AI DAILY READS · EXECUTIVE BRIEF

Software 3.0 and the Rise of Agentic Engineering

Karpathy at Sequoia Ascent 2026 — the context window is the new program, and the programmer is the orchestrator.

Andrej Karpathy · Sequoia Ascent 2026 · April 2026

3.0

Software era

Dec 2025

Agentic inflection point

10x+

Agentic engineer leverage

CONTEXT

From writing code — to orchestrating agents.

Karpathy frames a step change in late 2025: LLM-driven agents stopped being autocomplete and started being a programmable layer. The unit of programming shifted from typing lines to delegating macro actions.

OLD DEFAULT WORKFLOW

- Humans write explicit code (Software 1.0)
- Humans curate datasets, train weights (Software 2.0)
- Agents useful but require frequent correction
- Programmer = primary author of every line

SOFTWARE 3.0 DEFAULT

- Humans program LLMs through context, tools, examples
- Context window becomes the main lever
- Programmer delegates: implement, refactor, research, deploy
- Programmer = orchestrator of fallible agents

ACT

01

FINDINGS & TRENDS

Three shifts that define the new era.

IN THIS SECTION

- *December 2025 marked an agentic step change*
- *Software 3.0: the context window is the new program*
- *Some apps stop existing as apps*

ACT 01 · KEY FINDINGS

The headline shifts.

Software is being refactored around agents and verifiable feedback loops.

3.0

Software paradigm

1.0 explicit code → 2.0 learned weights → 3.0 prompts, context, tools, memory. The interpreter is the LLM.

Dec 2025

Agentic inflection

Generated chunks got larger, more coherent, more reliable. The unit of programming shifted from lines to macro actions.

4×

Capability levers

Capability spike ≈ verifiability × training attention × data coverage × economic value. Frontier models spike where labs trained hardest.

10x+

Agentic engineer leverage

Mastery of agentic workflows may extend the old "10x engineer" gap by an order of magnitude — orchestration is the new ceiling skill.

ACT

02

CORE VIEWS

Karpathy's framework for what changed.

IN THIS SECTION

- *Traditional software specifies; LLMs verify*
- *Capability is jagged, not smooth*
- *Models are ghosts, not animals*

ACT 02 · THESIS

What you can specify vs. what you can verify.

The dividing line between classical software and the LLM-RL frontier.

> *Traditional software automates what you can specify. LLMs and reinforcement learning automate what you can verify.*

— Andrej Karpathy · Sequoia Ascent 2026

> *Capability spike ≈ verifiability × training attention × data coverage × economic value.*

— Karpathy · Jagged Intelligence framework

WHY THIS MATTERS · *If your task is resettable, repeatable, and rewardable — models will fly. If not, they will fail in surprisingly basic ways. The founder question becomes: are you on the model's rails, or do you need your own evals, fine-tuning, or RL environment?*

03

THEMATIC ANALYSIS

Where the new opportunity actually sits.

IN THIS SECTION

- *Some apps disappear into direct model transformations*
- *The new user is the human's agent*
- *Verifiable environments are the new wedge*

ACT 03 · THEME A

Speeding up the old app vs. dissolving it.

AI is not just faster scaffolding — for some workflows, the scaffolding disappears entirely.

FASTER

Old apps, accelerated

AI as productivity boost

- OCR pipelines, image gen, deployment glue still in place
- AI makes each step faster but the stack is preserved
- Useful, but not the deepest shift
- Most current “AI features” live here

DISSOLVED

Apps that stop existing

Direct media-to-media transforms

- MenuGen example: photo → multimodal model → rendered menu
- Frontend, OCR, image API, UI — most of the app disappears
- Old stack was scaffolding for a transform the model now does
- “Some apps should stop existing as apps”

Do not only ask “what workflow can AI speed up?” Also ask “what information transformation was impossible before but is now natural?”

ACT 03 · THEME B

Build for the human — or for the human's agent.

Most products are still designed for humans clicking screens. The user is increasingly an agent acting on behalf of a human.

HUMAN-FIRST

Click-and-read UX

Default product surface

- GUI flows, dashboards, settings panels
- Docs assume manual navigation between pages
- Onboarding is a tutorial, not an instruction set
- Auth and deploy require visual confirmation

AGENT-NATIVE

Sensors and actuators

Surfaces an agent can act on

- Markdown docs, CLIs, APIs, MCP servers
- Structured logs, machine-readable schemas
- Copy-pasteable agent instructions
- Safe permissioning + auditable actions + headless setup

A sensor turns world state into digital information; an actuator lets an agent change something. The future stack is agents using sensors and actuators on behalf of organizations.

ACT

04

INSIGHTS

What founders and engineers should internalize.

IN THIS SECTION

- *Vibe coding raises the floor; agentic engineering raises the ceiling*
- *LLMs are ghosts, not animals — jagged, alien tools*
- *Understanding cannot be outsourced, even when thinking can*

ACT 04 · INSIGHTS

Two truths about the new professional skill.

Vibe coding democratizes creation. Agentic engineering is the discipline that keeps it shippable.

INSIGHT 01

Floor vs. ceiling

Vibe coding lets almost anyone build software by description. Agentic engineering is the professional craft of coordinating fallible agents while preserving correctness, security, taste, and maintainability.

SIGNAL

The MenuGen Stripe-vs-Google email bug: plausible code, bad system design. A human still needs persistent-user-ID judgment.

Implication · *Hiring should test agentic skill directly: decompose work for agents, write specs, review diffs, harden against adversarial agents.*

INSIGHT 02

Ghosts, not animals

LLMs have no biological drives, embodied survival, or intrinsic motivation. They are statistical simulations of human artifacts — shaped by pretraining, RL, product feedback, and economic incentives.

FRAMING

Anthropomorphic expectations mislead us. These systems are brilliant in one moment and bizarrely dumb the next — jagged, alien tools.

Implication · *The right posture is empirical familiarity: learn where they work, where they fail, what they were trained for, and build guardrails accordingly.*

ACT

05

ACTIONS

Five concrete moves for founders and leaders.

IN THIS SECTION

- *Find valuable verifiable environments*
- *Make products agent-native end to end*
- *Preserve human understanding as the scarce asset*

ACT 05 · PLAYBOOK

Five moves for founders and engineering leaders.

Reorganize the work around agents, sensors, actuators, and verifiable loops.

01	**Find verifiable wedges**	Hunt domains that are valuable, verifiable, and undertrained by frontier labs. Build an environment where agents can try, get reward, and improve.	*Coding and math are crowded · economically important verticals with latent verifiable structure are the real wedge*
02	**Audit what should disappear**	For each existing app or workflow, ask whether a multimodal model could perform the core transformation directly — dissolving the scaffolding.	*Some apps should stop existing as apps · don't build faster horses*
03	**Ship agent-native surfaces**	Treat the human's agent as the primary user. Expose CLIs, APIs, MCP servers, markdown docs, structured logs, machine-readable schemas, safe permissions.	*Auditable actions and headless setup beat click-through tutorials · adoption follows the agent*
04	**Hire for agentic engineering**	Replace traditional coding puzzles with: build a substantial project with agents, deploy it, secure it, then have adversarial agents try to break it.	*Tests real ceiling skills · decomposition, specs, review, security, leverage — not memorization*
05	**Protect human understanding**	Invest in distilled, inspectable artifacts (microGPT-style) and LLM-maintained knowledge bases that turn information into understanding — not just answers.	*Thinking can be outsourced; understanding cannot · keep judgment in the loop*

SYNTHESIS

AI is becoming a new operating layer for digital work.

The scarce thing has shifted — and the work itself is being reorganized.

01

What changed

Software 3.0 made the context window the main lever. December 2025 was a step change: agents stopped being autocomplete and started being a programmable layer for digital work.

02

What it means

Less scarce: code generation, API recall, first drafts, repetitive setup. More scarce: understanding, taste, eval design, security, system boundaries, and knowing when the model is off the rails.

03

What to watch

Jagged capability, agent-native infrastructure, verifiable environments, and the gap between humans who orchestrate agents skillfully and those who do not.

Define the context, define the tools, define the feedback loop, define the guardrails — let agents work, and preserve human understanding.

READ THE FULL TRANSCRIPT

Thank you.

Curated executive briefs · aidigest.club / reads

READ THE FULL REPORT

Karpathy · Sequoia Ascent 2026

https://karpathy.bearblog.dev/sequoia-ascent-2026/

Karpathy, A. (2026). Sequoia Ascent 2026: Software 3.0, Agentic Engineering, and Jagged Intelligence. Fireside chat with Stephanie Zhan.