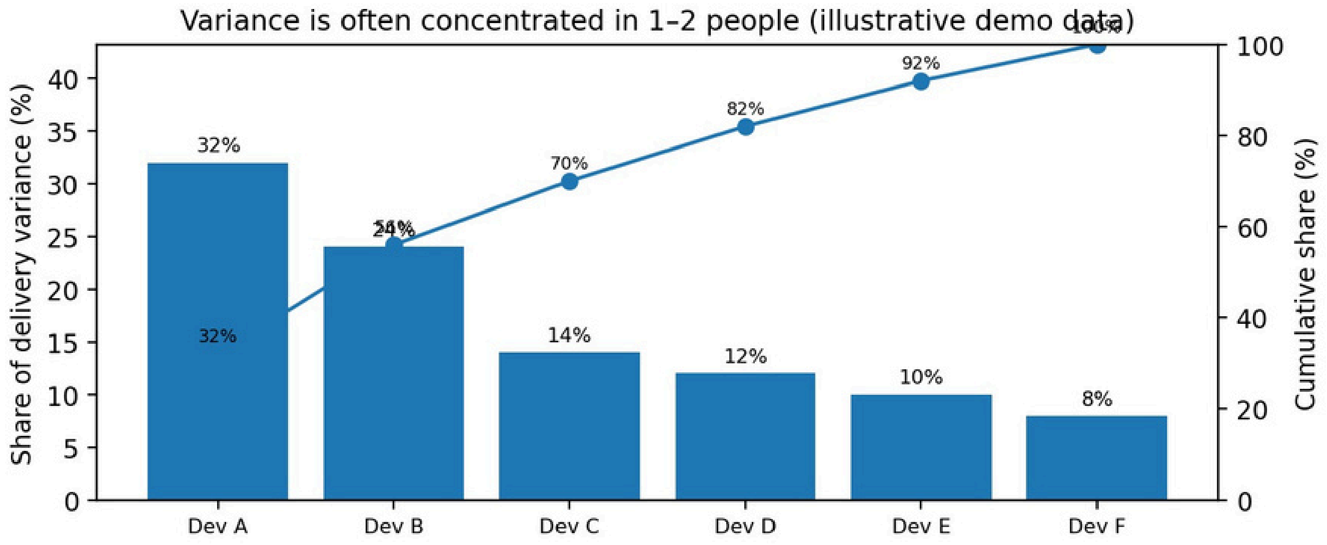
Delivery becomes fragile when variance concentrates in 1-2 people

Most delivery misses are not caused by the average pace of work. They happen when a small part of the system creates large swings - long tails, rework, and blocked work. In many teams, that variance is concentrated in a single role, dependency, or 1-2 people. When they slip, the whole schedule slips.

Rule of thumb:
If the top 2 contributors account for ~50%+ of critical-path variance, the plan is fragile - one vacation, incident, or hard bug can move the date.

The three metrics that reveal a variance sink

Metric	What it measures	Why it matters
Variance share (Top-1/Top-2)	Share of cycle-time variance held by the top contributor(s).	High concentration = fragile dates.
Tail cycle time (P85/P95)	How long work takes in the slow tail (not the average).	Tail risk drives missed commitments.
Bus factor (critical path)	How many people can unblock/finish the riskiest work.	Low bus factor turns noise into drift.



A simple Pareto view usually beats debating estimates. Find where variance lives; then reduce or diversify it.

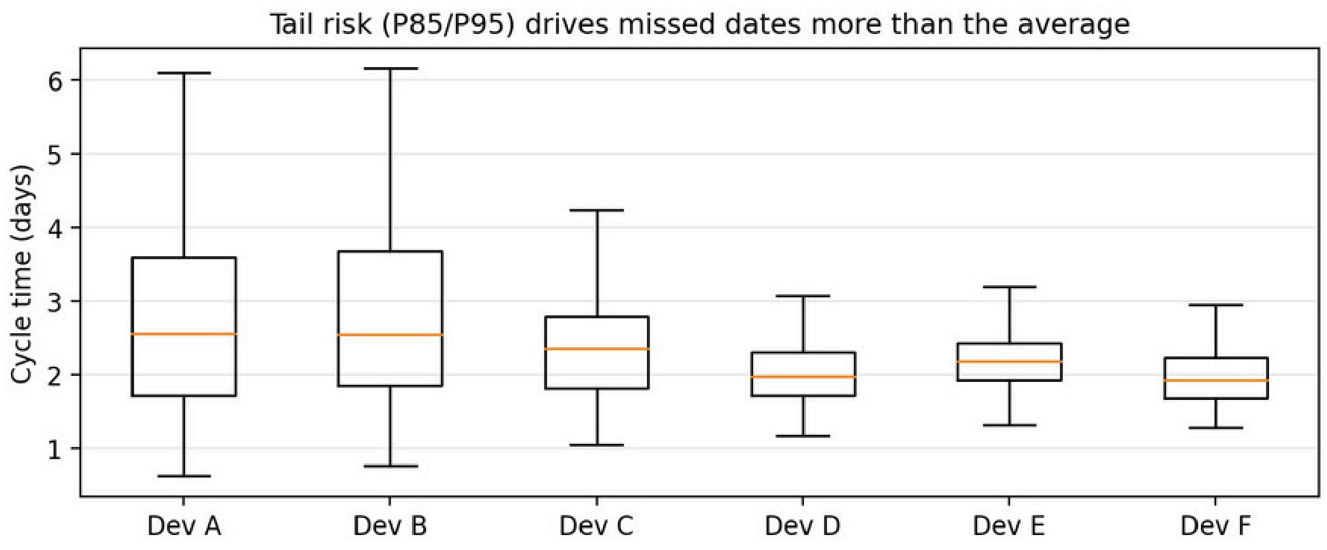
Page 2 - Why it happens (variance, not effort)

Variance sinks form when work is not evenly sliceable: deep domain knowledge, hidden dependencies, unplanned interrupts, or integration pain. The result is a long-tail distribution: most items are fine, but a few take 3-10x longer - and those few drive date misses.

1) Hero bottleneck
A small set of people own the hard parts; everything queues behind them.

2) Hidden rework
Work looks 'almost done' but bounces in review/QA/integration.

3) Unplanned interrupts
Incidents, support, and ad-hoc asks steal focus from the critical path.



Quick diagnostic (10 minutes)

- Do you see a long tail (P85/P95) that is 2-4x the median?
- Is the oldest in-progress work clustered around a person, dependency, or environment?
- Does 'almost done' work bounce between states (review, QA, integration)?
- When the date slips, can you point to one queue where it happened?

Page 3 - Fixes that reduce variance (and stabilize dates)

The goal is not to maximize average velocity. It is to reduce tail risk on the critical path.Teams that ship reliably treat variance like a defect: measure it, isolate it, and design it out.

Four levers that work in practice

- **Reduce batching:**
  Enforce small PRs and small slices; large batches create long tails.
- **Diversify the critical path:**
  Pair on the riskiest parts; rotate ownership; raise bus factor.
- **Make queues explicit:**
  Instrument review/QA/integration wait time; set SLAs for handoffs.
- **Protect focus time:**
  Create an interrupt budget; route incidents/support through a rotation.

This week's cadence (simple, enforceable):

- Monday: identify the top variance sink (person/queue/dependency) and pick one mitigation.
- Midweek: review aging work on the critical path; split, pair, or resequence.
- Thursday: run a 'tail review' (what took 3x longer? why?) and change one policy.
- Friday: track P85 cycle time + variance share; verify concentration is dropping.

Motionode’s solution

Finding variance sinks usually means pulling data from multiple systems and doing manual analysis on cycle times, aging work, and contributor concentration. Motionode surfaces variance concentration, tail risk, and the specific queues driving drift - then lets teams simulate mitigations (pairing, sequencing, WIP caps, staffing) to see which move improves schedule confidence with the least disruption.