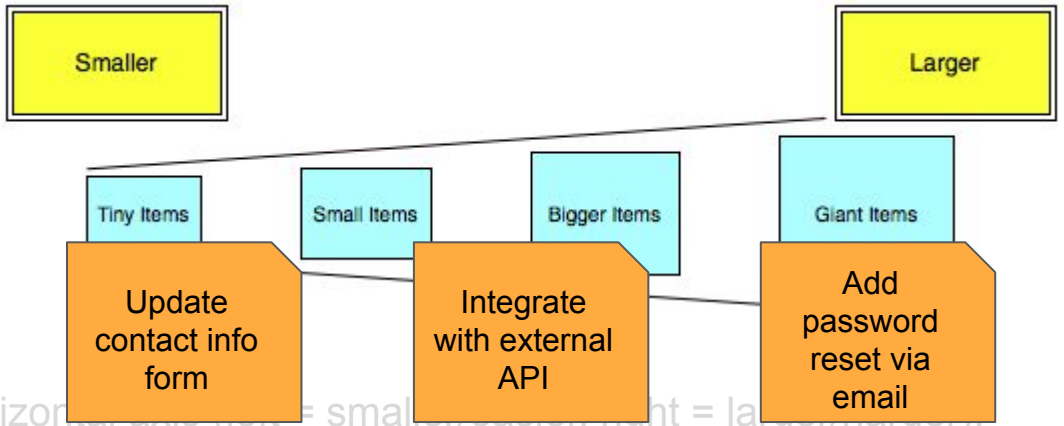
- A collaborative, **visual estimation technique.**
- Teams sort user stories by **relative size** through grouping and comparison.
- **Rather than assigning numeric values immediately.**
- To achieve **quick consensus** on large sets of stories.
- To promote shared understanding and **avoid long estimation debates**.

Affinity Estimation



- Create a horizontal axis (left = smaller/easier, right = larger/harder).
- **Team silently places stories on the scale according to perceived effort.**
- After all items are placed, **team reviews and discusses misplacements**.
- Adjust positions until relative order reflects team agreement.
- We can map positions to numeric story points.

Affinity Estimation: Example



- Create a horizontal axis (left = small, right = large).
- **Team silently places stories on the scale according to perceived effort.**
- After all items are placed, **team reviews and discusses misplacements**.
- Adjust positions until relative order reflects team agreement.
- We can map positions to numeric story points.

Affinity Estimation

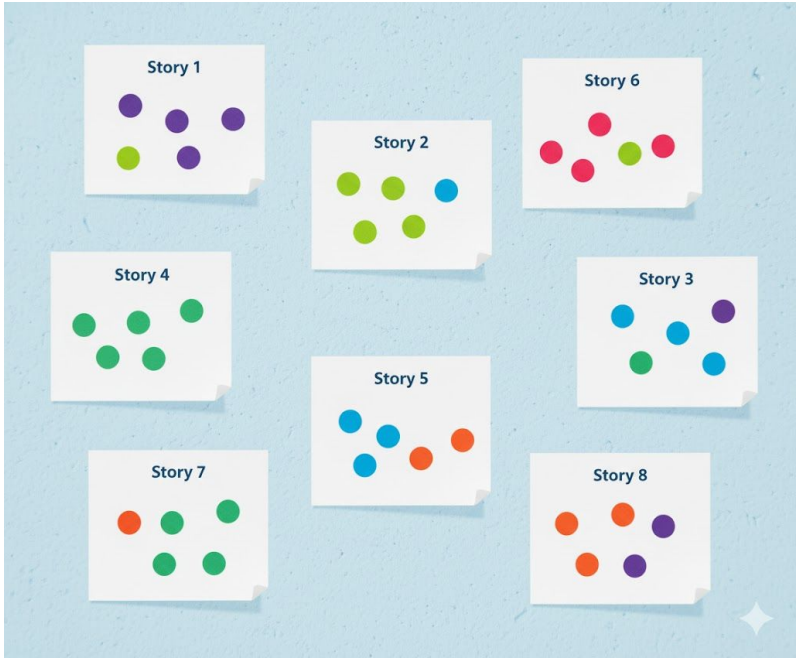
Pros	Drawbacks/Limitations
Fast and democratic	**Less precise** than Planning Poker.
Reduces anchoring bias through silent initial placement. **[revisited later]**	**Can be biased by dominant voices during group adjustments.**
Excellent for cross-functional teams with mixed expertise.	**Not suitable for vague items.**

Dot Voting

- A simple **prioritization and estimation** method.
- Team members **allocate a fixed number of votes (dots)** to items they believe are most significant, complex, or effort-heavy.
- Focus: team perception of value or difficulty, not numeric precision.
- Purpose:
  - To identify **consensus quickly among** many items.
  - To **highlight priorities or complex stories** requiring further discussion.

Dot Voting

- Present all backlog items.
- Each participant **receives N dots/votes** (usually 3–5).
- Members **distribute votes among items**, all on one item or spread out.
- **Count dots**: items with more votes **represent higher perceived value**, effort, or uncertainty.



Dot Voting

Pros	Drawbacks/Limitations
Fast and democratic	**Not a full estimation method** (gives ranking, not effort values.)
Focuses on what matters most	**Prone to popularity bias** (people follow visible trends).
Narrow down items for deeper estimation	Should be combined with another technique for quantitative results.
Useful for prioritization, risk identification, or complexity perception.	

Three-Point Estimation

- **A quantitative estimation technique** that models uncertainty by considering three scenarios for each task:
  - **Optimistic (O)**: Best-case scenario with minimal issues.
  - **Most Likely (M)**: Typical case under normal conditions.
  - **Pessimistic (P)**: Worst-case scenario accounting for risks and delays.
- Provides a **more realistic estimate** by incorporating uncertainty and risk.
- Used for **scientific, engineering, or research-oriented** Agile projects where precision matters.

Three-Point Estimation

- Identify task or story to estimate.
- Determine O, M, and P based on team experience.
- Compute expected value (E) and optionally variance.
- Aggregate estimates for total project/sprint forecasting.

Weighted Average

Emphasizes the **most likely** scenario while factoring in uncertainty.

$$E = \frac{O + 4M + P}{6}$$

Standard Deviation

To estimate confidence interval

$$\sigma = \frac{P - O}{6}$$

Three-Point Estimation: Example

- Values for estimating a task like login page:
  - Optimistic (O): 2 days
  - Most Likely (M): 4 days
  - Pessimistic (P): 8 days

$$E = \frac{2 + 16 + 8}{6} = \frac{26}{6} = 4.33$$

$$\sigma = \frac{8 - 2}{6} = 1day$$

$$\lceil 4.33 \rceil = 5$$

Three-Point Estimation: Example

- Values for estimating a task like login page:
  - Optimistic (O): 2 days
  - Most Likely (M): 4 days
  - Pessimistic (P): 8 days

$$E = \frac{2 + 16 + 8}{6} = \frac{26}{6} = 4.33$$

$$\lceil 4.33 \rceil = 5$$

$$5 \pm 1 \; days$$

$$\sigma = \frac{8 - 2}{6} = 1day$$

Three-Point Estimation

Pros	Drawbacks/Limitations
Accounts for **uncertainty** instead of a single-point guess.	**Time consuming**
Useful for complex or uncertain stories.	Relies on accurate judgment for O, M, and P values.
	Not intuitive for non-technical stakeholders.

	Planning Poker	T-Shirt Sizing	Three-Point
Use	Sprint estimation	Early backlog sizing	Risk-aware forecasting
Basis	Story points (Fibonacci)	Size buckets (XS-XL)	O, M, P scenarios
Speed	Medium	Fast	**Slow**
Accuracy	Medium to High	Low to Medium	High
Best For	**Teams with shared understanding**	**Quick early planning**	**Complex, uncertain tasks**
Pros	Encourages discussion & consensus	**Simple, visual, fast**	Models uncertainty & variance
Cons	Time-consuming for many stories	**Too coarse for tracking**	**Needs data & math effort**

Break

From Estimation to Forecasting

- Difference between Estimation and Forecasting
- Role of Historical Data
- Handling Uncertainty and Risk
- Continuous Refinement and Re-Forecasting

Difference Between Estimation and Forecasting

Estimation	Forecasting
The **process of evaluating the effort**, complexity, or size of individual tasks or user stories. It answers “How big or difficult is this work?”	The process of **predicting when work will be completed**, based on actual performance data (e.g., velocity). It answers “When will this work be done?”
Estimates are inputs to forecasting.	Combines **estimates with team velocity, capacity, and empirical data** to produce realistic delivery timelines.
More of speculations.	**Evolves empirically: using real sprint data instead of speculation.**

Role of Historical Data

- **Past performance records** including completed story points, sprint velocity…
- Helps the team **calibrate story points by comparing new work to similar completed stories.**
- Reduces **guesswork and bias**, especially for recurring or comparable tasks.
- **Reveals patterns of delays, bottlenecks, or rework.**
- Example: *The last 5-point API story took 2 days; this new one feels similar.*

Handling Uncertainty and Risk

- Agile tracks forecasts **with variability.**
- **Large stories carry more uncertainty.**
- Splitting **big stories into smaller**, clearer ones reduces estimation risk.
- **Teams may include capacity buffers** (like 80% of total velocity) to account for unknowns.
- **Estimates are refined iteratively** as more information becomes available.
- **Teams use retrospectives and historical data to improve estimation accuracy over time.**

Continuous Refinement and Re-Forecasting

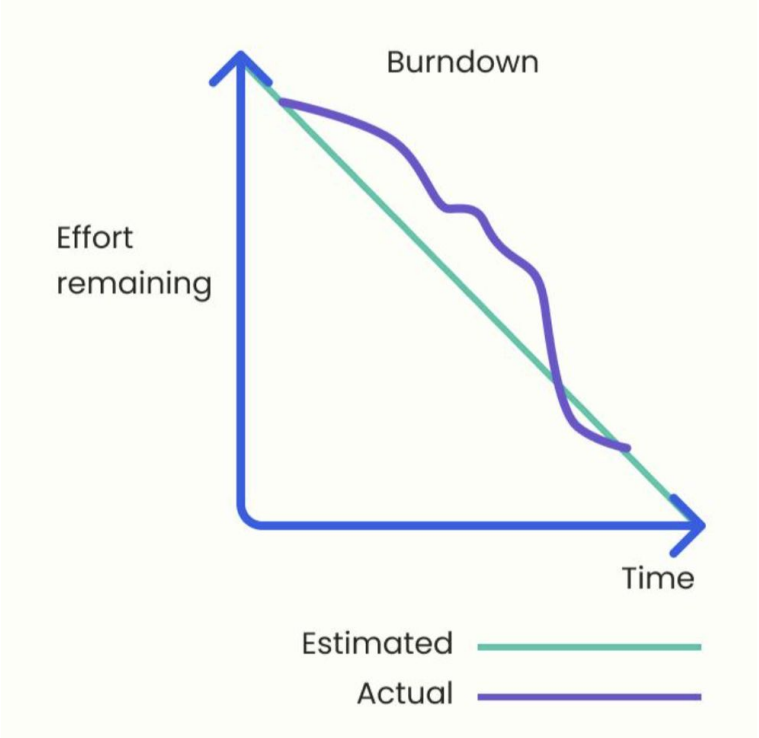
- Estimation **is not a one-time event, it's a living process.**
- Continuous refinement means **regularly revisiting and updating estimates as new information emerges.**
- Re-forecasting is the process of **adjusting delivery predictions** based on updated estimates, actual velocity, or changing priorities.
- Requirements evolve, risks surface, and team capacity fluctuates.
- **Without refinement, estimates quickly become outdated or misleading.**
- Regular updates maintain transparency and trust with stakeholders.

Continuous Refinement and Re-Forecasting

- When It Happens
  - **Backlog Refinement**.
  - **Sprint Reviews & Retrospectives**: compare estimates vs. actual outcomes.
  - **Major Changes**.
- Enables early detection of slippage or unrealistic goals.

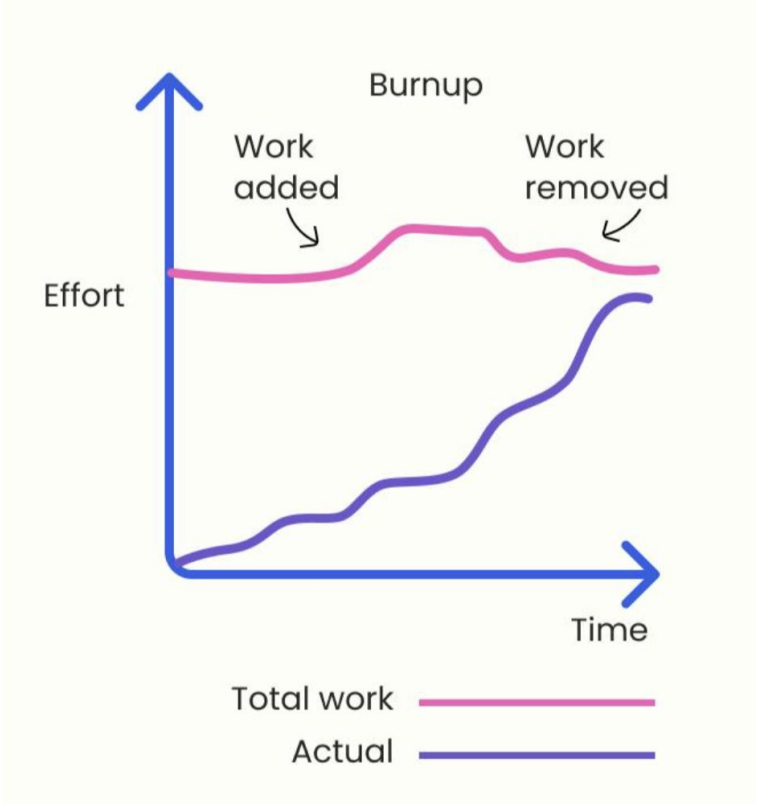
Burn-Down Charts

- **Shows how much work remains in a sprint or project over time.**
- Helps identify **if the team is on pace to complete planned work.**
- The ideal trend line shows expected progress
- The actual line reveals deviations or slowdowns.
- A flat or rising burn-down indicates blockers, scope changes, or underestimated effort.



Burn-Up Charts

- Shows how much work has been completed **toward the total scope**.
- Displays both total scope and completed work lines, making scope changes visible.
- **Useful for release forecasting based on team velocity and total story points.**
- Provides a positive visual cue: **progress rises instead of falls,** motivating the team.



Common Pitfalls and Biases

- Common Pitfalls
- Cognitive Biases in Estimation

Common Estimation Pitfalls

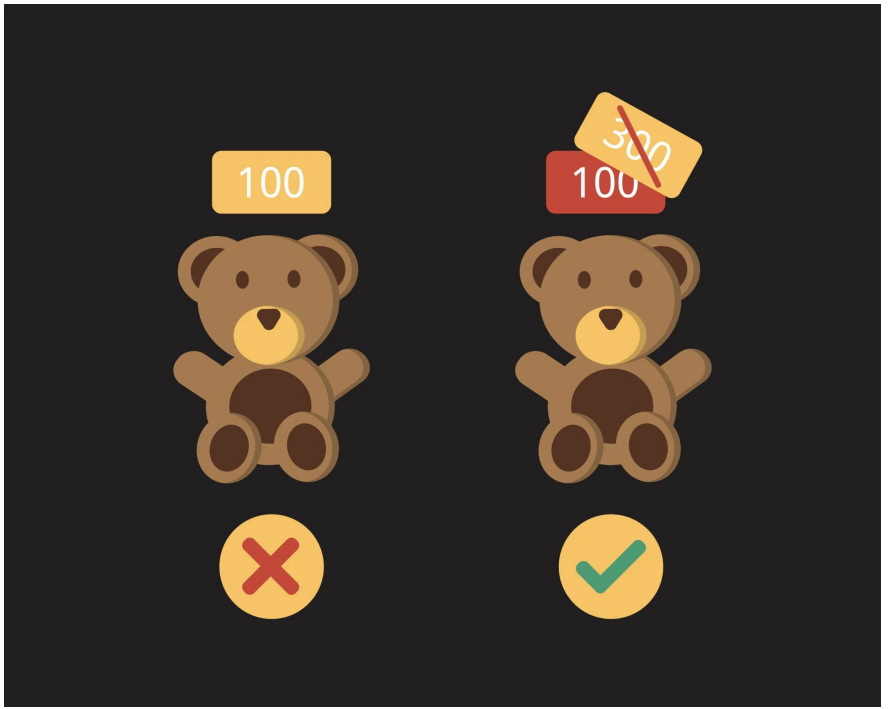
- Treating estimates as **commitments instead of forecasts** creates pressure.
- Spending too much time **chasing precision estimation.**
- **Ignoring uncertainty** and risks leads to underestimation and hidden blockers.
- **Making estimates individually** rather than collaboratively.
- **Comparing velocity across teams.**

Common Estimation Pitfalls

- Failing to **use historical data** prevents learning and improvement over time.
- Using **estimates for individual performance evaluation** creates fear and distorts results.
- Not splitting large stories:
  - Hides complexity
  - Increases delivery risk.
- Forgetting to include non-development tasks (testing, documentation, reviews) causes underestimation.

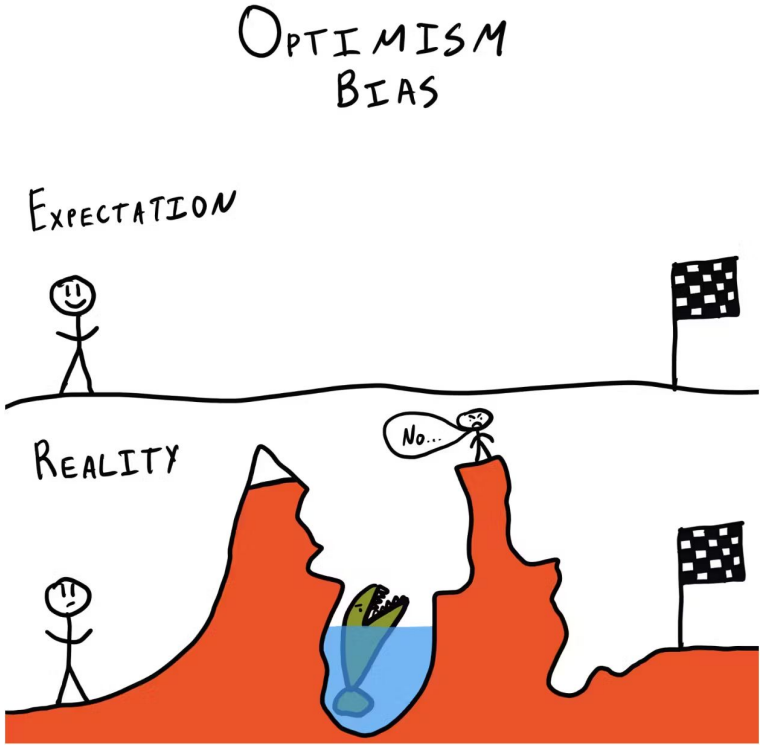
Anchoring bias

- Relying **too heavily on the first number mentioned** (an early estimate) influences later judgments.
- Even early estimates can distort later discussions.



Optimism bias

- Planning fallacy: Believing tasks will take less time than they actually do despite past evidence.
- Optimism bias: **Assuming everything will go smoothly**, leading to underestimated effort or duration.

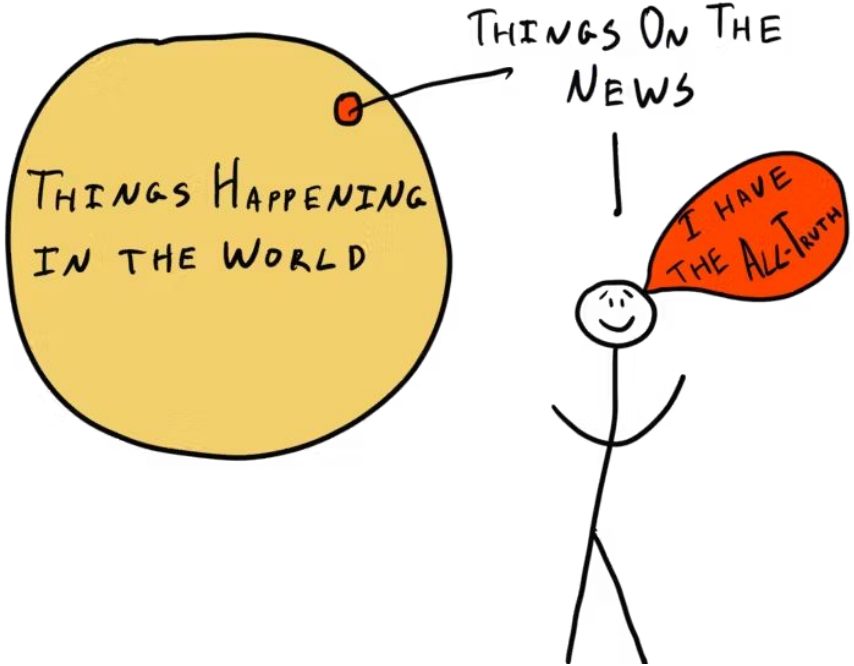


Confirmation Bias

- Focusing only on information that supports an initial estimate or assumption.

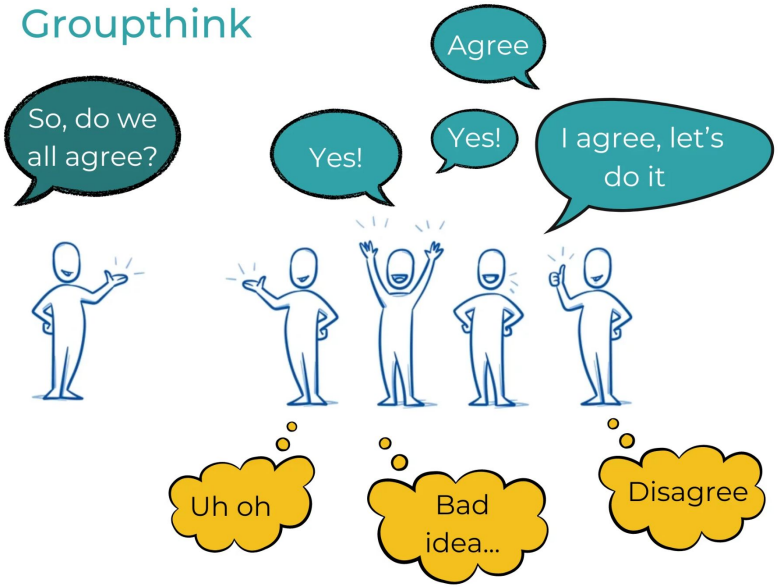
Availability Bias

- Basing estimates only on **known or memorable experiences** instead of representative data.



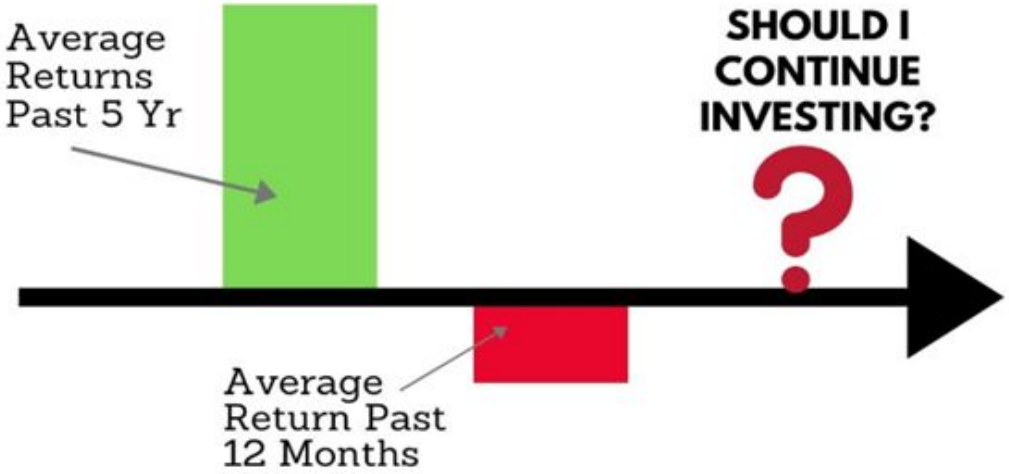
Groupthink

Agreeing too quickly with **the majority opinion**, discouraging alternative viewpoints.



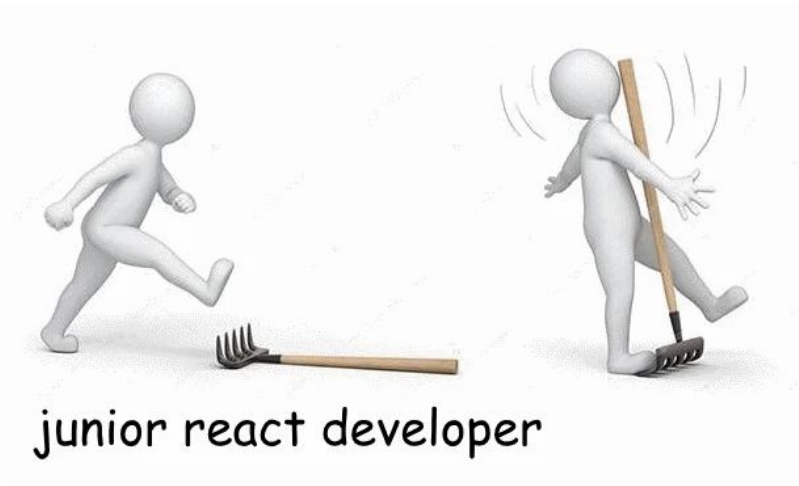
Recency Bias

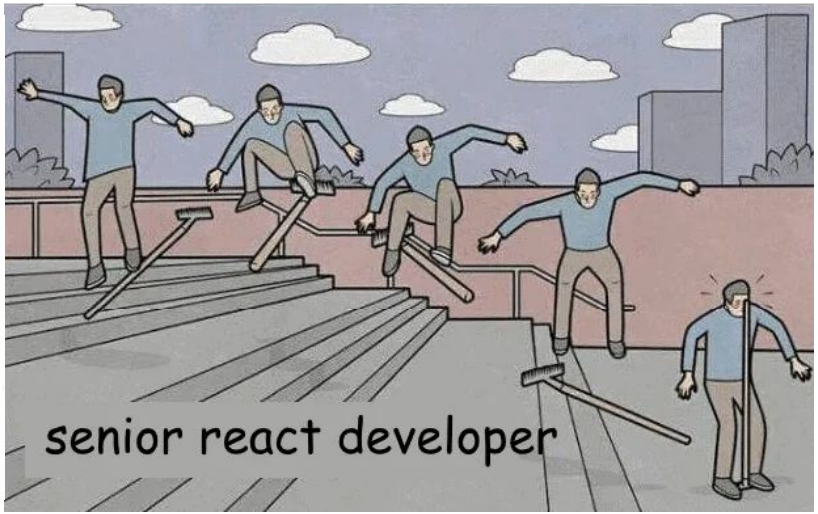
- Giving more weight to the **most recent sprint outcome** rather than the long-term trend.



Expert Bias

- Overvaluing senior team members' estimates while discounting others' input.





Best Practices

Best Practices for Accurate Estimation

- Involve the whole team.
- Use relative estimation.
- Break large stories into smaller ones.
- Leverage historical data.
- **Refine regularly**: revisit estimates during backlog grooming as new information emerges.
- Apply the **“Definition of Done”** to ensure all work (testing, documentation, reviews) is included.
- Account for uncertainty by including buffers or estimating in ranges instead of single numbers.
- **Avoid using estimates for performance evaluation.**

Retrospectives

- Retrospectives provide a **structured opportunity** for the team to reflect after each sprint.
- Encourage discussion on
  - **What went well.**
  - **What didn’t.**
  - **How estimation accuracy can improve.**
- Comparing estimated vs. actual effort helps identify recurring patterns of under-estimation or over-estimation.
- **Promote a blame-free culture focused on learning, not fault-finding.**

Retrospectives

- **Review velocity trends and capacity changes to improve future forecasts.**
- **Celebrate small improvements** to motivate consistent reflection and adaptation.
- Treat retrospectives as a **continuous feedback loop** to inspect, adapt, and evolve the team’s estimation practice.

Thank You!