[A THESIS FROM GRAYHAVEN LABS]

SOFTWARE eats SOFTWARE.

The economics of software are inverting. Most code, soon, will come from agents. The value drifts away from the code itself, and toward the institutions that reorganize around the models first.

GRAYHAVEN LABS
OKLAHOMA CITY, USA

VERSION 2.0
MAY 2026

1. Everything is changing.

1.1 We're inside a shift big enough that most of what we believed about how software and business work needs throwing out.

1.2 This won't resolve quickly. Expect years.

1.3 Nobody can see where it lands yet.

1.4 Sitting it out is worse than chasing it.

2. Still underestimated.

2.1 If you think you've already priced in exponential gains, you probably haven't. [1]

2.2 Perfection isn't the bar. Better than average breaks the old software economics.

2.3 Most of the scaffolding we built to prop the models up becomes dead weight.

- 2.3.1 You don't strap training wheels on a kid who never falls.
- 2.3.2 You don't hand a perfect typist a spellchecker.

2.4 Output quality will hinge on what you ask for, not on what the model can do.

- 2.4.1 Reviewing agent code stops being useful.
- 2.4.2 Bugs from sloppy expression disappear.

2.5 Loosen the leash every two months. Stay tight and you're stuck on a low point of the curve.

[1] On the Dwarkesh Patel podcast, October 17, 2025, Andrej Karpathy said: "Overall, the models are not there… I feel like the industry is making too big of a jump and is trying to pretend like this is amazing, and it's not. It's slop." (Fortune coverage, Simon Willison's notes on the interview). On February 25, 2026, he posted on X: "It is hard to communicate how much programming has changed due to AI in the last 2 months: not gradually and over time in the 'progress as usual' way, but specifically this last December. There are a number of asterisks but imo coding agents basically didn't work before December and basically work since."

3. Most code will come from agents.

3.1 The sidebar assistant is over. Agents aren't helpers anymore.

3.2 They run whether or not anyone is watching.

3.3 Longer runs, less supervision. More than people expect.

3.4 Force an agent to work like a person and you've wasted it.

3.5 The work unit becomes the delegated task. Not lines of code.

---

5. The bottleneck moved.

5.1 Writing code that runs is the trivial part now.

5.2 What's left: errors of engineering. Priorities, sequencing, tradeoffs.

5.5 Review shifts from code to decisions.

---

6. Software development is dead.

6.1 The profession assumes code is hard to write and easy to break.

6.2 The industry assumes code is scarce.

6.3 Neither assumption holds.

---

7. The rituals don't fit.

7.1 Planning, prioritization, handoffs, reviews. All of it came from a world where shipping was slow, expensive, buggy, and tied to human hands. [2]

7.2 When code is cheap, those rituals are pure friction.

- 7.2.1 Why burn an hour on prioritization when the work itself takes thirty minutes?
- 7.2.2 Why prototype when you can talk one into existence?
- 7.2.3 Why wait on a single reviewer when five agents can review in parallel?
- 7.2.4 Why pick one idea to test when you can run all five, drop four, and bruise nobody's ego?

7.5 AI isn't just faster X. It questions whether X should exist.

[2] From Linear's Issue Tracking Is Dead essay by CEO Karri Saarinen, March 24, 2026: "It was built for a handoff model of software development. A PM scoped the work, engineers picked it up later, and the system filled with prioritization, negotiation, and workflows to bridge the gap. That ceremony came from real constraints. Engineering time was scarce."

---

8. Software changes shape.

8.1 Software was built for people to use. Going forward, most of it gets built for agents. [3]

8.2 Less software gets built ahead of time. More gets spun up when it's needed.

8.3 The line between "using software" and "building software" fades. Maybe disappears.

[3] At NVIDIA's GTC conference, March 18, 2026, in conversation with Bill Dally, Jeff Dean argued that tools designed for human speed cap the gains from faster models. As summarized on X: "An AI agent can run 50x faster, but the tools it relies on don't. So even if the model gets infinitely fast, you only get 2-3x improvement overall. Amdahl's law still applies." (Recap of the Dean–Dally GTC session.) The "trillion dollars teaching sand to think" framing wrapping Dean's point appears to be commentary from the original author of this manifesto rather than Dean's own words.

---

9. Value drifts from code.

9.1 Software that does X is worth nothing the moment an agent does X.

9.2 Code as a product someone else wrote for you loses value.

9.3 Workflow software drops in worth.

9.4 What gains value: data, permissions, distribution, trust, compliance, regulatory standing, physical assets.

9.5 Vendors whose moat was "too expensive for customers to build themselves" get crushed.

9.6 Software companies will be the last to admit any of this. Their business models depend on the old scarcity.

---

10. Winners reorganize.

10.1 Slotting models into existing systems, org charts, and processes isn't enough.

10.1.1 Agents made to behave like people are wasted.

10.2 Whoever builds around the models wins.

10.2.1 A small crew with sharp judgment and a fleet of agents beats a big team grafting AI onto pre-transformation processes.

10.4 Keeping pace matters more than headcount.

---

11. Camp at the frontier.

11.1 Big companies don't get to skip this.

11.2 Committees won't teach them the future.

11.3 They need a small autonomous team built around the models, sitting at the edge.

11.3.1 Not to bolt AI onto the old way of working.

11.3.2 To find the new way and pull the company toward it.

11.3.3 Close enough to touch real systems, real data, real limits, real consequences.

11.3.4 What it produces is software, but also people and habits.

---

12. The engineer changes.

12.1 Engineers don't vanish. They change.

12.2 The work moves from code toward product thinking, shaping systems, weighing tradeoffs, and driving business outcomes. [4]

12.4 The most valuable engineers stay the ones who think in systems and move the business. That part doesn't change.

---

[4] Salvatore Sanfilippo (antirez), from his post Redis array type: short story of a long development: "You know what was the biggest realization of all that? For high quality system programming tasks you have to still be fully involved, but I ventured to a level of complexity that I would have otherwise skipped. AI provided the safety net for two things: certain massive tasks that are very tiring (like the 32 bit support that was added and tested later), and at the same time the virtual work force required to make sure there are no obvious bugs in complicated algorithms."

13. [Therefore.]

13.1 Build for a world where intelligence is cheap, constant, and everywhere.

13.2 Optimize for learning what that world looks like.

13.3 Don’t preserve old software rituals out of habit.

13.4 Don’t wait for the picture to clear up. Play.

13.5 Get to the frontier first. That’s where the future gets learned.

[CONTACT]

Grayhaven Labs
tyler@grayhavenlabs.com
(405) 320-8212
Oklahoma City, USA

[COLOPHON]

Version 2.0 · May 2026
Set in Geist & Iowan Old Style
Typeset with Typst
grayhavenlabs.com