Velocity & Forecasting Guide

Formats: Markdown (canonical) | Word (DOCX) | PDF **Updated:** 2026-03-22 **License:** CC BY 4.0 -- Kate Makrigiannis / k8mak.com

A coaching guide for understanding velocity as a capacity tool -- not a performance metric -- and using it to forecast delivery against a sized backlog. Companion to the Estimation & Sizing Guide.

When to use this

Your team has been estimating stories with relative sizing and now needs to answer "when will this be done?" or "how much can we deliver by this date?" This guide teaches velocity concepts, shows how to use velocity for forecasting, and addresses the most common misuses that erode team trust.

Core concept: Velocity is capacity, not performance

The gallon jug metaphor:

Think of your team's sprint capacity as a gallon jug. Each sprint, you can fill the jug with water (work). The jug holds the same amount every time -- that's your velocity.

Some sprints the water is clean and pours easily (well-defined stories, no blockers). Some sprints the water is muddy and pours slowly (ambiguous requirements, tech debt, team changes). The jug doesn't get bigger when the water is clean. And pushing more water into the jug doesn't make the jug bigger -- it just makes a mess.

Velocity is a measurement, not a target. You don't improve velocity by demanding higher numbers. You improve it by removing friction: clearer stories, fewer interruptions, less context switching, better tooling.

Measuring velocity

What counts toward velocity

Only count stories that are **done-done** at the end of the sprint. Done means:

- Code is merged and deployed (or deployable)
- Tests pass

- Acceptance criteria are met
- Product owner has accepted the story

Partially completed stories don't count. A story that's 80% done contributes zero points to velocity. This feels harsh, but it's essential. Counting partial work inflates your numbers and breaks your forecasts.

Calculating velocity

Track the total points completed per sprint. After 3-4 sprints, you have a usable average.

```
Sprint 1: 18 points completed
Sprint 2: 22 points completed
Sprint 3: 15 points completed
Sprint 4: 20 points completed

Average velocity: (18 + 22 + 15 + 20) / 4 = 18.75 ≈ 19 points/sprint
```

Use a range, not a single number. Your velocity isn't 19. It's 15-22. The range accounts for normal variation -- vacations, sick days, production incidents, the sprint where everything just clicks. Report velocity as a range and plan to the lower end.

When velocity stabilizes

- **Sprints 1-2:** Velocity is unreliable. The team is still calibrating their estimates. Don't forecast from this data.
- **Sprints 3-5:** Velocity starts to stabilize. You can make rough forecasts but flag them as low-confidence.
- **Sprints 6+:** Velocity is reliable enough for planning. You can forecast with reasonable confidence.

What makes velocity unstable:

- Team composition changes (someone joins or leaves)
- Major scope changes mid-sprint
- New tech stack or unfamiliar domain
- Sprint length changes

When any of these happen, reset your velocity baseline. Old data no longer applies.

Forecasting with velocity

Once you have a stable velocity, you can answer two questions:

Question 1: "When will this be done?"

You have a backlog of estimated work and need to predict a delivery date.

Steps:

1. Size the remaining backlog (total points of all stories)
2. Divide by your average velocity
3. Add buffer for unknowns

```
Remaining backlog: 95 points
Average velocity: 19 points/sprint
Sprint length: 2 weeks

Sprints needed: 95 / 19 = 5 sprints
Calendar time: 5 × 2 weeks = 10 weeks

With 20% buffer: 12 weeks
Forecast: "We expect to complete this backlog in 10–12 weeks"
```

The buffer is not optional. New stories will be added. Bugs will appear. Scope will change. A 20% buffer is the minimum for well-understood work. For less certain work, use 30-40%.

Question 2: "How much can we deliver by this date?"

You have a fixed deadline and need to predict how much of the backlog you can complete.

Steps:

1. Count the sprints between now and the deadline
2. Multiply by your velocity range (low end and high end)
3. Map the result to backlog items

```
Deadline: 8 weeks from now
Sprint length: 2 weeks
Sprints available: 4

Low estimate: 4 × 15 = 60 points
High estimate: 4 × 22 = 88 points

"We can deliver 60–88 points of work by the deadline.
That's the first 12–16 stories in the prioritized backlog."
```

Present both numbers. Stakeholders want the high number. Give them both and explain: "If everything goes perfectly, we get 88 points. If we hit normal friction, we get 60. Plan for 60, hope for 88."

Using velocity to plan a sprint

At the start of each sprint, pull stories from the top of the backlog up to your velocity.

```
Average velocity: 19 points
Buffer for unknowns: 15%

Target sprint commitment: 19 × 0.85 ≈ 16 points

Pull stories:
  – Story A: 5 points
  – Story B: 3 points
  – Story C: 5 points
  – Story D: 2 points
  – Story E: 1 point
  Total: 16 points ✓
```

Leave room. Don't fill the sprint to 100% of velocity. Production bugs, support requests, and urgent stakeholder asks will appear. A sprint planned at 85% of velocity feels sustainable. A sprint planned at 110% feels like a death march.

Stretch goals (optional): If the team finishes early, have 1-2 additional stories ready to pull in. But these are bonuses, not commitments.

What NOT to do

DO NOT use velocity as a performance metric. Velocity is a planning tool. The moment a manager says "why did velocity drop this sprint?" the team starts gaming the numbers -- inflating estimates, splitting stories to count more points, and marking stories as "done" when they're not.

DO NOT compare velocity across teams. Team A's 20-point velocity and Team B's 35-point velocity tell you nothing about which team is more productive. Different teams calibrate their scales differently. Cross-team comparison is meaningless.

DO NOT treat velocity as a commitment. "We committed to 20 points" is not how velocity works. You planned for 20 based on historical data. If you delivered 17, that's useful information for next sprint's plan -- not a failure.

DO NOT try to increase velocity. Velocity goes up when you remove obstacles, improve tooling, or reduce context switching. It does not go up when you pressure people to work faster. Artificial velocity increases are just estimate inflation in disguise.

DO NOT forecast from less than 3 sprints of data. Two data points don't give you a trend. Wait for the pattern to emerge.

Worked example

Team: Compass Squad, 2-week sprints, Fibonacci estimation

Velocity history:

Sprint	Points Planned	Points Completed	Notes
Sprint 3	20	18	One story blocked by API dependency
Sprint 4	19	22	Team in flow, no blockers
Sprint 5	20	15	Sarah (PM) out sick 3 days
Sprint 6	18	20	Normal sprint
Sprint 7	19	19	Normal sprint

Velocity range: 15-22 points/sprint **Average velocity:** 18.8 ≈ 19 points/sprint

Forecast request: "Can we ship the patient portal MVP by June 30?"

```
Remaining backlog for MVP: 72 points
Today's date: March 24
Deadline: June 30 (14 weeks away)
Sprints available: 7

Optimistic: 7 × 22 = 154 points → MVP done with room to spare
Realistic: 7 × 19 = 133 points → MVP done comfortably
Conservative: 7 × 15 = 105 points → MVP done with buffer

Forecast: "Yes, we can ship the 72-point MVP by June 30 with high
confidence. Even at our lowest historical velocity, we'd complete
105 points in 7 sprints — well above the 72 needed."
```

Facilitator tips

- **Make velocity visible.** Post the velocity chart where the team can see it. Not to judge -- to inform. A visible trend helps the team self-adjust without management pressure.
- **Celebrate consistency, not magnitude.** A team that delivers 18-22 points every sprint is healthier than a team that swings between 10 and 35. Predictability is the goal.
- **Talk about velocity in retrospectives, not standups.** Sprint-level velocity fluctuation is noise. Multi-sprint trends are signal. Discuss trends in retros where you have the space to investigate root causes.
- **When velocity drops, investigate -- don't blame.** Drops are caused by something: unclear stories, team changes, tech debt, external interruptions. Find the cause. Fix the system, not the people.

- **Teach stakeholders the range.** Every time you present a forecast, present it as a range with confidence levels. Over time, stakeholders learn that "10-12 weeks" is more honest than "11 weeks" and they'll trust your forecasts more.

How did it go?

After a few sprints of tracking velocity, check:

- [ ] Velocity is tracked as a range, not a single number
- [ ] Only done-done stories count toward velocity
- [ ] Partial work is never counted
- [ ] Sprint planning uses 85% of average velocity, not 100%
- [ ] Forecasts include a buffer (20% minimum)
- [ ] Nobody is using velocity to compare teams or evaluate performance
- [ ] Stakeholders receive forecasts as ranges with confidence levels
- [ ] The team has at least 3 sprints of data before forecasting

---

Part of the k8 Agent Toolkit. Download other formats at k8mak.com/resources.