

Research Context Engineering: Agentic Development in RSE

Annual Swiss RSE Day 2026

Markus Kreft

Zurich, 31 August 2026



rse.swiss/qa_1

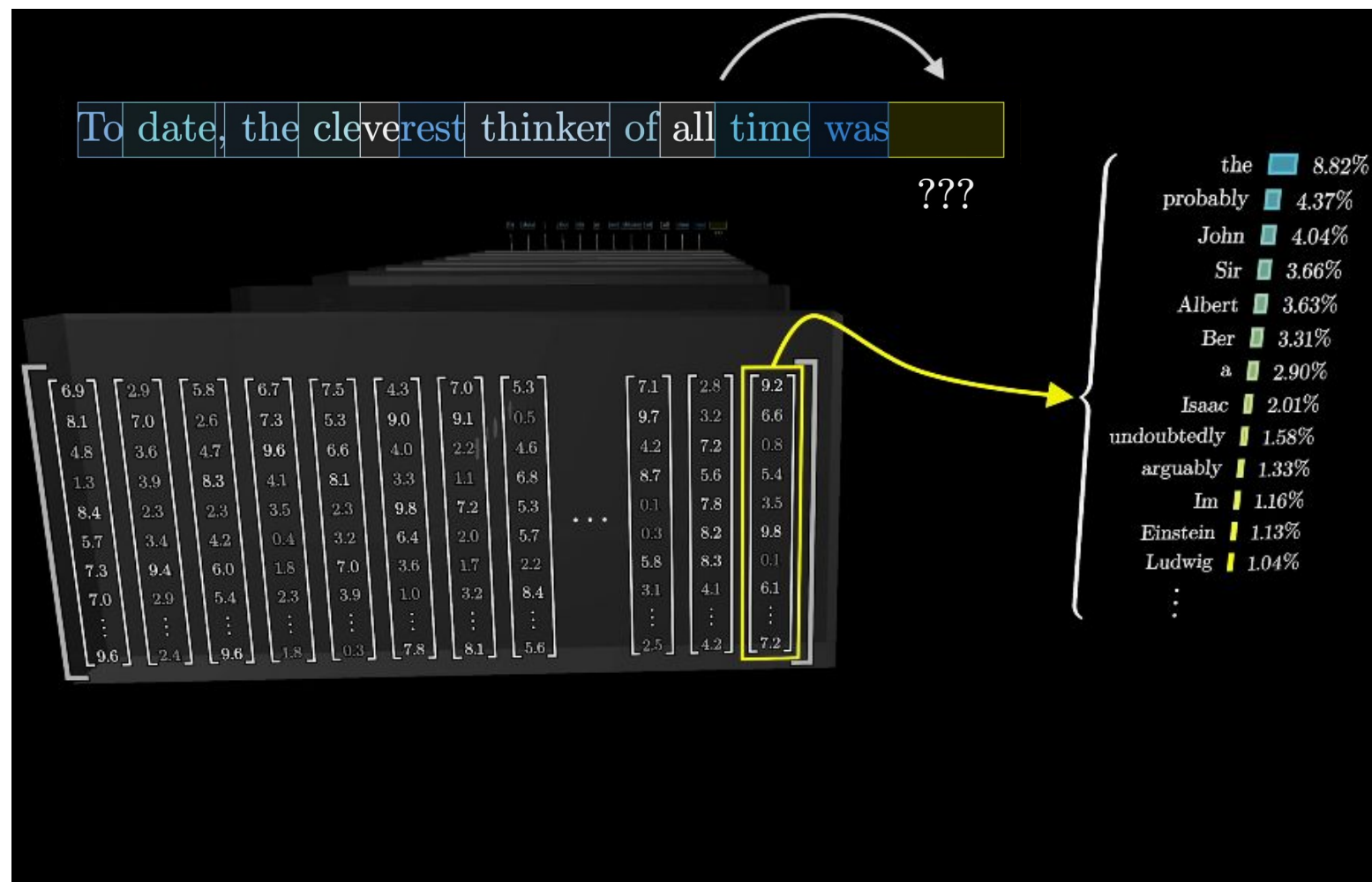


Image: 3Blue1Brown 2024, <https://www.youtube.com/watch?v=wjZofjX0v4M>

Agents

LLM in a loop + Structured Output + Tools

```
while True:
```

```
    y = LLM(x)
```

```
    x = action(y)
```

Terminology

2025 Vibe Coding¹

2026 Agentic Engineering²

≥ Agentic Development

≥ Agentic Programming

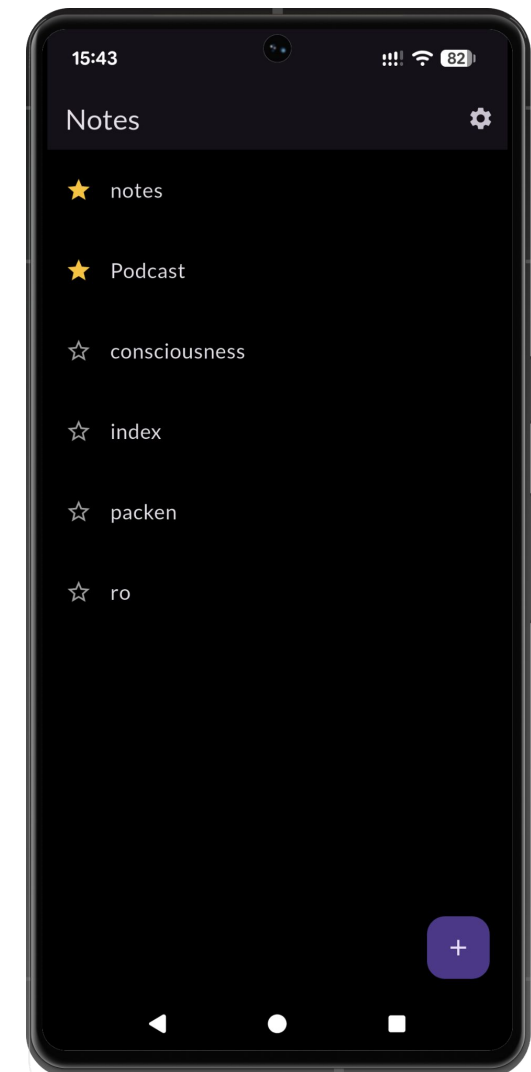
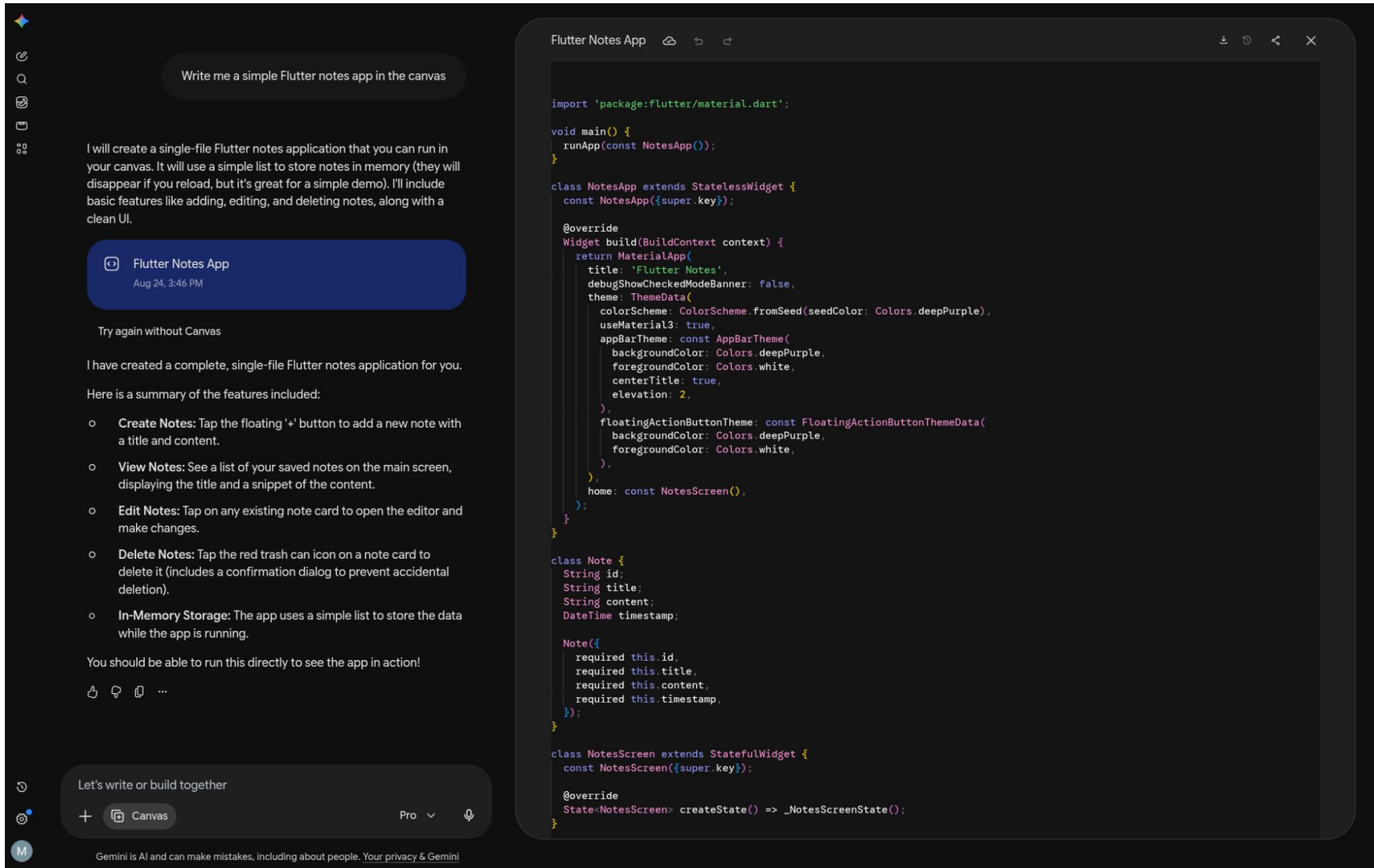
> Agentic Coding



Image: Peter Steinberger, creator of OpenClaw

¹Andrej Karpathy, <https://x.com/karpathy/status/1886192184808149383?lang=en>

²Andrej Karpathy, From Vibe Coding to Agentic Engineering, <https://www.youtube.com/watch?v=96jN2OCOfLs>

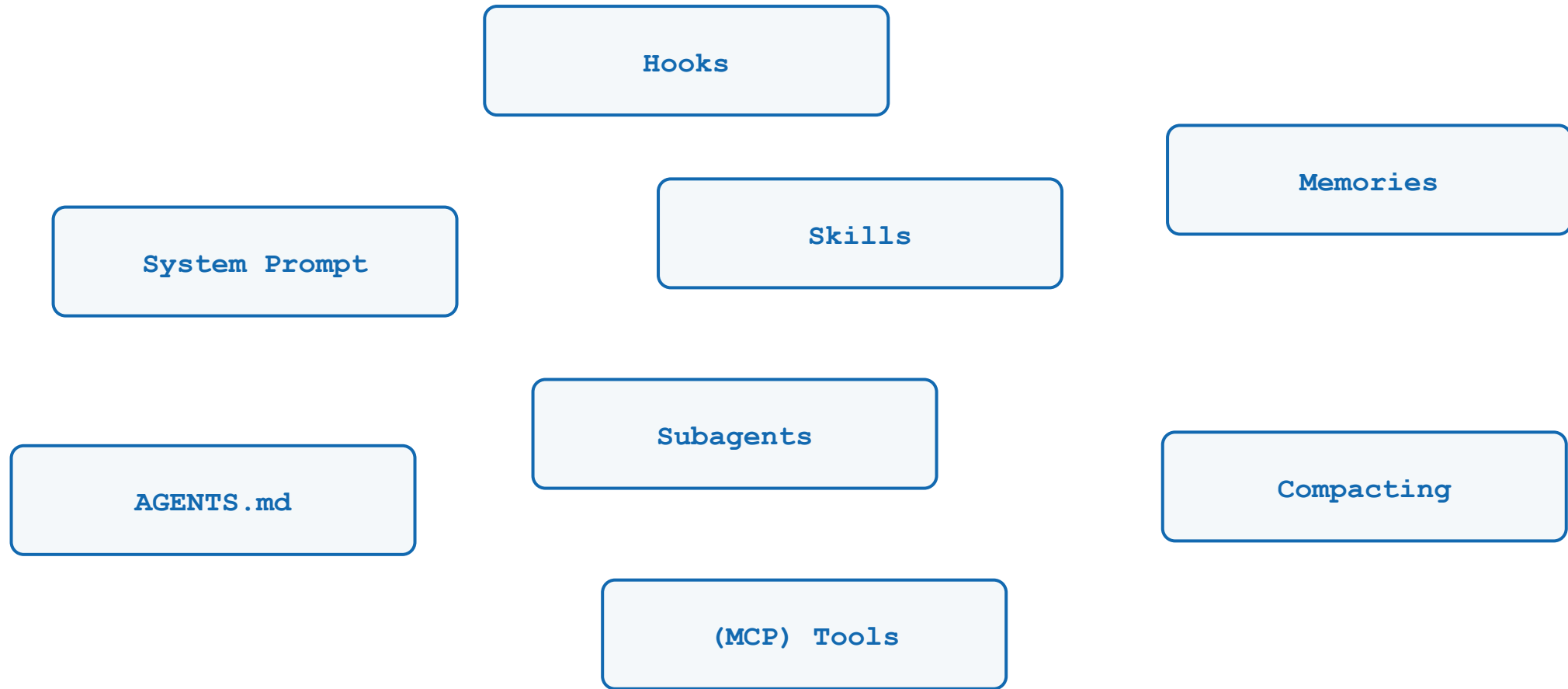


Context is Key

- > Context Rot: As context grows, reliability degrades
- > This is the problem with vibe coding
- > Most work on agents is about context engineering
- > Senior Software Engineers manage the context of Junior Developers

[illegible]

Agent Harness



What is AI?

- > Assembly
- > C
- > Managed languages
- > Frameworks
- > English?

Tom Enden 2026, <https://www.youtube.com/watch?v=VIYnXUQkKsU>

AI is Different

- > Everyone can produce code now
 - o This is new: Programmers lost something that made them unique
- > But not everyone can produce **good code**
 - o AI alone raises the floor, but not the ceiling
- > Bad *"code is cheap"*, good code is not¹

Agentic without
Engineering
=
Vibe Coding

Writing Code
≠
Software
Engineering

¹Matt Pocock 2026, <https://www.youtube.com/watch?v=v4F1gFy-hqg>

The Role of RSE

- > RSE is as important as ever
 - > AI increases the pressure to be a good RSE
 - Design, architecture and robustness
 - This was the role of RSE all along
 - > AI empowers RSEs
 - Learning tool
 - Productivity tool
- => Time for learning and experimentation
- How to use AI?
 - When to trust and when to question?
 - The next generation of devs is AI native

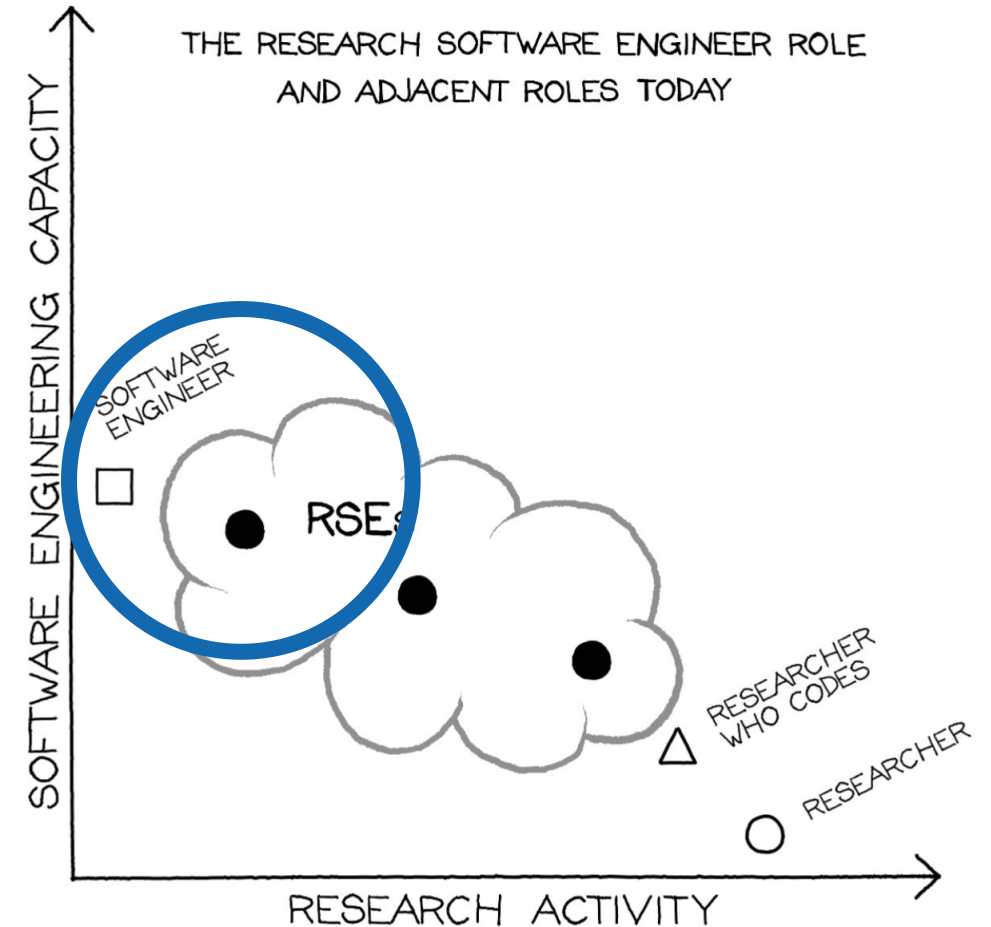


Image: Druskat et al. 2026 Research Software Engineers in the Age of GenAI: Same Value, Changing Practice <https://zenodo.org/records/20320179>

How to Use AI

- > Feedback loops: linting, tests, telemetry
 - LLMs are not (meant to be) provably correct like other tools
 - They need verification and dev tools (just like humans do)
- > Shared understanding with AI (e.g., /grill-me¹)
 - Empathy for the Agent² (but don't anthropomorphize)
- > Architecture: Design the interface, delegate the implementation³

¹<https://github.com/mattpocock/skills/blob/main/skills/productivity/grilling/SKILL.md>

²Peter Steinberger 2026, <https://youtu.be/YFjBk8HI5o?t=4660>

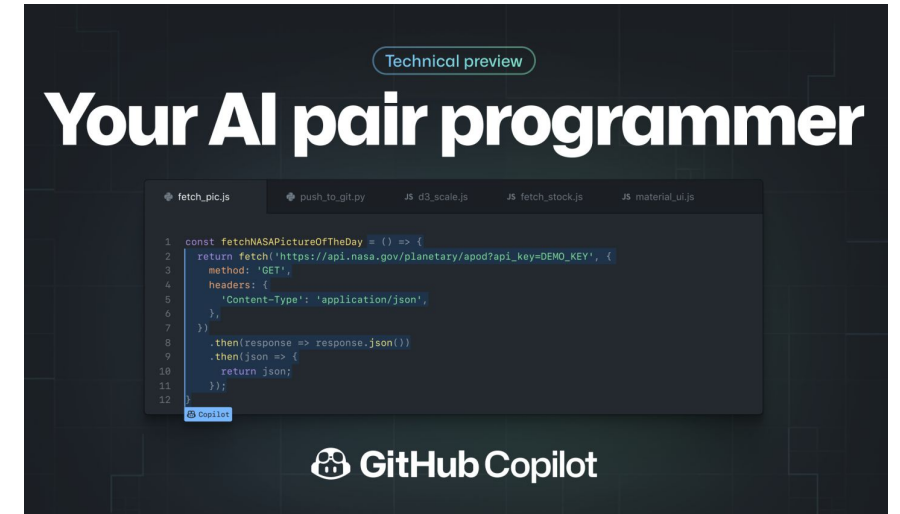
³Matt Pocock 2026, <https://www.youtube.com/watch?v=v4F1gFy-hqg>

Interactive: No Code Written

- > Comprehension: Inherited codebase, author left two years ago
- > Sparring partner: Many RSEs work in small teams
- > Ideation: What's the standard way to do this?
- > Learning: AI is decently good in a huge variety of topics

Copilot: Delegating Keystrokes

- > Non-Agentic: Syntax and API specs
 - Wrong signature: fails immediately
 - Wrong architecture: compiles, passes, but may be bad or insecure
- > Agentic: Tooling and build systems



GitHub Copilot 2022

Coding agents are a compression of a lot of mundane programming knowledge¹

¹Andrej Karpathy 2026, <https://youtu.be/96jN2OCOfLs>

Junior Dev: Delegating Changes

- > Routine maintenance: dependency bumps, deprecations
 - o The FOSS/RSE work nobody has time for
- > Increase test coverage
- > Small changes in a well-structured project
 - o Often the hard part isn't the few lines, but knowing where they go
- > Prerequisite: Good structure and test suite

One-shot: Delegating Disposable Artifacts

- > Complex but self-contained projects: Throwaway analysis, PoCs, Demos
- > Review behaviour, not code
- > Re-implement things that work afterwards by hand
 - o Danger: Demos escape and become production code

--dangerously-skip-permissions

- > No one reads the code (except AI)
- > Works when: The artifact is small, self-contained, and its output is obviously right or wrong
- > Breaks when: The code is the method
 - Unreviewed code is unreviewed methodology
 - No compiler, no test, no reviewer can tell
- > Only consider when building tooling around science, not the science

When to Avoid AI: Costs too High

- > When the prompt gets longer than the code
 - Programming languages are precise
 - Unless spec is your durable artifact
- > Cognitive debt^{1,2}, manage it like technical debt: take on, but repay
 - delegate knowledge, not understanding
- > Review capacity is the bottleneck
 - generation got cheap, review didn't
 - FOSS maintainers flooded with drive-by AI PRs (e.g., curl bug bounty)

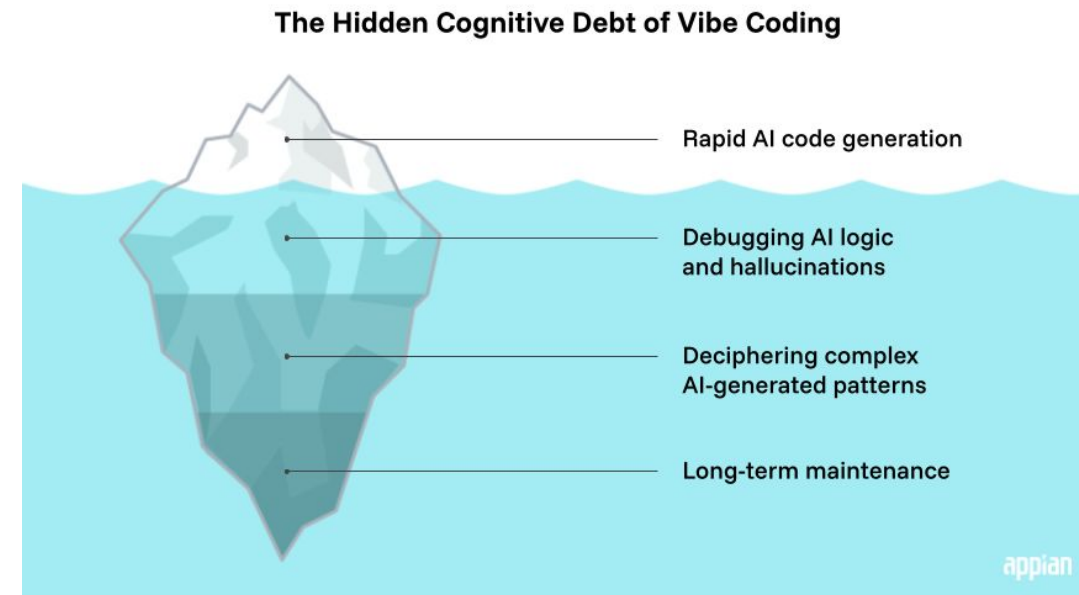


Image: <https://appian.com/blog/2026/vibe-coding-and-cognitive-debt>

¹Margaret-Anne Storey 2026, <https://margaretstorey.com/blog/2026/02/09/cognitive-debt/>

²Nataliya Kosmyna *et al.* 2025, <https://arxiv.org/abs/2506.08872>

When to Avoid AI: Invisible Failure

- > Plausible-but-wrong semantics
 - Context outside the code (units, storage conventions)
 - Caught by analytic solutions, convergence studies, magnitude checks
- > Sensitive, embargoed, or restricted data
 - Unpublished results, patient data, ethics approval, DUA
 - Irreversible!

Ownership

- > You take responsibility for the code you deliver
 - o Let Agents co-author commits
- > You have always taken some responsibility for the libraries you choose

"Would you be comfortable owning a production incident tied to this PR?"¹

¹Matthew Binshtok 2026, Agent responsibly, Vercel

What makes RSE unique?

Technical

- > Reproducibility is essential
- > Testing a method is hard
 - No expected value to compare
 - Only properties that must hold
 - Tolerance is a judgment call
- > No domain expert / analyst
 - RSE has to argue the science and fill gaps in implementation

Institutional

- > Rigor is self-imposed, not regulated
 - Every practice to increase trust has to be argued for
 - > Short grant cycles, no maintenance budget
 - > Wide duty spectrum within one role
- => This is exactly what RSE is solving

feat(note-list): add option to disable swipe actions #3088

Merged AndyScherzinger merged 2 commits into nextcloud:main from markus-kreft:feat/disable-swipe-actions on Mar 17

Conversation 6 Commits 2 Checks 14 Files changed 6 +52 -1

Commit 59b8c43

Submit comments

Filter files...

- app/src/main
- java/it/niedermann/owncloud/...
- main/items/list
 - NotesListViewItemTouchHel...
- preferences
 - PreferencesFragment.java
 - NotesApplication.java
- res
 - drawable
 - ic_swipe_grey600_24dp.xml
 - values
 - strings.xml
 - xml
 - preferences.xml

feat(note-list): add option to disable swipe actions

Add a toggle in the "Appearance and behavior" settings to enable or disable swipe gestures (delete and favorite) in the note list.

The default value is set to enabled, preserving existing behavior while allowing users who accidentally trigger these actions to disable them.

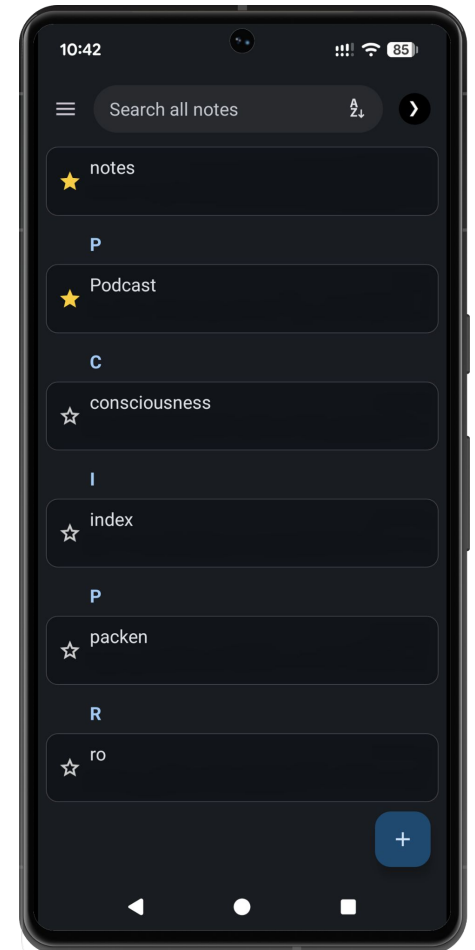
Changes:

- Add 'swipe_actions' preference to 'preferences.xml'
- Add 'isSwipeEnabled' global state in 'NotesApplication'
- Update 'NotesListViewItemTouchHelper' to respect the setting
- Create 'ic_swipe_grey600_24dp.xml' for the settings icon

< Prev Next > Browse files

markus-kreft authored and AndyScherzinger committed on Mar 17 commit 59b8c43

```
...loud/notes/main/items/list/NotesListViewItemTouchHelper.java
@@ -24,6 +24,7 @@
24 24 import com.google.android.material.snackbar.Snackbar;
25 25
26 26 import it.niedermann.owncloud.notes.R;
27 + import it.niedermann.owncloud.notes.NotesApplication;
```



Closing Thoughts

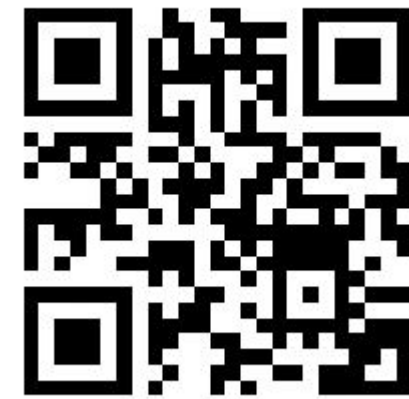
- > `git (-worktree)` is a must
- > Use a sandbox
- > Write your docs for agents, ship skills for your tools
- > Delegate knowledge, not understanding

Research Context Engineering: Agentic Development in RSE

Annual Swiss RSE Day 2026

Markus Kreft

Zurich, 31 August 2026



rse.swiss/qa_1