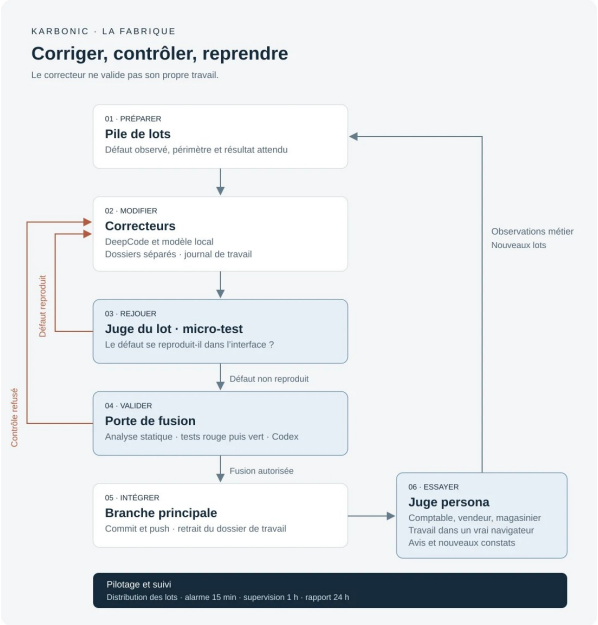
LOGICIELS ET IA

Die Fabrik: ein Jahr Entwicklung mit Agenten

Du grand prompt à une équipe d'agents qui travaille presque seule : les essais, les modèles locaux, les juges, les erreurs, et ce qui tient après une an

Publié le 12 novembre 2026 — Imprimé le 27 novembre 2026



Veröffentlicht am 12. November 2026 · Karbonic

Mit Agenten arbeiten? Nichts Magisches. Es ist wie bei einem Team: Man muss vertrauen, aber … genau kontrollieren. Innerhalb eines Jahres ist die Werkstatt vom grossen Prompt, der «einfach mal so» abgeschickt wurde, zu Agenten gelangt, die vollständige Verwaltungssoftware fast allein voranbringen.

Man sieht sehr beeindruckende Demonstrationen. Der Stadtplan aus der Vogelperspektive, gebaut aus OpenStreetMap und ein paar Anweisungen: Man startet den Agenten am Abend, am nächsten Morgen schaut man sich das Ergebnis an. Das ist wirklich verblüffend.

Eine Unternehmenssoftware ist etwas anderes. Verkauf, Einkauf, Lager, Buchhaltung, Löhne, Stunden, Kundendienst: viel Code, viele Daten und Bereiche, die voneinander abhängen. Die Funktionen müssen zusammenhalten, die Daten stimmig bleiben, und die Benutzerin oder der Benutzer muss arbeiten können, ohne die Kniffe der Entwicklerin oder des Entwicklers zu kennen. Eine solche Software beurteilt man nicht beim Aufwachen, nach einem Gesamteindruck.

Die Agenten produzieren Code mit beeindruckender Geschwindigkeit. Das sagt noch nichts darüber, ob die Änderungen richtig sind, ob sie das gestellte Problem lösen oder ob sie im Nachbarbereich ein neues schaffen. Die ganze hier beschriebene Organisation ist aus diesen Fragen entstanden.

Der Blickwinkel

Eine Ausbildung zur Wirtschaftsinformatikerin oder zum Wirtschaftsinformatiker lehrt zuerst ein Handwerk: die Anfrage einer Kundin oder eines Kunden in Software übersetzen. Verstehen, wie eine Buchhalterin, ein Verkäufer oder eine Lagerverwalterin arbeitet, und es dann so schreiben, dass eine Maschine es ausführt, ohne den Bedarf zu verfälschen.

Der berufliche Weg hat anschliessend beide Seiten eines Entwicklungsteams gezeigt. Auf der einen Seite die Entwicklerin oder der Entwickler, Mitglied von Teams an grossen Systemen. Auf der anderen der IT-Verantwortliche: Verwaltung von Infrastrukturen, Servern und Netzwerken, Führung von Entwicklungsteams.

Diese beiden Seiten nützen jeden Tag im Umgang mit den Agenten. Der Entwickler weiss, was ein gut zugeschnittenes Los erleichtert und was eine unklare Anweisung hervorbringt. Der Verantwortliche weiss, dass ein Team so viel wert ist wie das, was man kontrolliert, und dass eine Fertigmeldung keine Abnahme ist. Die Agenten ändern an diesen Grundsätzen nichts. Sie zwingen dazu, sie mit viel mehr Strenge anzuwenden, weil sie schnell arbeiten und nie zweifeln.

Ein Jahr in vier Phasen

Der grosse Prompt. Wie alle beginnt man mit einem grossen Prompt und schaut, was passiert. Damals wurden uns Prompt-Manager verkauft und die Idee, man müsse vor allem sehr gut im Formulieren werden. Für eine kleine Anwendung ist das ideal: «Analysiere diese Rechnung und übernimm sie» funktioniert sehr gut. Einen Agenten alles allein machen zu lassen und erst am Schluss zu beurteilen, trägt nicht mehr, sobald mehrere Bereiche gleichzeitig zu bearbeiten sind. Entscheidend ist nicht die Formel des Prompts. Es kommt darauf an, von Anfang bis Ende eine Denklogik beizubehalten: aufteilen, ordnen, überprüfen.

Man hat auch Repositories aufblühen sehen, die ein fast magisches Reverse Engineering einer bestehenden Software versprachen. Bei einer Verwaltungssoftware mit ihren Dutzenden von Tabellen und ihren Geschäftsregeln muss der Beweis noch erbracht werden. Vielleicht muss man versierter sein, um Nutzen daraus zu ziehen. Die Erfahrung der Werkstatt geht in eine andere Richtung: Es ist ein gutes, gut geführtes Team, das beginnt, ein brauchbares Ergebnis zu liefern.

Mehrere Modelle parallel. Dann kommt die Phase, in der man alles ausprobiert: Claude, Codex, Gemini, GLM, Kimi, Grok, DeepSeek über DeepCode. Jedes hat seine Stärken. Manche sehr leistungsfähige und günstige Modelle brauchen eine enge Führung; andere denken lange nach, wo eine einfache Arbeit genügt hätte. Mehrere Agenten ohne gemeinsame Regeln am selben Repository arbeiten zu lassen, führte zu Arbeiten, die sich widersprachen oder sich gegenseitig überschrieben.

Zu viele Sicherungen. Die natürliche Antwort war, einzurahmen. Wächter, Judges, Sperrklinken, Orchestratoren, Monitore: Bis zu dreizehn Mechanismen bestanden nebeneinander, und am Ende blockierten sie sich gegenseitig. Ein Wächter lehnte wegen eines Zählfehlers alle Zusammenführungen ab; ein Orchestrator lief leer, bis die Festplatte voll war. Es musste ausgedünnt werden.

Die Fabrik. Die heutige Organisation ist einfacher: kleine Lose, austauschbare Agenten, jeder in seinem Bereich, unabhängige Judges, ein Pilot, der verteilt, und ein lesbarer Bericht. Sie läuft auf vier Maschinen, bei Tag und bei Nacht. Das Gespräch mit einem Modell dient nur noch den Entscheiden.

In Arbeit, mit Fortschritten fast jeden Tag: OpenClaw und Hermes, um die Agenten untereinander zu verbinden und ihnen ein gemeinsames Gedächtnis zu geben.

Was die Frontier-Modelle sehr gut können

Code schreiben, natürlich. Aber eine ihrer nützlichsten Begabungen liegt woanders: sich in jemanden hineinzuversetzen. «Guten Tag, ich bin Martine, seit zwanzig Jahren Buchhalterin. Ich muss die Lieferantenrechnungen des Monats erfassen und den Abschluss vorbereiten.» Ein Frontier-Modell spielt diese Rolle mit überraschender Treffsicherheit. Es sucht die Funktionen dort, wo eine Buchhalterin sie suchen würde, ärgert sich über dieselben Umwege und nennt, was sie daran hindert, ihre Arbeit zu beenden.

Diese Fähigkeit macht heute den Persona-Judge der Werkstatt aus, der weiter unten beschrieben wird. Sie ersetzt die technischen Tests nicht. Sie misst etwas anderes: Kann eine Person aus dem Fach mit dieser Software arbeiten?

Welches Modell für welche Daten

Die Wahl eines Modells richtet sich nicht nur nach seiner Leistung. Sie hängt zuerst davon ab, was man ihm anvertraut. Es gibt keinen Grund, vertrauliche Daten an ein amerikanisches oder ein chinesisches Modell zu übermitteln. Es gibt dagegen viele Gründe, ihre Leistung

zu nutzen, um Code zu schreiben oder ein Bild ohne Brisanz zu bearbeiten.

Die gewählte Einteilung hat drei Stufen:

| Art der Information | Wo sie verarbeitet wird | Beispiele | | --- | --- | --- | | Vertraulich: Rechtsakten, Buchhaltung, Löhne, Bilder eines Unternehmens | Lokal, auf den Maschinen der Werkstatt | Zusammenfassung von Dossiers, Produktbildbearbeitung | | Geschäftlich, ohne Geheimnis, aber mit Verbleib in der Schweiz | In der Schweiz gehostete Modelle (Infomaniak) | Lesen von Rechnungen, in die Software integrierter Assistent | | Quellcode, fiktive Testdaten | Frontier-Modelle, amerikanisch oder chinesisch | Schreiben und Korrigieren von Code, Geschäftssimulationen, Audits |

Jedes Modell hat Vor- und Nachteile, und alles entwickelt sich sehr schnell: Die beste Wahl von vor drei Monaten ist es oft nicht mehr. Daher der Wert, von keinem abhängig zu sein, und der Wert von Modellen mit offenen Gewichten, die man selbst hosten, untersuchen und spezialisieren kann.

Lokale KI: ein Jahr Versuche

Die Dokumente. ClearMyDocs, die Engine der Werkstatt für Dokumentenzusammenfassungen, diente als erster Prüfstand. Anfang 2026 wurden auf einer Grafikkarte mit 8 GB mehrere kleine Modelle an einem echten Rechtsdossier mit fünfzig Dokumenten verglichen. Manche passten schlicht nicht in den Speicher. Andere liefen, erzeugten aber einen ausufernden und wenig verlässlichen Text oder lasen das falsche Dokument. Die kleinsten gewannen an Geschwindigkeit, was sie an Genauigkeit verloren.

Der Prüfstand wurde anschliessend um eine obere Referenz ergänzt: ein deutlich grösseres, in der Schweiz gehostetes Modell, um zu wissen, wie das Mögliche bei derselben Aufgabe aussieht. Ausserdem wurde Ollama zugunsten von llama.cpp verlassen, um genau zu kontrollieren, was an das Modell geschickt wird, ohne verstecktes Template.

Die nützlichste Lehre kam nicht von den Modellen. Bei einem Versuch leerte eine Folge überschrittener Zeitlimiten die Suchachsen, und die Kette meldete eine Abdeckung von 100% … auf null Achsen. Sie hielt an, nachdem sie ein einziges Dokument gelesen hatte, und erklärte sich dennoch für vollständig. Ein Modellwechsel hätte daran nichts geändert. Die Kette musste überarbeitet werden, danach kam eine deterministische Kontrolle hinzu: Jeder extrahierte Fakt wird gegen das Quelldokument geprüft, bevor er in eine Zusammenfassung eingeht.

Seit etwa sechs Monaten arbeitet ClearMyDocs v4 mit einem Modell bescheidener Grösse, das sich auf gängiger Hardware korrekt laden lässt und echte Daten verarbeitet, ohne «Marmelade zu liefern». Dieselbe Engine kann sich auch auf ein in der Schweiz gehostetes Modell oder ein Frontier-Modell stützen, ohne dass sich am Rest etwas ändert. Die Treuekontrolle hängt von keinem Modell ab, wodurch sich die Modelle an ein und demselben Dossier ehrlich vergleichen lassen.

Der Code. Auf handelsüblichen Grafikkarten mit 8 bis 12 GB, verteilt auf mehrere Maschinen, laufen verkleinerte, quantisierte Modelle, die für kurze Aufgaben nützlich sind. Ein ernsthaftes Code-Modell passt dort nicht hinein. Daher der Kauf eines Mini-PCs mit 128 GB Unified Memory, installiert unter Fedora mit llama.cpp.

Bei dieser Art von Maschine begrenzt der Durchsatz des Speichers, nicht seine Grösse. Ein grosses dichtes Modell mit 70 Milliarden Parametern erzeugt dort einige Wörter pro Sekunde: in einer Agentenschleife unbrauchbar. Modelle mit Mixture-of-Experts (MoE) ändern die Lage: Alle ihre Gewichte passen in den Speicher, aber nur ein kleiner Teil arbeitet für jedes erzeugte Wort. Am Ende einer Versuchsreihe wurde Qwen3 Coder Next gewählt, angebunden an OpenCode.

Es bearbeitet kleine, kurze und begrenzte Lose und gehört wie die anderen Agenten zum Team: elf Lose in den letzten vierundzwanzig Stunden. Es ersetzt die gehosteten Modelle bei den grossen Vorhaben nicht, wo sehr viele Dateien sehr schnell gelesen werden müssen. Es übernimmt seinen Teil der laufenden Arbeit, ohne Abonnement, und nichts verlässt das Haus.

Die nächste Spur: einen Teil der Geschäftssimulationen einem lokalen Modell anvertrauen, mit einem Frontier-Modell als Judge zweiter Stufe. Das Kriterium ist im Voraus festgelegt: Wenn das lokale Modell bei einer gleichen Simulation mindestens 80% der Mängel findet, die das Frontier-Modell findet, übernimmt es.

Das Training. LoRA-Versuche laufen. Diese kleinen Anpassungsschichten spezialisieren ein Modell, ohne es vollständig neu zu trainieren. Die Idee ist, dem lokalen

Modell beizubringen, was bereits als gut beurteilt wurde: akzeptierter Code, freigegebene Bildschirme, ein Produkt, das ohne Verzerrung darzustellen ist. Die Arbeit vorab einrahmen, statt hinterher zurückgewiesen zu werden.

Ein automatisierter Bildablauf

Das andere grosse lokale Vorhaben betrifft die Produktfotografie in der Uhrmacherei. Ziel ist es, die Handgriffe einer professionellen Retuscheurin oder eines professionellen Retuscheurs zu automatisieren: das offizielle Foto einer Uhr nehmen und in eine Szene einfügen (aufgestellt auf einer Theke, auf Holz, auf Wasser), ohne je eine Ziffer des Zifferblatts, einen Zeiger oder ein Glied des Armbands zu verzerren. Die Bilder eines Unternehmens dürfen nicht abfliessen: Alles läuft lokal.

Der Ablauf folgt der Reihenfolge der manuellen Arbeit in Photoshop:

1. **Freistellen**, Teil für Teil: Zifferblatt, Gehäuse, Hörner, Krone, Glieder.
2. **Eine Ebene pro Objekt**, mit zusammenhängenden Pixeln, wie es eine Retuscheurin tun würde.
3. **Positionieren, dann verformen**: Winkel, Perspektive, Verringerung der Gliederzahl.
4. **Das Material nur dort neu bearbeiten**, wo es nicht sauber ist, etwa eine schlecht erkannte gebürstete Fläche.
5. **In die Szene einsetzen**, in der richtigen Grösse.
6. **Schatten und neue Beleuchtung.**

Jeder Schritt hat sein Werkzeug. Beim Freistellen erwies sich CLIPSeg als zu grob, bleibt aber nützlich, um Bereiche zu erkennen. SAM 2.1 und BiRefNet liefern ein feines Freistellen, Metallkanten eingeschlossen. Die Erzeugung der Hintergründe und die Bearbeitung laufen über ComfyUI, ohne Oberfläche über seine API gesteuert, wobei mehrere Modellfamilien nacheinander ausprobiert wurden: SDXL mit ControlNet, FLUX.2, Qwen-Image. Blender wird per Skript automatisiert. Es modelliert die Uhr nicht: Es berechnet das Licht, die Schatten, die Reflexe und die Geometrie der Positionierung. OpenCV wendet die Verformungen an.

Die Judges werden vor jeder Einstellung festgelegt. Ein Pixel-für-Pixel-Vergleich prüft, dass sich das Zifferblatt gegenüber dem Original nicht bewegt hat. Ein Vision-Modell beurteilt, ob die Szene glaubwürdig ist. Wenn man die Ziffern nicht lesen kann oder die Uhr nicht echt wirkt, geht das Bild nicht durch.

Die Fehler waren lehrreich. Ein generatives Modell, dem man die Geometrie überlässt, erfindet die Form: ein plausibles, aber falsches Gehäuse. Die echten Pixel Zeile für Zeile wieder aufzukleben, ergibt das Gegenteil: Risse und eine «Pixelsuppe». Die richtige Arbeitseinheit ist das Objekt, wie bei einer Retuscheurin. Eine weitere Lehre: Die Lizenzen ändern sich von einer Version zur nächsten. Ein Modell, das unter freier Lizenz veröffentlicht wurde, kann in seiner nächsten Version zu einer nichtkommerziellen Forschungslizenz zurückkehren. Man prüft, bevor man integriert.

Dieser Ablauf ist noch ein Prototyp. Er liefert bereits realistische Inszenierungen, aber noch nicht die Regelmässigkeit eines Produktionsablaufs. Die Methode dagegen ist dieselbe wie beim Code: protokollierte Schritte, im Voraus festgelegte Judges und ein Journal dessen, was funktioniert und was nicht.

Der aktuelle Ablauf

Der Stapel von Losen. Jeder Bereich der Software hat eine Kontrollmatrix. Ein Feld, das nicht besteht, wird zu einem Los: ein aus Sicht des Benutzers beschriebenes Ziel, eine gemessene Feststellung und die benachbarten Funktionen, die funktionsfähig bleiben müssen. Technische Spuren werden als Hypothesen festgehalten, die die Beobachtungen ausschliessen können. Das Los bleibt klein, auch wenn der dem Agenten gelieferte Kontext gross ist. «Mach das Modul neu» lässt zu viel Interpretationsspielraum und macht das Ergebnis unkontrollierbar.

Die Korrekturagenten. DeepCode erledigt den Grossteil der Korrekturen auf zwei dedizierten Maschinen; das lokale Modell übernimmt die kleinen Lose. Jeder Agent arbeitet in seinem eigenen Ordner und in einem festgelegten Dateibereich. Er hält fest, was er tut, während der Arbeit und nicht nur am Ende: Wenn die Sitzung abbricht, bleiben die Versuche und Zwischenergebnisse verfügbar. Ein gemeinsames Postfach, das jeder vor dem Start liest, ersetzt wiederholte Anweisungen.

Der Mikrotest. Ein Agent, der meldet, er sei fertig, liefert einen Bericht, keine Freigabe. Der Korrekturagent schliesst sein eigenes Los daher nie. Ein anderer Agent spielt zuerst den Fehler in der Oberfläche nach: etwa fünf Minuten, gegenüber fünfundvierzig für einen voll-

ständigen Durchlauf. Tritt das Problem wieder auf, geht das Los an den Korrekturagenten zurück.

Das Zusammenführungstor. Es vereint die statische Analyse, die Tests und ein von Codex abgegebenes Urteil. Ein Test, der den Fehler abdecken soll, muss vor der Korrektur als gescheitert und danach als bestanden gesehen worden sein. Ohne diese Gegenprobe kann er bestehen, ohne etwas zu erkennen. Nach der Freigabe stösst die Änderung zum Hauptzweig, und der Arbeitsordner wird entfernt.

Der Persona-Judge. Eine Reihe freigegebener Korrekturen beweist nicht, dass eine Software brauchbar ist. In regelmässigen Abständen übernimmt ein Modell die Rolle einer Buchhalterin, eines Verkäufers oder einer Lagerverwalterin und benutzt die Software in einem echten Browser. Es erhält eine zu erledigende Arbeit, ohne dass man es um Nachsicht bittet. Sein Urteil betrifft das, was es tun konnte, und das, was es daran gehindert hat. Seine Beobachtungen werden wieder zu Losen. Von Zeit zu Zeit geht ein neuer Judge, der nichts über den bisherigen Verlauf weiss, hinter den bestehenden Judges noch einmal durch: Diese akzeptieren manchmal eine Schwierigkeit schlicht, weil sie sie schon gesehen haben.

Der Pilot. Auf dem Hauptrechner verteilt ein Pilot die Lose. Ein Alarm ohne KI prüft alle Viertelstunde, dass kein Prozess stillschweigend ausgefallen ist; ein Supervisor läuft jede Stunde; ein Bericht fasst die letzten vierundzwanzig Stunden zusammen. Die Git-Repositories, die Zweige und die Ergebnisse der Kontrollen geben Zugang zu den Einzelheiten: Man muss nachvollziehen können, warum eine Änderung akzeptiert oder abgelehnt wurde, ohne Dutzende von Gesprächen nachzulesen.

Was Zeit gekostet hat

Das in YAML beschriebene Team. Die ersten Versuche beschrieben die gesamte Organisation in YAML-Dateien: Prompt-Manager, Bereiche, Verantwortliche, Entwickler, Judges. Auf dem Papier war alles da. Es hat nicht funktioniert.

Der Chef-Agent. Ein Agent, der für alle anderen nachdenken und ihre Arbeit verteilen sollte, lieferte nie ein zufriedenstellendes Ergebnis. Ein Pilot, der präzise Lose verteilt, ist nützlich; ihm das gesamte Verständnis des Projekts anzuvertrauen, ist es nicht.

Die Punktzahlen. «1’980 Tests von 1’983» sagt nicht, ob die drei Fehlschläge die Arbeit verhindern oder ob die anderen Tests prüfen, was zählt. Die Liste der Blockaden samt Ursache ist für die Entscheidung über das Weitere viel nützlicher.

Die Wechsel des Orchestrators. Eine Übergabe an ein anderes Modell mitten im Vorhaben verbrauchte den Grossteil eines Kontingents für einen minimalen Fortschritt. Das Verständnis eines laufenden Projekts wiederherzustellen, kostet mehr, als man denkt.

Die Tests, die nicht klicken. Ein Test, der die API abfragt, kann bestehen, während der Bildschirm kaputt ist. Er behält seinen Wert für die Verarbeitungen, aber ein Benutzerablauf wird in der Oberfläche validiert.

Was trägt

- Kleine Lose, jedes in seinem Bereich, mit einem Eingangs- und Ausgangsvertrag.
- Erst die Messinstrumente, dann die Korrekturen: Ein zweifelhafter Bericht erzeugt zweifelhafte Lose.
- Nach zwei erfolglosen Versuchen an demselben Fehler wird der Code instrumentiert, damit er selbst anzeigt, wo er bricht.
- Die Judges und ihre Schwellenwerte werden festgelegt, bevor irgendetwas eingestellt wird.
- Die Validierung erfolgt auf einer frischen Installation, nicht auf einer über Wochen von Hand reparierten Datenbank.
- Man committet und pusht oft; kein Arbeitsordner bleibt verwaist.
- Keine Maschine ruht, solange noch bereite Lose vorhanden sind.

Vierundzwanzig Stunden Arbeit

Vom Morgen des 11. November bis zum Morgen des 12. November haben vier Maschinen 110 Lose gestartet und 86 Zusammenführungen erzeugt. Fünfundachtzig dieser Lose stammten direkt aus den Beobachtungen des Persona-Judge.

Der Judge ist vierzehnmal durchgegangen, über acht Bereiche. Bei jedem Durchgang blieb sein Urteil: noch nicht brauchbar. Genau das ist seine Rolle.

Diese Messung betrifft einen Tag Aktivität, nicht einen Produktivitätsgewinn im Vergleich zu einem menschlichen Team. Sie zeigt das mögliche Arbeitsvolumen und

die Lücke, die zwischen diesem Volumen und einem einsatzbereiten Produkt bestehen bleiben kann.

Für alle, die anfangen

- **Dem grossen Prompt misstrauen.** Er passt für eine kleine Anwendung. Darüber hinaus macht die Aufteilung das Ergebnis, nicht die Formel.
- **Mit der Aufteilung beginnen, nicht mit dem Modell.** Ein gutes Los mit einem durchschnittlichen Modell gibt bessere Ergebnisse als eine vage Anweisung mit dem besten Modell.
- **Einen Agenten nie seine eigene Arbeit freigeben lassen.** Ein zweiter Agent oder ein Mensch stellt fest.
- **Wie ein Benutzer testen.** Klicken, den Bildschirm lesen, die verlangte Arbeit ausführen; ein Frontier-Modell, das die Buchhalterin spielt, macht das sehr gut.
- **Den Zustand aufschreiben lassen.** Ein Agent erinnert sich an nichts; er weiss nur, was aufgeschrieben wurde.
- **Das Modell nach den Daten wählen,** dann nach Aufgabe und Kosten.
- **Das Lokale messen, bevor man daran glaubt.** Die Grösse des Speichers und sein Durchsatz entscheiden, was wirklich läuft.
- **Die Kontrollen regelmässig verschlanken.** Eine Sicherung, die ohne Grund blockiert, kostet so viel wie ein Fehler.

Tauschen wir unsere Erfahrungen aus

Nach einem Jahr hat die Codeproduktion eine andere Grössenordnung erreicht. Die Arbeit der Einrahmung hat sich mit ihr verändert: Lose vorbereiten, Kontrollen pflegen, Rückmeldungen aus der Praxis prüfen, Prioritäten festlegen. Das ist im Grunde das Handwerk der Wirtschaftsinformatik: einen Bedarf in Software übersetzen und überprüfen, dass die Software den Bedarf auch erfüllt.

Dieses System ist nicht starr, und es ist zu einem guten Teil auf Fehlern aufgebaut. Viele Teams und Selbstständige führen heute dieselben Versuche durch, jedes in seiner Ecke. Unsere Methoden zu vergleichen, würde allen Zeit sparen: wie Sie Ihre Agenten organisieren, wie Sie deren Arbeit beurteilen, welche Modelle Sie lokal behalten und warum, was funktioniert hat und was gescheitert ist. Erfahrungsberichte sind willkommen.