

Working with AI: Lessons from software's head start

Evolutionary Parallels: Software Engineering vs. Excel AI

Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.

**SOFTWARE
ENGINEERING**



**EXCEL / FINANCIAL
KNOWLEDGE WORK**

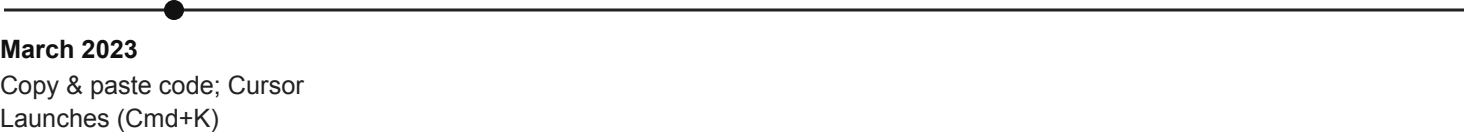


Evolutionary Parallels: Software Engineering vs. Excel AI

Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.

STAGE 1
AI assists

SOFTWARE
ENGINEERING

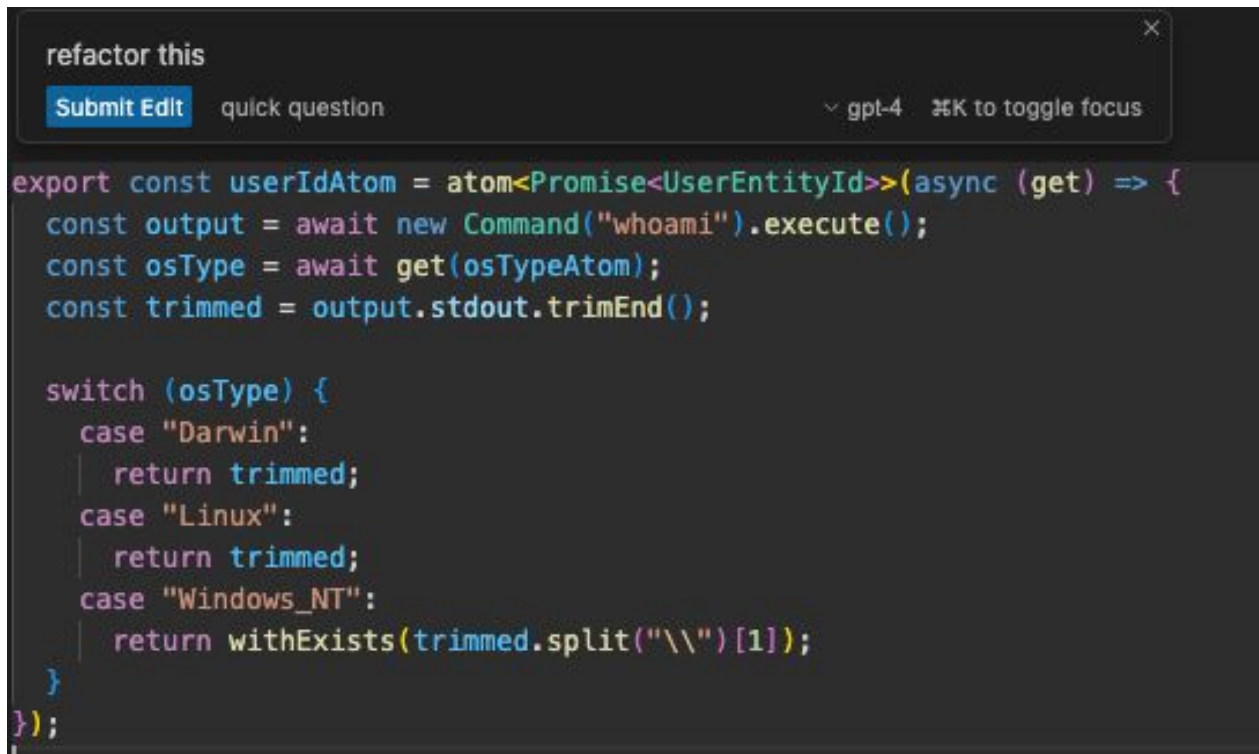


EXCEL / FINANCIAL
KNOWLEDGE WORK



Spring 2023: AI enters the programming workflow

Assists with the existing way of working

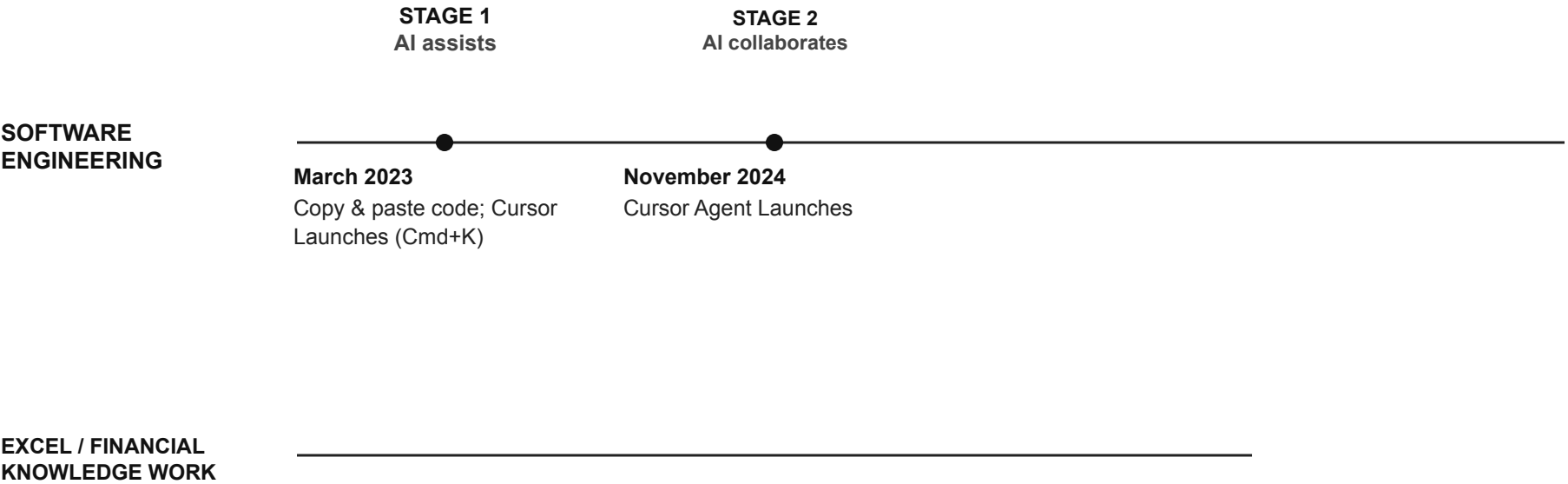
A screenshot of the Cursor IDE interface. At the top, a dark header bar contains the text "refactor this" on the left, a "Submit Edit" button in the center, and a "quick question" input field on the right. Below the header, a code editor displays a JavaScript code snippet. The code defines a function "get" that uses "atom" and "Command" to execute a "whoami" command and return the OS type. A switch statement handles different OS types: "Darwin", "Linux", and "Windows_NT". The code is syntax-highlighted with various colors. The editor is set to a dark theme.

```
export const userIdAtom = atom<Promise<UserEntityId>>(async (get) => {
  const output = await new Command("whoami").execute();
  const osType = await get(osTypeAtom);
  const trimmed = output.stdout.trimEnd();

  switch (osType) {
    case "Darwin":
      return trimmed;
    case "Linux":
      return trimmed;
    case "Windows_NT":
      return withExists(trimmed.split("\\\\")[1]);
  }
});
```

Evolutionary Parallels: Software Engineering vs. Excel AI

Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.



Fall 2024: AI becomes a first-class participant in the workflow

Collaborates in the existing way of working

The screenshot displays the Cursor IDE interface with a code diff for `controllers/setController.js`. The diff shows changes to the `listSets` method, adding query parameters for `includeExercises`, `includeWorkout`, and `includeMuscleGroup`. The right sidebar shows the Cursor Composer chat interface, which provides a summary of the changes and offers options to accept or reject the updates.

Diff View:

```
7 exports.listSets = async (req, res) => {
8   try {
9     const { workout_id } = req.query;
10    const whereClause = workout_id ? { workout_id } : {};
11
12    const sets = await Set.findAll({
13      where: whereClause,
14      include: [
15        {
16          model: Exercise,
17          include: [MuscleGroup],
18        },
19        Workout,
20      ],
21      order: [["order", "ASC"]],
22    });
23  }
24}
```

Composer:

I want to standardize the objects returned by the API. Update all the API methods in the controllers for the list methods so that I can specify which related entities to return as part of the response. Add sane defaults so that the methods return expected entities that are used together.

I want to standardize the objects returned by the API. Update all the API methods in the controllers for the list methods so that I can specify which related entities to return as part of the response. Add sane defaults so that the methods return expected entities that are used together.

To standardize the API responses and allow for specifying related entities, I'll update the list methods in each controller. The changes will include adding query parameters to specify which related entities to include, along with sane defaults for each method.

Here are the updates:

- `setController.js` 1/1
- `exerciseController.js` 1/1
- `muscleGroupController.js` 1/1
- `workoutSetController.js` 1/1
- `workoutController.js` 1/1

Summary of Changes:

- Added query parameters to each list method to specify which related entities to include in the response.
- Default behavior is to include commonly used related entities, which can be overridden by setting the corresponding query parameter to `false`.

Completed Save all Reply all Reject all **Accept all**

+ `setController.js` × `exerciseController.js` × `muscleGroupController.js` ×

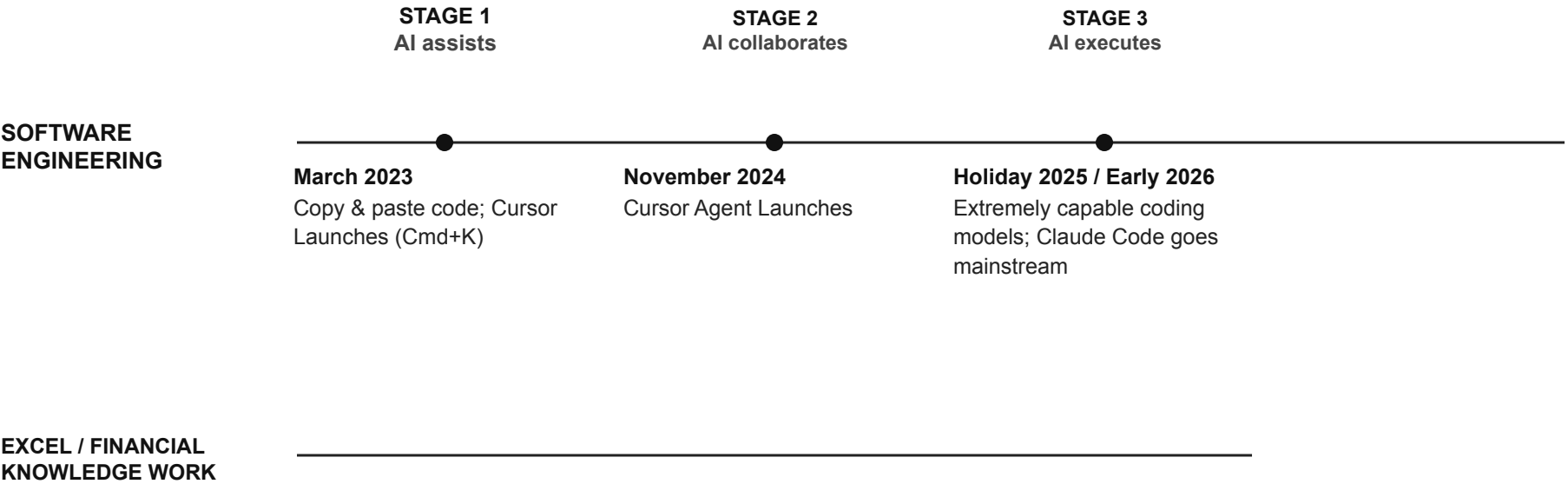
Type your instructions, / for commands

Esc to close

Cursor Composer,
September 2024
(Agent launched
November 2024)

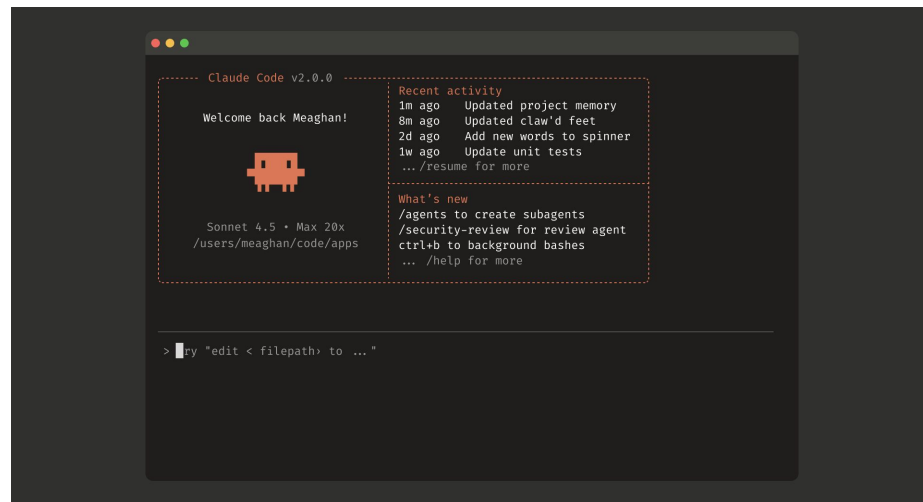
Evolutionary Parallels: Software Engineering vs. Excel AI

Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.



Winter 2025: Coding agents become extremely capable

Way of working changes; AI executes independently & users interface with the AI



Andrej Karpathy 
@karpathy

Jan 26, 2026  

A few random notes from claude coding quite a bit last few weeks.

Coding workflow. Given the latest lift in LLM coding capability, like many others I rapidly went from about 80% manual+autocomplete coding and 20% agents in November to 80% agent coding and 20% edits+touchups in December. i.e. I really am mostly programming in English now, a bit sheepishly telling the LLM what code to write... in words. It hurts the ego a bit but the power to operate over software in large "code actions" is just too net useful, especially once you adapt to it, configure it, learn to use it, and wrap your head around what it can and cannot do. This is easily the biggest change to my basic coding workflow in ~2 decades of programming and it happened over the course of a few weeks. I'd expect something similar to be happening to well into double digit percent of engineers out there, while the awareness of it in the general population feels well into low single digit percent.

Anthropic acquires Bun as Claude Code reaches \$1B milestone

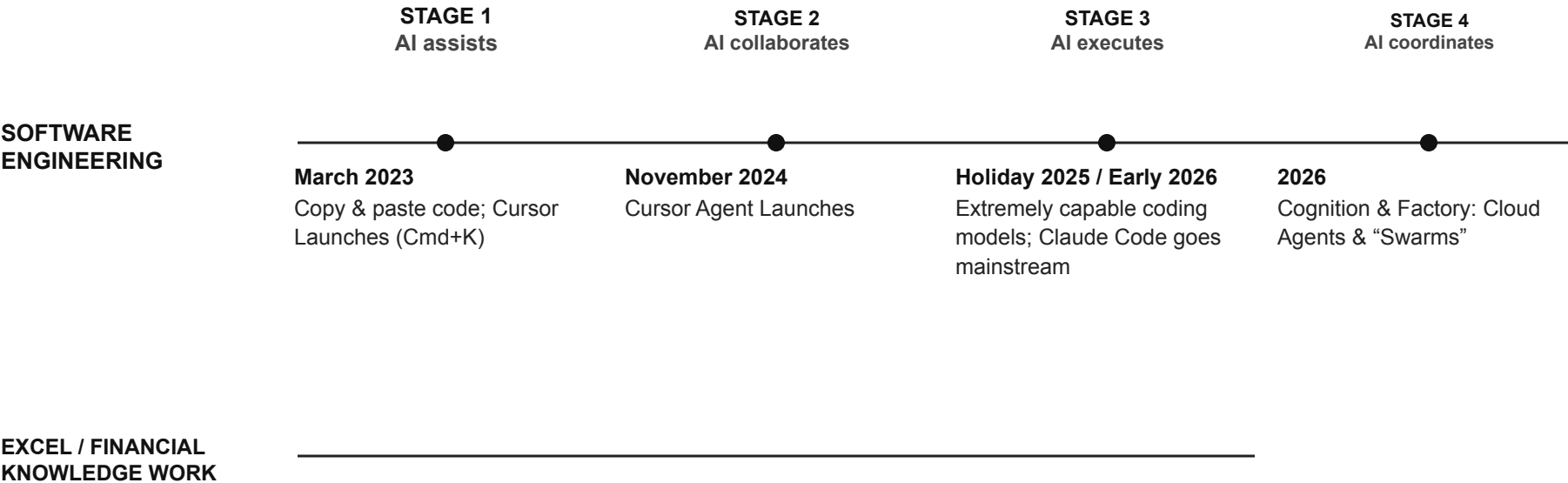
Claude is the world's smartest and most capable AI model for developers, startups, and enterprises. Claude Code represents a new era of agentic coding, fundamentally changing how teams build

Published

Dec 3, 2025

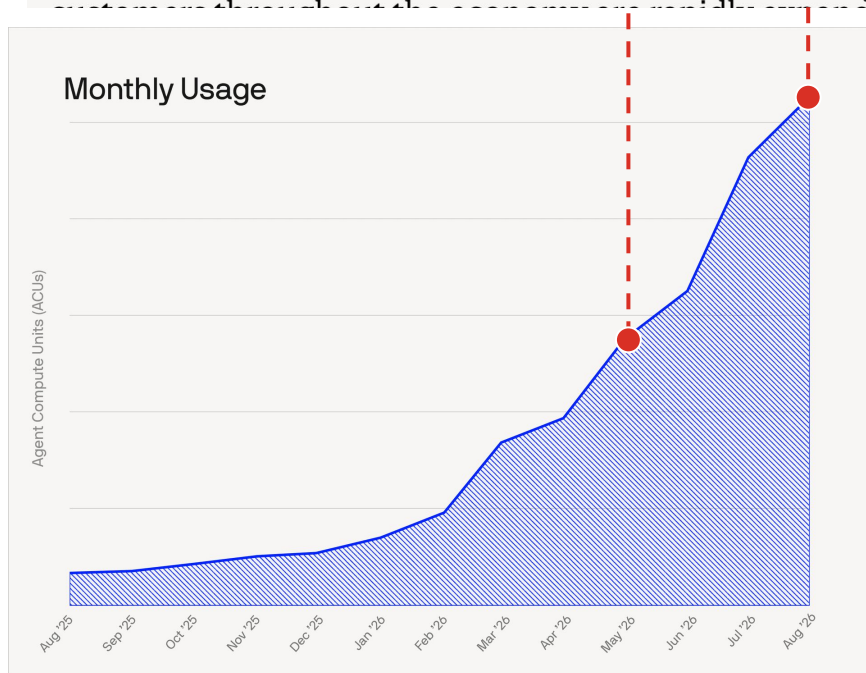
Evolutionary Parallels: Software Engineering vs. Excel AI

Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.



2026: From individual agents to autonomous cloud workforces

Since our last round in May, Cognition's run-rate revenue has grown from \$492M to almost \$900M. As Devin becomes an essential part of how companies build software,

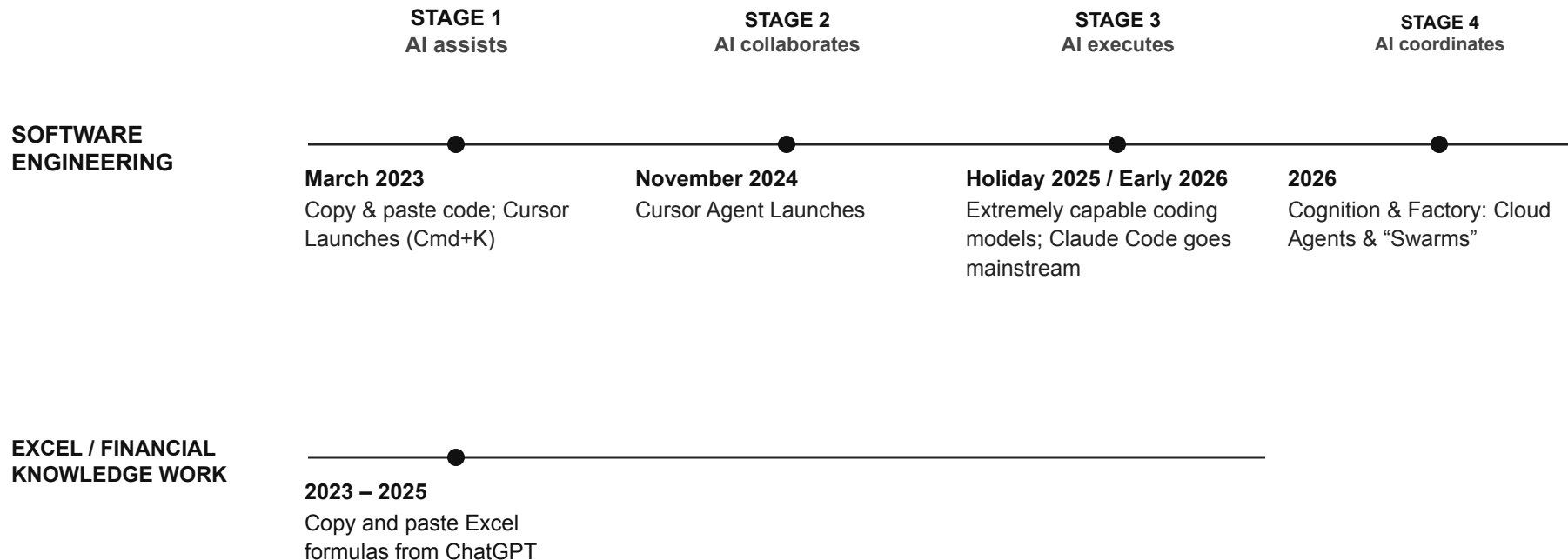


**Cognition AI raises \$2 billion
at \$48 billion valuation**

**Factory raises \$200M at
\$5B valuation**

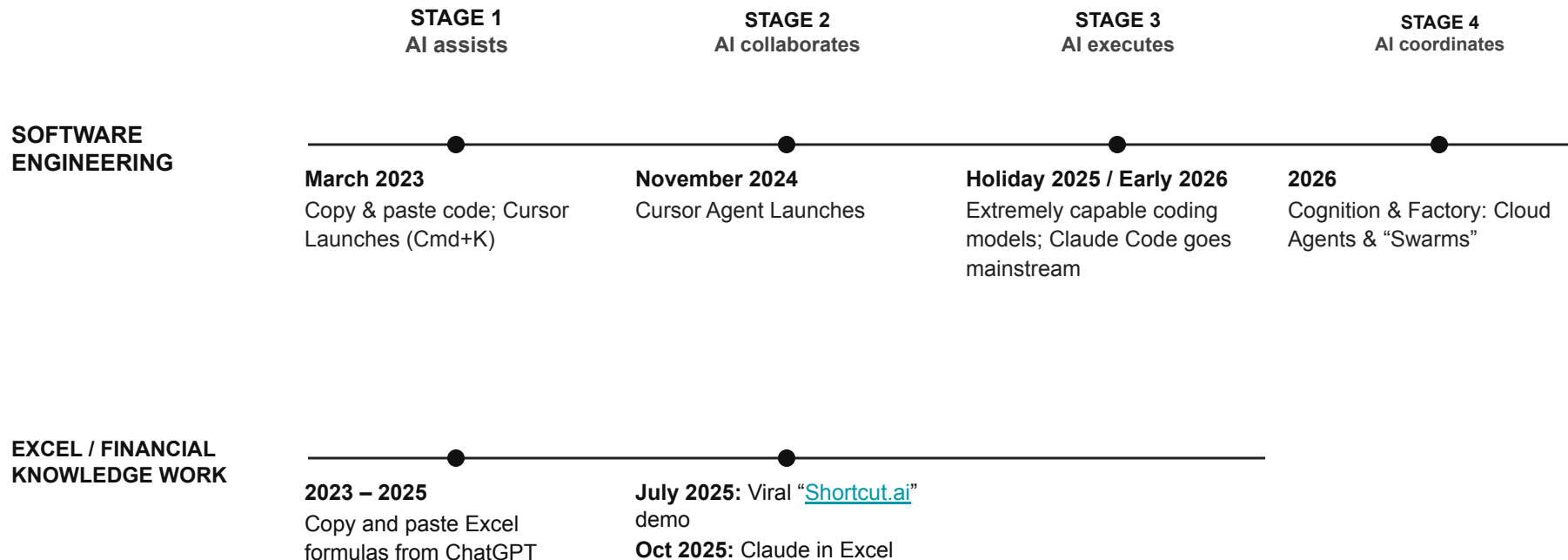
Evolutionary Parallels: Software Engineering vs. Excel AI

Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.

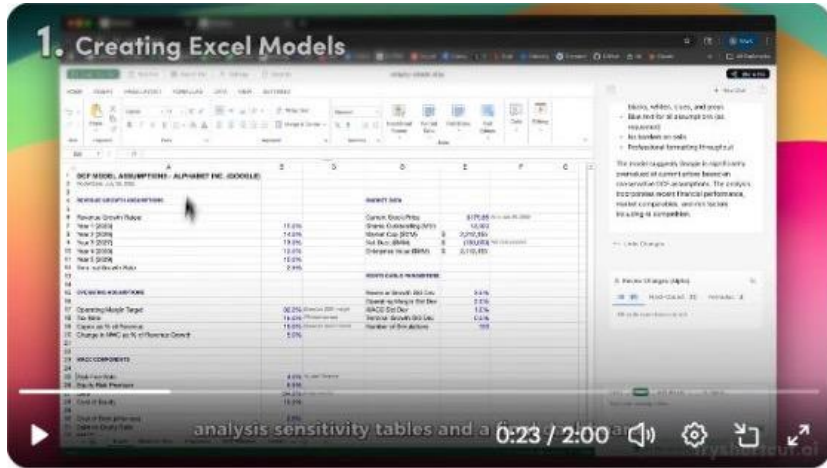


Evolutionary Parallels: Software Engineering vs. Excel AI

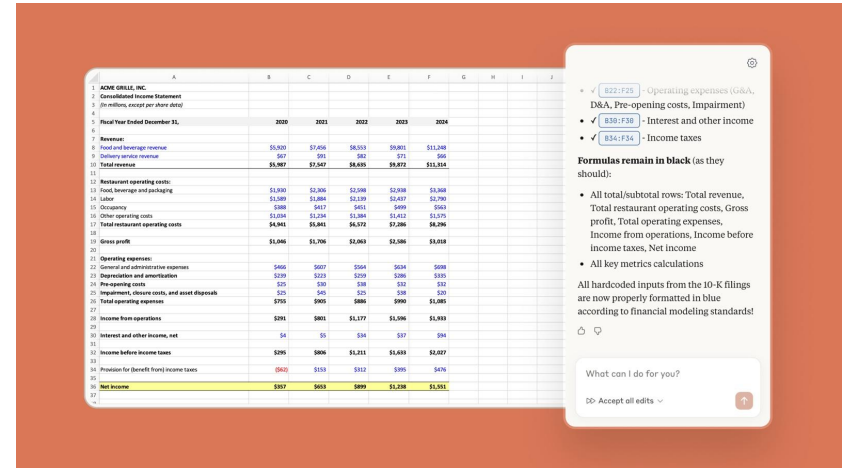
Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.



Excel has its Cursor Agent moment



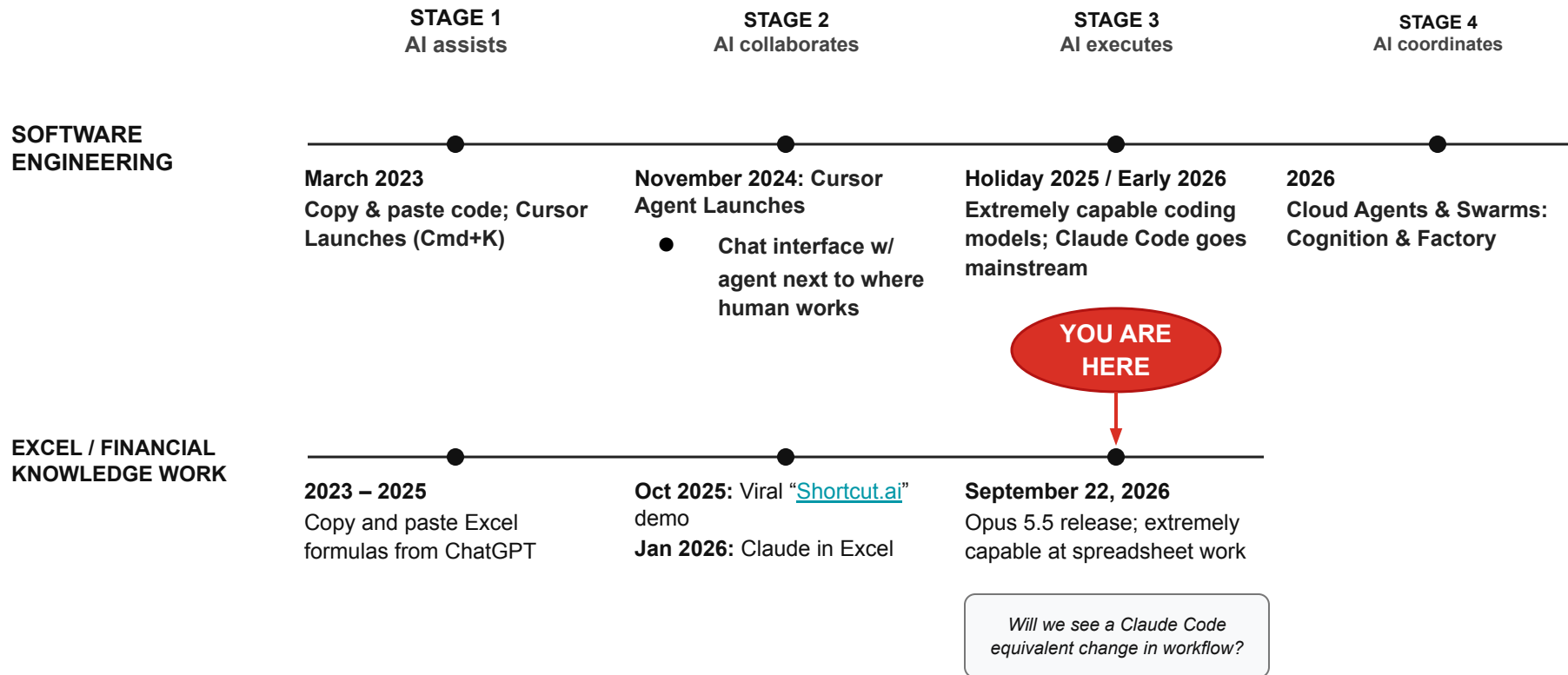
Shortcut (July 2025)



Claude in Excel (Oct 2025)

Evolutionary Parallels: Software Engineering vs. Excel AI

Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.



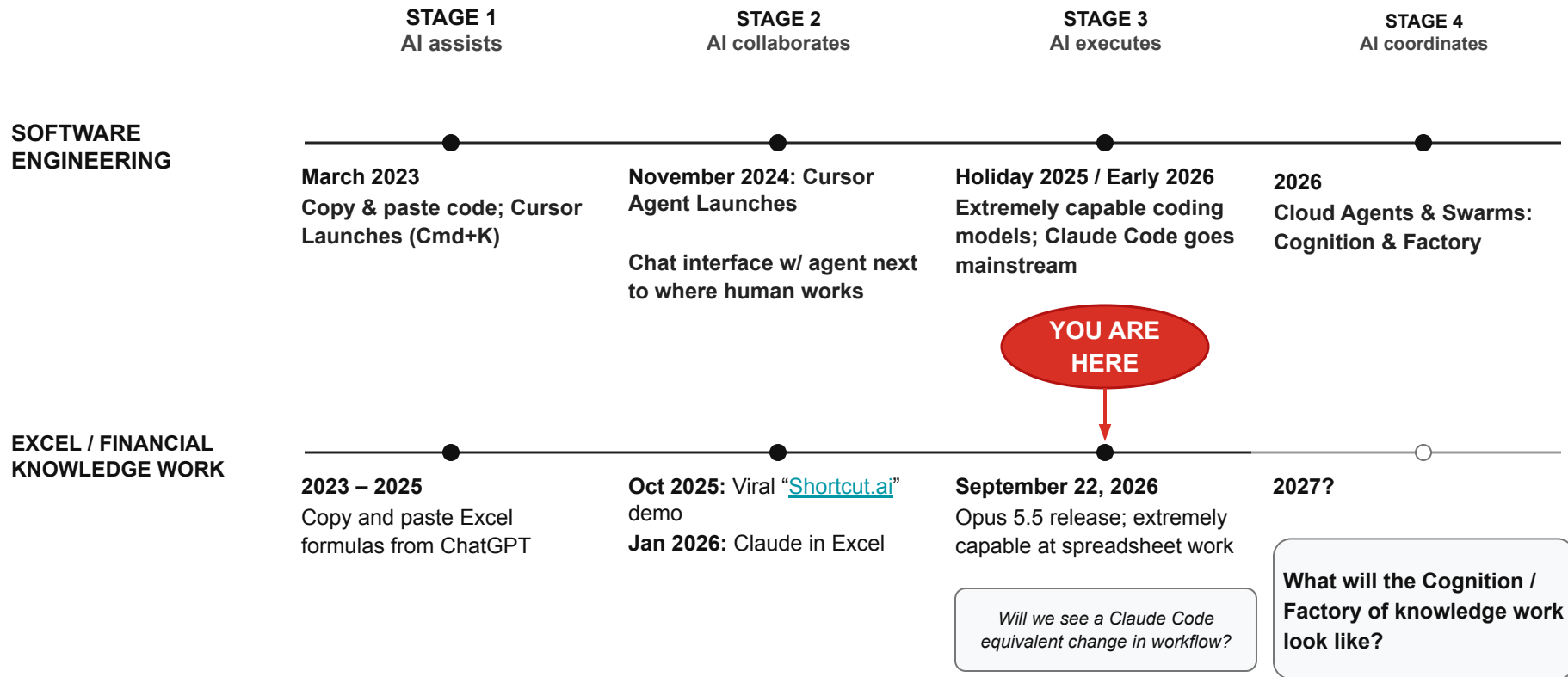
Is Excel approaching its end-to-end agent moment?

Is Excel approaching its end-to-end agent moment?

I.e., how long until the average spreadsheet user no longer has the actual spreadsheet open for a majority of their work?

Evolutionary Parallels: Software Engineering vs. Excel AI

Mapping capability & interaction paradigm stages shows a ~12–18 month adoption lag between code and spreadsheets.



What will the Cognition / Factory of knowledge work look like?

Attributes of “cloud agents:”

- **Independent:** works without your laptop or an open chat.
- **Proactive:** starts work from schedules, events, or incoming requests.
- **Scalable:** distributes & delegates work across multiple independent workers.

How to prepare for changing capabilities & interaction paradigms: lessons from software teams

Lesson from software

How to prepare

How to prepare for changing capabilities & interaction paradigms: lessons from software teams

Lesson from software

Bottlenecks shift (e.g., from code production to code review).

How to prepare

Anticipate new emphasis on reviewing the work of AIs.

How to prepare for changing capabilities & interaction paradigms: lessons from software teams

Lesson from software

Bottlenecks shift (e.g., from code production to code review).

Institutional knowledge becomes necessary infrastructure.

How to prepare

Anticipate new emphasis on reviewing the work of AIs.

Invest effort to make implicit firm knowledge written & accessible.

How to prepare for changing capabilities & interaction paradigms: lessons from software teams

Lesson from software

Bottlenecks shift (e.g., from code production to code review).

Institutional knowledge becomes necessary infrastructure.

Individuals can do more; delegating well becomes a core skill (everyone becomes responsible for getting value from their AI budget).

How to prepare

Anticipate new emphasis on reviewing the work of AIs.

Invest effort to make implicit firm knowledge written & accessible.

Lines get blurred: train managers on AI tools and ICs on delegation.

A smarter approach to AI cost controls: allocating spend by efficiency

COGNITION | OCTOBER 19, 2026

<https://devin.ai/blog/efficiency-based-cost-controls>

Blanket usage caps are akin to a hiring freeze in the AI era.

Many enterprises have turned to uniform caps to control AI spend, but they are blunt instruments that limit productivity. They hold back the engineers who are the most efficient users of AI. And engineers who use AI less effectively don't get feedback on what to change.

The better approach is to allocate spend based on engineering efficiency and give less efficient engineers precise suggestions on how to improve.

This has historically been hard to manually scale. So, today we're introducing three enterprise features to help:

- **Efficiency Score grades how efficiently each engineer uses Devin.** For each session, including investigations and planning that never produce a PR, Devin estimates the associated productive engineering hours, meaning how long a human engineer would have needed for the same work. The grade reflects the productive hours an engineer gets per unit of spend across recent sessions, with a smaller credit for volume.
- **Devin Coach gives engineers feedback on a prompt before it runs.** It flags common mistakes, such as a vague request, an expensive mode for a simple task, or unrelated tasks in one session, and suggests a fix.
- **Efficiency-based approvals raise limits automatically for efficient engineers.** This enables more precise cost controls without ballooning admins' review burden. When an engineer with a Healthy or Satisfactory grade nears their limit, Devin raises it automatically, up to a ceiling the admin sets. Other engineers' requests are sent to an admin for review.

A smarter approach to AI cost controls: allocating spend by efficiency

COGNITION | OCTOBER 19, 2026

<https://devin.ai/blog/efficiency-based-cost-controls>

Blanket usage caps are akin to a hiring freeze in the AI era.

Many enterprises have turned to uniform caps to control AI spend, but they are blunt instruments that limit productivity. They hold back the engineers who are the most efficient users of AI. And engineers who use AI less effectively don't get feedback on what to change.

The better approach is to allocate spend based on engineering efficiency and give less efficient engineers precise suggestions on how to improve.

This has historically been hard to manually scale. So, today we're introducing three enterprise features to help:

- **Efficiency Score grades how efficiently each engineer uses Devin.** For each session, including investigations and planning that never produce a PR, Devin estimates the associated productive engineering hours, meaning how long a human engineer would have needed for the same work. The grade reflects the productive hours an engineer gets per unit of spend across recent sessions, with a smaller credit for volume.
- **Devin Coach gives engineers feedback on a prompt before it runs.** It flags common mistakes, such as a vague request, an expensive mode for a simple task, or unrelated tasks in one session, and suggests a fix.
- **Efficiency-based approvals raise limits automatically for efficient engineers.** This enables more precise cost controls without ballooning admins' review burden. When an engineer with a Healthy or Satisfactory grade nears their limit, Devin raises it automatically, up to a ceiling the admin sets. Other engineers' requests are sent to an admin for review.

Efficient user of AI

→ More AI spend

A smarter approach to AI cost controls: allocating spend by efficiency

COGNITION | OCTOBER 19, 2026

<https://devin.ai/blog/efficiency-based-cost-controls>

Blanket usage caps are akin to a hiring freeze in the AI era.

Many enterprises have turned to uniform caps to control AI spend, but they are blunt instruments that limit productivity. They hold back the engineers who are the most efficient users of AI. And engineers who use AI less effectively don't get feedback on what to change.

The better approach is to allocate spend based on engineering efficiency and give less efficient engineers precise suggestions on how to improve.

This has historically been hard to manually scale. So, today we're introducing three enterprise features to help:

- **Efficiency Score grades how efficiently each engineer uses Devin.** For each session, including investigations and planning that never produce a PR, Devin estimates the associated productive engineering hours, meaning how long a human engineer would have needed for the same work. The grade reflects the productive hours an engineer gets per unit of spend across recent sessions, with a smaller credit for volume.
- **Devin Coach gives engineers feedback on a prompt before it runs.** It flags common mistakes, such as a vague request, an expensive mode for a simple task, or unrelated tasks in one session, and suggests a fix.
- **Efficiency-based approvals raise limits automatically for efficient engineers.** This enables more precise cost controls without ballooning admins' review burden. When an engineer with a Healthy or Satisfactory grade nears their limit, Devin raises it automatically, up to a ceiling the admin sets. Other engineers' requests are sent to an admin for review.

Efficient user of AI

→ More AI spend

Inefficient user of AI

→ Coaching on how to be more efficient

How to prepare for changing capabilities & interaction paradigms: lessons from software teams

Lesson from software

Bottlenecks shift (e.g., from code production to code review).

Institutional knowledge becomes necessary infrastructure.

Individuals can do more; delegating well becomes a core skill ([everyone becomes responsible for getting value from their AI budget](#)).

Greater individual leverage changes optimal team design.

How to prepare

Anticipate new emphasis on reviewing the work of AIs.

Invest effort to make implicit firm knowledge written & accessible.

Lines get blurred: train managers on AI tools and ICs on delegation.

Be open to rethinking team composition.