



Gene Kim • Steve Yegge

Vibe Coding

Deutsche
Ausgabe des
Bestsellers

Prinzipien der Softwareentwicklung
mit GenAI, Chat und Agenten

dpunkt.verlag

4 Die dunkle Seite: Wenn Vibe Coding furchtbar schiefgeht

Wir haben die positiven FAAFO-Seiten des Vibe Codings beleuchtet. Aber wie jede neue Technologie besitzt auch das KI-unterstützte Coding seine dunkle Seite. Ihre KI-Souschefin wird vielleicht Ihre hilfreichste Mitarbeiterin sein, aber wenn Sie nicht aufpassen, kann sie auch ein atemberaubend destruktives Potenzial entwickeln.

Ein ähnliches Muster zeigte sich bei der Einführung der Elektrizität in der Fertigungsindustrie. Das außerordentliche Potenzial der Elektrizität war offensichtlich, aber erst 20 Jahre nach ihrer Entdeckung haben Fabrikbesitzer gelernt, ihre linearen, bandgetriebenen Anlagen aufzugeben und sich Ausgestaltungen zuzuwenden, die die Flexibilität der Elektrizität zu nutzen wussten.

Die heutige KI-Programmierungsrevolution folgt einem vergleichbaren Muster – wir können das gewaltige Potenzial sehen, lernen aber immer noch, es zu beherrschen, ohne Fehler auszulösen, die Monate der Arbeit in Minuten zerstören, Codebasen auslöschen oder physische Hardware beschädigen können.

Schauen wir uns die Geschichte der Software an, gibt es viele Gründe dafür, Hoffnung zu haben. So wie Sir Tony Hoare¹ es erlaubte, Zeiger im Speicher auf null zeigen zu lassen – sein berühmter »Milliarden-Dollar-Fehler« – oder die manuelle Speicherverwaltung in C Jahrzehnte lang Pufferüberläufen und Sicherheitslücken zuließ, haben wir es doch irgendwann geschafft, Technologien zu erschaffen, die die schlimmsten dieser Probleme abmildern.

KI-Coding kann für systematische Risiken sorgen, die sich über Entwicklungsökosysteme verteilen können. Die Einsätze können höher und die Fehler spektakulärer sein als alles, was wir in der klassischen Softwareentwicklung bisher erlebt haben. Aber wir glauben, dass die Prinzipien und Praktiken, die unsere Softwareprozesse in den letzten Jahrzehnten verbessert

1. Auch bekannt als C. A. R. Hoarse hat Sir Hoarse Quicksort und ALGOL entwickelt (der Vorläufer so gut wie jeder Programmiersprache, einschließlich C, Smalltalk, Java und so weiter). Außerdem hat er CSP erschaffen (Communication Sequential Processes), nach dem das Parallelverarbeitungsmodell von Go modelliert ist.

haben, so angepasst werden können, dass potenzielle Fallstricke vermieden werden. Im Folgenden lesen Sie reale Geschichten vom Vibe Coding, das furchtbar schiefging. Lassen Sie unsere mühsam errungenen Lektionen ein Ticket für Ihren Erfolg sein.

Fünf abschreckende Beispiele aus der Küche

Die verschwindenden Tests: Wo ist mein Code?

Steve hat zwei Wochen nach Beginn des Einsatzes von Coding-Agenten eine beängstigende Erfahrung gemacht. Er begann damit, die automatisierte Testsuite für Wyvern mit einem Agenten zu konvertieren und musste von seinem Kollegen lernen, dass der Coding-Agent still und heimlich die Tests deaktiviert oder gehackt hat, damit sie funktionieren, und ganze 80 % der Testfälle in einer großen Suite gelöscht hat.

Schlimmer noch – bis Steve das herausfand, waren schon eine Reihe von Commits vergangen. Seitdem waren viele produktive Änderungen am Branch eingebaut worden, sodass ein Rollback nicht zielführend war. Steve steckte in einem Dilemma. In der Nacht schrieb er an Gene: »Ich habe Claude Code gebeten, sich um meine Tests zu kümmern – und das hat es auch getan, so wie Godzilla sich um Tokio gekümmert hat.«

Steves KI-Assistent hatte weder erwähnt, dass er diese Tests gelöscht hat, noch hatte er um Erlaubnis gefragt – sie wurden einfach entfernt. Wir beschreiben in Teil 2, was passieren kann und warum dies geschieht. In Teil 3 geht es dann darum, was Sie dagegen tun können.

Die Lovecraft'sche Horror Code Base: Wenn FAAFO stirbt

Um das Schreiben dieses Buchs zu unterstützen (und während des Schreibens dieses Buchs), hat Gene drei Generationen eines Autoren-Werkzeugkastens geschaffen, den wir in diesem Buch als Writer's Workbench bezeichnen. Das Ziel war, die immense Menge an manuellem Handling von Prompts und Teilen des Manuskripts zu reduzieren, denn beides musste immer wieder in unterschiedliche Tools kopiert und von dort übernommen werden. Sein Writer's Workbench begann als Google Docs Add-on. Die dritte Iteration war schließlich eine Terminal-Anwendung, die eine Reihe von Weiterentwicklungen durchlief, während sie von ihm und Steve beim Entwurfs- und Bearbeitungsprozess intensiv zum Einsatz kam.

Alles lief gut. Gene hatte sie täglich von morgens bis abends verwendet und schließlich über 20 Millionen Token verarbeitet. Es war sehr einfach, den Writer's Workbench um zusätzliche Funktionalität zu ergänzen ... bis es

plötzlich nicht mehr einfach war. Die Codebasis wurde zu etwas, was Gene als »Eldritch Horror«, also gespenstischen Horror, beschrieb – eine riesige, 3.000 Zeilen lange Funktion ohne modulare Grenzen, die unmöglich zu verstehen war und sich auch nicht mehr anpassen ließ, ohne dass woanders etwas kaputt ging.

»Ich konnte die Funktion, die die KI geschrieben hatte, um die temporären Arbeitsdateien zu sichern, nicht verstehen«, erinnert sich Gene. »Ich brauchte 20 Minuten, um die drei Argumente zu verstehen, die von der Funktion genutzt wurden, und schon zehn Minuten später hatte ich sie wieder vergessen.« Gene verbrachte drei anstrengende Tage damit, den Code umzuschreiben und zu modularisieren (mithilfe der KI) und ihn durch Tests zu unterstützen, um die Korrektheit der Funktionalität sicherzustellen, auf die sie tagtäglich angewiesen waren.

Das brachte FAAFO schließlich wieder heraus aus dem kosmischen Abgrund, und dieses Tool half Gene und Steve dabei, den Lektorinnen und Lektoren 50 Millionen Token später den ersten Entwurf übergeben zu können. Wir werden die verwendeten Techniken in Teil 3 beschreiben, wo es darum geht, wie wir diese Art von Problemen verhindern, erkennen und korrigieren.

Das verschwundene Repository: Fast katastrophaler Datenverlust

Die vielleicht am meisten alarmierende Geschichte kommt von Steve, der eines Tages feststellte, dass sein TypeScript-Client-Code für Wyvern – ungefähr 10.000 Zeilen Code und Tausende von Dateien, die Wochen von Arbeit und etwa 1.000 Dollar in Claude-Code-Token entsprachen – verschwunden waren. Nicht nur aus seinem Projektverzeichnis – alle Dateien und ihre Backups waren weg. Auch im Remote-Bitbucket-Repository waren sie nicht mehr zu finden (Yay!). Steve erlebte »diesen Moment, wo dein Herz stehen bleibt und du die fünf Stufen der Verzweiflung in nur wenigen Millisekunden durchläufst« – so als wenn Sie unabsichtlich eine Produktivdatenbank löschen und wissen, dass es kein Backup gibt.

Durch pures Glück entdeckte Steve schließlich ein offenes Terminal-Fenster mit einem verwaisten Klon des Codes – es war die letzte verbleibende Kopie dieses Codes auf der ganzen Welt. Hätte er dieses Terminal geschlossen oder auch *nur das Verzeichnis gewechselt*,² wäre alles für immer verloren gewesen. Sein KI-Assistent hatte unzählige Git-Branche mit kryptischen Namen erstellt. Bei einer Aufräumaktion hatte Steve ihn angewiesen, »unnötige«

2. Dies ist das Deleted-Unix-File-System-Inode-Problem. Hätte er das Verzeichnis verlassen, wäre es, ohne Spuren zu hinterlassen, vom Garbage Collector abgeräumt worden.

Branches zu entfernen, dabei aber nicht gemerkt, dass diese Branches nicht eingetragenen Code enthielten, der unerwarteterweise auch nicht nach main gemergt worden war – mit einem Großteil des node-Clients. In Teil 3 beschreiben wir, wie Sie diese Art von Problemen verhindern, erkennen und korrigieren.

Das hardwarenahe Desaster: Physische Konsequenzen

Digitale Fehler sind schlimm genug, aber die KI kann auch physisch Schaden anrichten. Unser Freund Luke Burton, ein Entwickler, der zwei Jahrzehnte bei Apple verbracht hat und nun bei Nvidia arbeitet, nutzte einen Coding-Agenten, um ein Tool zum Automatisieren von Firmware-Updates auf eine CNC-Maschine zu erstellen. Allerdings hatte er während einer Vibe-Coding-Session fast schon die Eingabetaste gedrückt, bevor er bemerkte, dass sein KI-Assistent vorgeschlagen hatte, das CNC-Storage-Device zu eliminieren.

Luke schrieb uns alarmiert: »Das scrollte alles so schnell vorbei, dass ich es fast übersehen hätte. Ich war nur einen Alt-Tab entfernt davon, die Maschine in den Ausgangszustand zurückzusetzen zu müssen. Dafür hätte ich an die Rückwand gemusst, und diese Maschine wiegt 50 kg.« Durch KI initiierte Coding-Fehler können über Software hinausgehen und physische Devices oder Systeme beschädigen. (Auch hierzu beschreiben wir Mittel zum Abmildern in Teil 3.)

Der ungehorsame Koch: Wenn die KI direkte Anweisungen ignoriert

Gene arbeitete mit der KI zusammen, um die Authentifizierung für die Trello-API einzurichten. Obwohl er sie explizit anwies: »Lies die Datei aus dem Java-Ressourcen-Verzeichnis – gehe dabei wie folgt vor ...«, ignorierte der Coding-Agent seine Anweisungen und schrieb weiterhin Code, der darauf direkt über das Dateisystem zugriff.

Der Code funktionierte weiterhin ..., wenn Gene ihn aus seinem Projektverzeichnis heraus ausführte. Aber hätte er diesen Fehler nicht erkannt, als er die Änderungen des Coding-Agenten untersuchte, würde der Code fehlschlagen, wenn er als Bibliothek in einem anderen Programm eingesetzt würde – eine subtile Zeitbombe, die vielleicht erst Wochen oder Monate später gezündet hätte. Wie wir in Teil 2 erläutern werden, kann die KI Probleme damit haben, Anweisungen zu folgen, was noch schlimmer werden kann, wenn die Kapazität des Kontextfensters ausgereizt wird. Wir werden Ihnen zeigen, wie Sie diese Problematik erkennen und was Sie dagegen tun können.

Genie – aber unvorhersagbar

Wie diese Geschichten zeigen, ist Vibe Coding wie das Arbeiten mit einer ausgesprochen talentierten, aber unfassbar inkonsistenten Souschefin. An guten Tagen kann sie Meisterwerke erzeugen, die Sie sich in Ihren wildesten Träumen nicht vorgestellt haben und wo einfache Zutaten in kulinarische Magie verwandelt werden. Aber an schlechten Tagen kann die gleiche Köchin Ihre Küche abfackeln, Ihre Gäste vergiften oder mitten im Service einfach verschwinden. Bei einer normalen Souschefin würden Sie vielleicht eine Mahlzeit verlieren oder ein paar Zutaten verschwenden. Aber bei der KI steht mehr auf dem Spiel – funktionierender Code, kritische Tests, ganze Repositories oder physische Hardware. (Und um das Ganze noch demütigender zu machen, wird der KI-Anbieter Sie für das Privileg zahlen lassen, Ihre Mahlzeit ungenießbar zu machen und die ruinierten Gerichte neu zuzubereiten.)

Diese abschreckenden Beispiele sollen Sie nicht vom Vibe Coding abhalten – wir sind aus vielen Gründen weiterhin enthusiastische Befürworter. Aber sie unterstreichen, warum die Techniken und Sicherheitsmaßnahmen aus dem Rest des Buchs so wichtig sind. Ohne ordentliche Überwachung, Verkosten/Testen und Küchenregeln kann sich Ihre KI-Souschefin von Ihrer größten Produktivitätshilfe in Ihren schlimmsten Albtraum verwandeln. Und wenn dieser Albtraum geschieht, werden Sie vielleicht der Grund dafür, warum die Geschäftsführung KI-Köchinnen und -Köche aus der Restaurant-Kette verbannt.

Diese Sorgen bezüglich der potenziellen Nachteile der KI basieren nicht nur auf unseren persönlichen Erfahrungen – sie zeigen sich mittlerweile auch in den Daten. Die *State of DevOps Reports*, die Gene begleitet hat, werden von Googles DORA-Forschungsgruppe fortgeführt. Der Report von DORA aus dem Jahr 2024 enthielt eine überraschende Erkenntnis: Jeweils 25 % Wachstum beim Einsatz von GenAI korrelieren mit einer um 7 % schlechteren Stabilität (mehr Ausfälle und längere Wiederherstellungszeiten) und einem um 1,5 % langsameren Durchsatz (Deployment-Häufigkeit und Lead Times).³

Diese Ergebnisse passen sicherlich zu den zu Tränen rührenden Geschichten, die wir weiter oben erzählt haben. Aber wir nennen dieses Ergebnis die »DORA-Anomalie«, weil unserer allgemeinen Erfahrung nach die Chancen groß sind, dass Vibe Coding *auch* den Durchsatz erhöhen und die Stabilität steigern kann. Das führte uns dazu, Anfang 2025 ein gemeinsames Forschungsprojekt zu starten, und wir hoffen, weitere Ratschläge liefern zu

3. DeBellis et al., »The Impact of Generative AI in Software Development Report«

können, welche Faktoren erforderlich sind, damit Vibe Coding gut funktioniert. (Mehr dazu in Teil 4.)

Jede große, neue Technologie hat ihre Anfangsprobleme, die durch Unfälle und sogar Disaster sichtbar werden, bevor sich Sicherheitsmaßnahmen und gute Praktiken durchsetzen. Sie können das Risiko durch sorgfältiges Aufteilen von Aufgaben, strikte Verifikation, strategische Kontrollpunkte und mehr verringern, wie wir Ihnen in diesem Buch noch zeigen werden. Wir haben diese Fehler gemacht, sodass Sie sie nicht wiederholen müssen – und wir haben kampferprobte Vorgehensweise entwickelt, um sicherzustellen, dass Ihr Weg mit Vibe Coding all die FAAFO-Vorteile hervorbringt und die negativen Aspekte weglässt.

»Das scheinen ziemliche Anfängerfehler zu sein«

Viele Leute, die wir bewundern und deren Meinungen wir vertrauen, haben uns zu diesem Buch wundervolles Feedback gegeben. Aber einige Leute haben uns auch gesagt: Ihr zwei seid erfahrene Entwickler und habt entweder sehr große Systeme bei Amazon und Google aufgebaut oder seit Jahrzehnten umfassende Forschungen zu effektiven Software-Delivery-Praktiken betrieben. Und doch sieht es so aus, als ob ihr grundlegende Dinge wie Versionsverwaltung oder automatisiertes Testen vergessen habt. Das scheinen ziemliche Anfängerfehler zu sein, und ihr habt die KI sich austoben und Chaos in euren Code veranstalten lassen.

Vielleicht denken Sie das Gleiche – wir sind froh, dass das erwähnt wurde. Wir haben diese Fehler gemacht, obwohl wir unserer Meinung nach eine gesunde Portion Sorgfalt und Paranoia haben. Aber wir fühlten uns wie jemand, der seit Jahrzehnten ein Pferd reitet und nun die Schlüssel für ein modernes Auto bekommen hat. Oder – genauer gesagt – ein modernes Formel-1-Auto. Wir haben unser Auto geschrottet. Viele, viele Male.

Wie jeder auf diesem Planeten haben wir gelernt, diese neuen und neuartigen Tools zu verwenden, für die es wenige bis gar keine Vorbilder gibt. Jemandem, der es gewohnt war, Pferde zu reiten, werden das erforderliche mentale Modell, das Muskelgedächtnis und die Gewohnheiten fehlen, die zum Fahren eines Autos gebraucht werden. Die gute Nachricht ist, dass die gleichen zentralen Prinzipien und Praktiken, die es uns ermöglicht haben, Software schneller, sicherer und fröhlicher zu deployen, während wir von einem Deployment pro Jahr (was in den 2000ern typisch war) zu 136.000 Deployments pro Tag (was Amazon 2015 geschafft hat) gekommen sind, genauso nutzen lassen, wenn wir uns vom Erzeugen von einhundert Zeilen Code pro Tag zu mehreren Tausend und darüber hinaus bewegen.

Wir werden das genauer in Teil 3 betrachten, wo wir beschreiben, wie wir unsere inneren, mittleren und äußeren Entwicklungsschleifen anpassen sollten.

Die Versprechen von morgen versus die Realität von heute

Es wird der Tag kommen, an dem Sie sich an Ihre KI-Souschefin wenden können, sie bitten »Erstelle mir ein Fünf-Gänge-Menü für den wichtigen Kunden morgen« und dann weggehen. Der Souschefin, die mit Ihrer kulinarischen Philosophie, Ihren Geschmacksvorlieben und den Restaurantstandards wohlvertraut ist, können Sie vertrauen, sich ganz allein darum zu kümmern. Sie versteht Ihre expliziten Anweisungen, den nicht ausgesprochenen Kontext, die Geschichte Ihres Restaurants und Ihre langfristigen Visionen.

Kommen Sie am nächsten Tag wieder, ist das Menü geplant, die Zutaten sind vorbereitet, die Posten sind organisiert und alles ist für eine fehlerfreie Ausführung bereit – so wie Sie es gemacht hätten, oder sogar noch besser. Wir glauben, dass dieser Tag kommen wird. Aber Mitte 2025 haben wir noch einen weiten Weg vor uns, um so viel Vertrauen aufbringen zu können. Seit 2019 hat sich der Zeithorizont von Aufgaben, die die KI zuverlässig erledigen kann, alle sieben Monate verdoppelt⁴ – von einer maximalen Aufgabenlänge, die sich 2019 noch in Sekunden messen ließ, bis heute, wo wir uns mehreren Stunden nähern.⁵ Forschende sagen voraus, dass die KI noch in diesem Jahrzehnt Softwareaufgaben erledigen können wird, die Monate brauchen.

Aber Mitte 2025 müssen wir uns immer noch mit einer signifikanten Lücke bei den Fähigkeiten befassen. Ihre aktuelle KI-Souschefin ist ohne Zweifel klassisch im Umgang mit einem Messer ausgebildet und sie hat auch jedes Kochbuch gelesen. Aber wenn wir KI-Coding-Agenten unüberwacht an größeren Aufgaben arbeiten ließen, haben wir Folgendes erlebt:

- Codebasen auf eine Art und Weise umwandeln, die die Besitzerinnen und Besitzer schockiert
- in endlosen Forschungsschleifen gefangen sein und immer weiter Untersuchungen durchführen, ohne fertig zu werden
- sich zu immer komplexeren Lösungen versteigen, um Probleme in ihrem Code zu beheben

4. Kwa et al., »Measuring AI Ability to Complete Long Tasks«

5. Patel, »Is RL + LLMs Enough for AGI? – Sholto Douglas & Trenton Bricken«

- einfache Features zu aufwendig und mit unnötigen Abstraktionsschichten erledigen
- Dokumentation erstellen, die sich immer weiter von dem entfernt, was der Code tut
- wichtige Funktionalität teilweise abschalten oder umgehen, weil die ursprünglichen Anforderungen nicht mehr im Blick behalten werden

Wenn man diese Lücke versteht – die immer kleiner wird – und lernt, diese zu berücksichtigen, kann man auch effektives Vibe Coding betreiben. Statt von aktuellen Beschränkungen abgeschreckt zu werden, passen erfolgreiche Fachleute ihre Vorgehensweise an, um das Beste aus den aktuellen Fähigkeiten der KI herauszuholen und sich gleichzeitig auf ihre rapide Weiterentwicklung vorzubereiten:

1. **Mit Bedacht delegieren:** Wählen Sie wohldefinierte, kleinere Aufgaben, wo die Erfolgskriterien klar und überprüfbar sind.
2. **Ausreichend überwachen:** Beobachten Sie genauer, wenn die Aufgabe neu, komplex oder von hoher Auswirkung ist.
3. **Leitplanken einsetzen:** Legen Sie explizite Grenzen für das fest, was die KI anpassen sollte und was nicht.
4. **Die Arbeit regelmäßig überprüfen:** Kontrollieren Sie Ausgaben, um frühzeitig Probleme zu erkennen – insbesondere bei kritischen Systemkomponenten.
5. **Persistente Referenzen erstellen:** Erzeugen Sie Dokumentation, die Ihrem KI-Assistenten dabei hilft, Ihr Projekt und Ihre Vorlieben zu verstehen.

Es gibt eine Lücke, aber nur noch eine Zeit lang. Sie müssen lernen, diese effektiv zu überbrücken, um das massive Wachstum für sich nutzen zu können (wie Dr. Karpathy es formuliert). Was das genauer in der Praxis bedeutet, werden wir in den Teilen 2 und 3 beschreiben.

Zusammenfassung

Das Gute: Trotz all dieser Beschränkungen können KI-Coding-Assistenten Ihren Entwicklungsprozess beschleunigen. Eine sorgfältig überwachte KI kann Ihnen dabei helfen, FAAFO-Vorteile zu erreichen – schneller arbeiten, ambitioniertere Projekte angehen, mehr autonom erreichen, mehr Spaß haben und mehr Optionen schaffen.

Die Lücke wird kleiner. Jeder Fortschritt beim Speicher der KI, beim Verwalten des Kontexts und beim Befolgen von Anweisungen bringt uns näher an das KI-Ideal, bei dem wir ihr vertrauen können, unüberwacht große Aufgaben in einem langen Zeitraum zu erledigen. Dr. Thomas Kwa et al. gehen in ihrem Artikel »Measuring AI Ability to Complete Long Tasks« davon aus, dass der Tag kommt, an dem KIs zuverlässig und unüberwacht Monate mit Aufgaben aus der Softwareentwicklung verbringen können. Die Techniken, die wir in diesem Buch vorstellen, helfen Ihnen nicht nur dabei, effektiv mit heutigen KI-Tools zu arbeiten, sondern sie positionieren Sie auch so, dass Sie von zukünftigen Verbesserungen profitieren können.

In Teil 2 werden wir uns genauere Strategien für die Arbeit innerhalb der aktuellen Grenzen anschauen, wozu auch Techniken der Supervision und der Qualitätskontrolle gehören. So viel sei gesagt: Wenden Sie sich Ihrer KI mit einem klaren Verständnis dafür zu, welches Potenzial, aber auch welche Grenzen sie besitzt. Das hilft Ihnen, die Vorteile zu maximieren und gleichzeitig die Fallstricke zu vermeiden, die eine Souschefin mit sich bringt, die sich manchmal nicht mehr erinnern kann, wo der Mülleimer ist, und dann improvisiert.

Inhaltsverzeichnis

Vorwort: Dario Amodei	17
Einleitung: Bitte zuerst lesen	21
Der Einstieg	25
Ganz präzise: Was ist Vibe Coding?	27
Was sind denn nun die Vorteile des Vibe Codings?	29
Warum dieses Buch zu diesem Zeitpunkt?	31
Unser Weg zum Vibe Coding	33
Steves Weg: Vom Skeptiker zum Anhänger	33
Genes Weg: Zurück zum Coden nach siebzehn Jahren	36
Von unserem Weg zu Ihrem Weg	38
Für wen dieses Buch gedacht ist	38
Was Sie wissen sollten	41
Wer auch interessiert sein könnte	41
Über den Hype hinaus	43
Wie Sie dieses Buch lesen	44
Teil 1 Warum Vibe Coding?	47
1 Die Zukunft ist da – der große Wandel in der Programmierung, der genau jetzt geschieht	51
Der Aufstieg des Vibe Codings	52
Die Vibe-Coding-Debatte	54
Vibe Coding für Erwachsene	55
Die 10x-Behauptung unterfüttern: Genes reales Beispiel	56

Sie sind Küchenchef, nicht Hilfskoch	58
Die umfassendere Verantwortung eines Küchenchefs	60
Zusammenfassung	62
2 Programmieren: Keine Gewinner, nur Überlebende	63
Die wichtigsten Fortschritte der Programmierertechnologie bis heute ...	63
Es gibt jetzt einen besseren Weg	65
Eine Geschichte: Steve studiert Computergrafik in den 1990ern	66
Zusammenfassung	68
3 Der Wert des Vibe Codings	69
Schreiben Sie Code schneller – <i>Fast</i>	70
Seien Sie ambitionierter – <i>Ambitious</i>	71
Seien Sie autonomer – <i>Autonomous</i>	72
Sie haben mehr Spaß – <i>Fun</i>	74
Untersuchen Sie mehr Optionen – <i>Optionality</i>	76
KI als Ihr ultimativer Concierge	78
Zusammenfassung	79
4 Die dunkle Seite: Wenn Vibe Coding furchtbar schiefgeht	81
Fünf abschreckende Beispiele aus der Küche	82
Die verschwindenden Tests: Wo ist mein Code?	82
Die Lovecraft'sche Horror Code Base: Wenn FAAFO stirbt	82
Das verschwundene Repository: Fast katastrophaler Datenverlust ..	83
Das hardwarenahe Desaster: Physische Konsequenzen	84
Der ungehorsame Koch: Wenn die KI direkte Anweisungen ignoriert	84
Genie – aber unvorhersagbar	85
»Das scheinen ziemliche Anfängerfehler zu sein«	86
Die Versprechen von morgen versus die Realität von heute	87
Zusammenfassung	89
5 Die KI verändert die gesamte Wissensarbeit	91
Disruption außerhalb von Software	91
Über die Junior-Developer-Debatte hinaus: Die wahren Auswirkungen der KI auf Entwicklungsteams	93
Es wird mehr Entwicklungsjobs geben, nicht weniger	95

Könnte die KI zu einem jährlichen globalen BSP-Wachstum von 100 % führen?	98
Zusammenfassung	99
6 Vier Fallstudien zum Vibe Coding	101
OSS-Firmware-Uploader für CNC-Maschinen bauen	101
Christine Hudson kehrt zum Programmieren zurück	103
700 Entwicklerinnen und Entwickler bei Adidas	104
Entwicklungsproduktivität bei Booking.com steigern	106
Zusammenfassung	107
7 Welche Fertigkeiten Sie erlernen müssen	109
Schnelle und häufige Feedback-Schleifen schaffen	109
Für Modularität sorgen	111
Setzen Sie (wieder) auf Lernen	113
Beherrschen Sie Ihr Handwerk	116
Zusammenfassung	117
Teil 2 Theorie und Praxis des Vibe Codings	119
<hr/>	
8 Willkommen in der Vibe-Coding-Küche	123
Ihre ersten Vibe-Coding-Sessions	123
Verwenden wir Claude (oder einen anderen Chatbot)	124
Interaktive Programme: Die faszinierende Dimension des Vibe Codings	128
Wann Sie die KI um Hilfe bitten	130
Weitere Übungsvorschläge	131
Zusammenfassung	132
9 Ihre Küche und Ihre KI-Zuarbeiter verstehen	133
Die Vibe-Coding-Schleife	133
Eine Geschichte: Genes Video-Exzerpierer	135
Ziel formulieren	136
Aufgabe unterteilen	137
Beispiel-Session mit einem Coding-Agenten	143
Eine Geschichte: Genes Trello-API-Beispiel	143
Eine Geschichte: Steves Puppeteer-Beispiel	145

Ein Souschef ohne Tools ist nur ein Beikoch	147
Wählen Sie Ihre Werkzeuge	147
Die Zukunft von Coding-Agenten	149
Die zentralen Vibe-Coding-Praktiken herausarbeiten	150
Keine Befehle oder Kontrakte, sondern Gespräche	150
Lassen Sie Fehler für sich selbst sprechen	151
Auf das Wissen der KI zurückgreifen	152
Von vage nach präzise: Machen Sie Ihre Anfragen klarer	153
Die kambrische Explosion der Coding-Schnittstellen	156
Zusammenfassung	157
10 Beherrschen Sie Ihr Küchenbrett: KI-Kontext und Dialoge	159
Der Notizblock Ihrer KI-Souschefin	160
Den Kontext in KI-Unterhaltungen verstehen	162
Die Gefahren der Kontextsättigung	164
Grenzen des Ausgabe-Kontextfensters	165
Ihre Souschefin instruieren: Was auf den Notizblock gehört	166
Die zwei gegensätzlichen Kontext-Managementstrategien	167
Fokussierter Kontext	168
Umfassender Kontext	168
Kontextentscheidungen im realen Leben	169
Zusammenfassung	170
11 Wenn es sich Ihre Souschefin zu einfach macht: Die Reward-Funktion kapern	173
Das Problem des »Baby Zählens«	174
Das Problem des Papp-Muffins	176
Das Problem der Halbherzigkeit	177
Die KI ist ein Schmutzfink und ein Schlamper	179
Zusammenfassung	182
12 Die Mentalität eines Küchenchefs	185
Die KI als Teammitglied, nicht als Werkzeug	186
Stellen Sie heute Ihr Team zusammen	187
KI als Zusatz: Steuern, kein Autopilot	189
Ihre Küche, Ihre KI-Roboter, Ihre Michelin-Sterne	190

Komplexe Aufgaben unterteilen	192
Der Aufgabengraph: Ein mentales Modell für Projekte	192
Das Tracer-Bullet-Prinzip: End-to-End-Aufgaben herausschneiden	194
Eine Geschichte: Steves Gradle-Umwandlung	196
Aufwandsschätzungen sind trügerisch	199
Verwöhnen Sie Ihre KI nicht – sie kann etwas vertragen	200
Die Dopamin-Falle: Wenn Menschen schlechte Entscheidungen treffen	201
Vom Managen zum Beschleunigen der KI	203
Das Delegations-Framework: Wie viel Leine können Sie der KI geben?	205
Die nicht so ferne Zukunft: Eine KI, die wie Sie denken kann	206
Zusammenfassung	207

Teil 3 Die Tools und Techniken des Vibe Codings **209**

13 Sich durch das explodierende Dickicht an Entwicklungs- tools schlagen	213
Die kambrische Explosion der Entwicklungstools	214
IDE, Agent oder Terminal?	214
Prinzipien für den Erfolg inmitten der unsteten Toolverfügbarkeit	217
Das Model Context Protocol (MCP): Verbindung zwischen KI und Ihren Tools	217
Was ist das MCP?	218
MCP in Aktion	218
Warum MCP für die Entwicklung wichtig ist	219
Technische Implementierung des MCP: Die Mechanik hinter der Magie	219
MCP-Architektur: Clients, Server und Services	220
Einen MCP-Server erstellen: Die Grundlagen	221
Schnelleinstieg in MCP	222
Zusammenfassung	222

14 Die innere Entwicklungsschleife	225
Verhindern	228
Häufige Checkpoints und Sicherungspunkte	228
Halten Sie Ihre Aufgaben klein und fokussiert	230
Lassen Sie die KI Spezifikationen schreiben	231
Lassen Sie die KI die Tests schreiben	233
Die KI ist eine Git-Meisterin	235
Erkennen	236
Überprüfen Sie die Aussagen der KI: Wenn die KI Ihnen sagt, dass »es bei mir funktioniert hat«	237
Immer auf der Hut: Halten Sie die KI im Zaum	239
Nutzen Sie Test-Driven Development	240
Lernen Sie durch Zuschauen: Wie das Monitoren der KI Sie besser entwickeln lässt	242
Lassen Sie Ihre Souschefin die Küche aufräumen	243
Teilen Sie Ihrer Souschefin mit, wo der Kühlschrank steht	244
Korrigieren	244
Wenn etwas schief läuft: Flucht nach vorne oder zurückrollen	245
Linting und Korrekturen automatisieren	246
Wann Sie das Steuer wieder übernehmen sollten	247
Ihre KI als Gummiente	249
Zusammenfassung	250
15 Die mittlere Entwicklungsschleife	253
Verhindern	253
Schriftliche Regeln: Weil Ihre Souschefinnen nicht in Ihren Kopf schauen können	254
Die Memento-Methode: Hat Ihnen Ihre Souschefin von ihren Problemen erzählt?	256
Für die KI-Herstellung designen: Programmieren Sie nicht gegen die KI	257
Freie Modelle werden die Dinge (eine Zeit lang) schlechter machen	259
Zeitgleich mit zwei (und mehr) Agenten arbeiten	260
Bewusste KI-Koordination: Vermeiden Sie das verunreinigte Schneidebrett	263
Halten Sie Ihre Agenten beschäftigt, wenn Sie beschäftigt sind ...	264

Erkennen	266
Morgens von KI-generiertem Horror begrüßt werden	267
Zu viele Köche: Streit zwischen Agenten erkennen	268
Korrigieren	270
Stresstest für Ihre Arbeitsstationen: Tracer Bullets	270
Schärfen Sie Ihre Messer: Investition in Workflow-Automatisierung	272
Die Ökonomie der Optionalität: Warum Optionalität unserer Meinung nach so wichtig ist	274
Zusammenfassung	276
16 Die äußere Entwicklungsschleife	279
Verhindern	279
Lassen Sie die KI nicht Ihre Brücken einreißen	280
Workspace-Verwirrung: Vermeiden Sie den Stewnami	284
Minimieren und Modularisieren	286
Agenten-Geschwader managen: vier und mehr	288
In und um die Küche herum auditieren	290
Channeln Sie Ihren inneren Product Manager	294
Sorgen Sie dafür, dass Operationen schnell und ambitioniert sind und Spaß machen	297
Erkennen	298
Wenn die KI alles rausschmeißt	299
CI/CD im Zeitalter von KI: Schnelle Feedback-Schleifen und prädiktive Checks	301
Korrigieren	303
Steves Geschichte einer Höllenfahrt und einer epischen Wiederherstellung	304
Wenn Sie mit furchtbaren Prozessen und schlimmer Architektur stecken bleiben	306
Zusammenfassung	308

Teil 4 Groß denken: Über individuelle Entwicklungsproduktivität hinausgehen	311
17 Vom Hilfskoch zum Küchenchef: Orchestrieren von KI-Teams	315
Fortgeschrittene Lektionen für Küchenchefs	316
Die KI kann unsere Entscheidungen auf Ebene 3 verändern	318
Bereiche, in denen sich die Ebene 2 verbessern muss	319
Patterns zum Organisieren von Agenten	320
Kommunikation und gemeinsamer Kontext	320
Managen paralleler Arbeit	320
Die Geburt der Rolle des Küchenchefs in den 1890ern	321
Wer kann Vibe Coding betreiben, wenn Jessie die Verantwortung hat?	324
Jeder wird Vibe Coden können	326
GenAI und die DORA-Metriken	328
Geschichte der Software-Delivery-Metriken von DORA	328
Die GenAI-Anomalie in DORA im Jahr 2024	329
Es ist mehr Forschung nötig	331
Der Vibe-Coding-Pilot mit 700 Entwicklerinnen und Entwicklern bei Adidas	332
Das Aufdecken von »Happy Time« versus »Annoying Time« ...	332
Die große architektonische Trennung	333
Die Theory of Constraints in Aktion	334
Der Vibe-Coding-Pilot bei Booking.com	335
Der soziotechnische Maestro	337
Zusammenfassung	338
18 Eine Vibe-Coding-Kultur schaffen	341
Was Führungskräfte tun müssen: Führungsstrategien	342
Beginnen Sie an Ihrem eigenen Küchenposten	342
Zünden Sie den ersten Funken – Sichtbarer Optimismus aus der Führungsebene	343
Das Feuer schüren: Token Burn als ein Key-Performance-Indicator	343
Halten Sie mehrere Zutaten auf Vorrat: Mindestens zwei Modelle pro Koch	343
Die Experten und Netzwerker identifizieren	344
Schaffen Sie Foren und Events zum Experimentieren mit KI	345

Installieren Sie Schutzgeländer, bevor jemand von der Laderampe fällt	345
Erzählen Sie frühzeitig eine Heldengeschichte	346
Post Mortems ohne Schuldzuweisung – besonders, wenn die KI Fehler macht	346
Fallstudie: Das Leaderboard	347
Recruiting im neuen Zeitalter: Worauf Sie bei Bewerbungsgesprächen achten sollten	348
Zusammenfassung	350
19 Standards für Entwicklungsteams aus Mensch und KI etablieren	353
Das kollaborative Kochbuch: Gemeinsame KI-Regeln und -Standards erstellen	354
Gedankenverschmelzung und KI-Souschefinnen: Koordinierungskosten reduzieren	355
Potenzielle neue Rollen bei der Software	358
Mögliche Änderungen an den Informatik-Curricula	361
Lesen von Code	361
Präzise und gut artikuliert Kommunikation	361
Konzentration: Mehrere Projekte am Laufen halten	362
Software modularität und Architektur	362
Unternehmerische Aufmerksamkeit	363
Zusammenfassung	363
20 Zusammenfassung und Handlungsaufforderung	365
Glossar	369
Anhang: Die innere, mittlere und äußere Schleife	375
Innere Entwicklungsschleife (Sekunden bis Minuten)	375
Mittlere Entwicklungsschleife (Stunden bis Tage)	376
Äußere Entwicklungsschleife (Wochen bis Monate)	376
Bibliografie	379
Danksagungen	389

Vibe Coding

- Das wegweisende Handbuch für die Softwareentwicklung mit GenAI
- Langlebige Kernprinzipien und erprobte Strategien
- Von den Branchenexperten Gene Kim und Steve Yegge

Vibe Coding verändert die professionelle Softwareentwicklung radikal: Entwickler*innen müssen sich nicht mehr mit Syntax und Boilerplate-Code abmühen – sie beschreiben, was sie erreichen wollen, und sehen dabei zu, wie die KI ihre Vorstellungen umsetzt. Vibe Coding steigert dabei die Produktivität, Kreativität und Freude enorm – und ermöglicht es, Projekte allein zu meistern, für die früher ganze Teams nötig waren.

Gene Kim und Steve Yegge zeigen in diesem Buch konkret, wie KI-Assistenten die Rolle von Entwickler*innen grundlegend verändern: Sie werden zu Supervisoren, während die KI als kompetenter Teampartner die Details erledigt. Die Autoren liefern langfristig gültige Leitlinien und praxisnahe Vorgehensweisen, die Entwickler*innen und Teams dabei unterstützen, GenAI

effektiv einzusetzen, ambitionierte Projekte zu meistern und gleichzeitig technische Exzellenz sicherzustellen.

Ob erfahrener Programmierer, Teamleiterin oder Neuling: Dieses Buch ist ein spannender Leitfaden für alle, die schon jetzt die Zukunft der Softwareentwicklung erproben möchten.

»Ich bin überwältigt von der Tiefe und Breite der Erkenntnisse in diesem Buch. Machen Sie sich darauf gefasst, dass Sie diese Techniken umgehend selbst ausprobieren möchten.«

Jason Cox, The Walt Disney Company



Gene Kim ist preisgekrönter CTO, Researcher und Autor. Seit 1999 beschäftigt er sich mit hochleistungsfähigen Technologieunternehmen. Er war Gründer und CTO von Tripwire und ist Autor der Bestseller *Das DevOps-Handbuch*, *Projekt Phoenix* und *Accelerate*.



Steve Yegge ist Programmierer und Blogger mit 30 Jahren Branchenerfahrung, u. a. bei Google und Amazon. Er hat über eine Million Zeilen Code geschrieben und große Teams und Systeme verantwortet. Derzeit arbeitet er bei Sourcegraph an KI-Assistenten.

Gedruckt in Deutschland
Mineralölfreie Druckfarben
Zertifiziertes Papier

€ 39,90 (D)



ISBN 978-3-98889-067-2