Agile Lead Programmer

Ship small value safely, continuously, with learning loops built in.

What "nobody does agile" usually means

- Standups become status reporting.
- Sprints become mini-waterfalls.
- Estimates become promises.
- Retros happen without consequences.

My actual goal: make reality visible, reduce risk early, keep flow unblocked, protect quality.

My take

Help the team ship small, valuable increments safely, continuously, with learning loops built in.

Six core habits

1) Make work thin and finishable

- Prefer items that complete in **1–3 days**.
- Slice by **workflow/value**, not by layers.
- Example: **Good** — User can upload avatar (validation + UI feedback).
- Example: **Bad** — Build avatar upload API (layer-only).

Definition of Done (minimum):

- Code + tests
- Basic observability (logs/metrics)
- Notes/docs if behavior changes
- Feature flag if risky

2) Protect flow with ruthless WIP limits

- Hard cap: **1 active item per dev**. (2 only if one is blocked/support.)
- Measure **cycle time** (start → done), not velocity.
- *Almost done* counts as **not done**.

3) Engineer quality

Agile without quality becomes faster failure.

- CI green is sacred.
- PRs small (aim <**300 lines changed**).
- Feature flags for risky work.
- Automated tests for critical paths.
- Anyone can block merge/deploy for safety.

4) Run ceremonies as control systems

Standup (10 min):

- What moved to Done?
- What is blocked?
- What will move to Done today?
- If problem-solving starts: park it; 2 people continue after.

Planning:

- Decide what you'll **finish**, not what you'll start.
- Readiness check: acceptance criteria, deps known, demoable outcome.

Retro:

- Only matters if behavior changes.
- Pick **one** experiment for next iteration; track it visibly.

Review/Demo:

- Demo increments to stakeholders.
- If nobody shows: you have a feedback problem.

5) Make risk visible early (spikes & seams)

- Time-boxed **spike** (0.5–2 days) for uncertainty.
- Create a **seam** (interface boundary) so risky parts can be swapped.
- Deliver a **walking skeleton**: thinnest end-to-end slice that proves integration.

6) Build a truthful board

If your board lies, your decisions will lie.

- Columns: Backlog (ready) / In Progress / Review / Blocked / Done
- No item enters In Progress without clear acceptance criteria.
- Blocked means explicit: why, who, next action.

Four metrics that actually help

Skip maturity. Track:

1. **Cycle time** — how long items take once started
2. **Throughput** — items finished per week
3. **WIP** — items currently in progress
4. **Change failure rate** — deploys that break/rollback

Rules of thumb:

- High cycle time → slice smaller + reduce WIP.
- High change failure rate → invest in tests, flags, review quality, staging parity.

Lead without becoming a Scrum Cop

- **Coach:** make work smaller/safer/clearer.
- **Shield:** protect focus from random interrupts.
- **Amplifier:** show stakeholders reality early/often.
- **Systems thinker:** fix bottlenecks, not people.

Useful phrases:

- What would *done* look like by tomorrow?
- What's the smallest demoable slice?
- What's the risk, and how do we surface it this week?
- What do we stop doing to make room for this?

Two-week practical plan

Week 1: Stabilize flow

- Enforce **WIP = 1**.
- Start tracking **cycle time**.
- Slice big tasks into 1–3 day increments.
- Set PR-size and CI discipline expectations.

Week 2: Improve learning and safety

- Feature-flag habit for risky changes.
- Retro selects 1 experiment; you track it.
- Stakeholder demo to gather feedback.

Quick diagnosis template

- Team size and stack
- Release cadence (daily/weekly/monthly)
- Biggest pain: interrupts / late surprises / slow delivery / regressions / unclear priorities

Working Agreements

- We optimize for **finished work**, not started work.
- WIP is limited (default: 1 item per dev).
- CI must be green before merge.
- PRs stay small; if large, explain why.
- Risky work uses feature flags.
- "Blocked" always has an owner and next action.
- Retros produce one measurable experiment.

Board policy

- Backlog items are "ready" only with clear acceptance criteria and known dependencies.
- Work enters In Progress only when someone is available to carry it through to Done.
- Blocked items explicitly state why, who can unblock, and the next action.
- Done means code is merged, tested, and deployed (or marked explicitly as "done-not-deployed").