Das Update des Beststellers
„Transformationale Produkte“

TRANSFORMATIONALE PRODUKTE

APRIL2026

VOM ENDE DER IT, WIE WIR SIE KENNEN, UND WARUM JETZT ENDLICH ALLES GUT WIRD MIT DER DIGITALISIERUNG.

Matthias Schrader

ÜBER DEN AUTOR

Matthias Schrader gehört zu den digitalen Pionieren in Deutschland. Mitte der 1990er-Jahre gründete er SinnerSchrader und entwickelte E-Commerce-Lösungen für Start-ups, deren Produkte in kürzester Zeit börsenreif wurden.

1999 ging SinnerSchrader selbst an die Börse und gehörte zu den wenigen Unternehmen, die den Neuen Markt nicht nur überlebten, sondern sogar aus dieser Zeit gestärkt hervorgingen. 2006 gründete Matthias Schrader die NEXT Conference, die sich innerhalb weniger Jahre als führende Konferenz für die digitale Transformation in Deutschland etablierte. In der Folge unterstützte SinnerSchrader mit über 500 Beratern, Designern und Software-Entwicklern hauptsächlich DAX-Konzerne bei der Entwicklung digitaler Produkte.

2017 übernahm die weltweite Management- und Technologieberatung Accenture das Unternehmen für einen dreistelligen Millionenbetrag. In der Folge war der Autor bis Ende 2022 für das Geschäft von Accenture Interactive/Song in Deutschland, Österreich und der Schweiz verantwortlich. 2024 gründete er gemeinsam mit zahlreichen Weggefährten die AI-native Beratung und Agentur OH-SO Digital in Hamburg, Berlin, München und Prag.

Inhaltsverzeichnis

User Manual

Dieser Text ist auf Denglisch verfasst. Im digitalen Kontext haben wir grundsätzlich mit vielen englischsprachigen Fachbegriffen zu tun und eine strikte Eindeutschung wäre für viele Teile des Fachpublikums unverdaulich gewesen. Wir haben uns bemüht, die zentralen Termini in einem Glossar zusammenzustellen. Bei der ersten Verwendung des Begriffs erscheint er unterstrichen.

Das vorliegende Buch macht an verschiedenen Stellen Rückgriff auf das Buch „Transformationale Produkte“, welches der Autor im Jahr 2017 veröffentlicht hat und zum Standardwerk für die Entwicklung digitaler Produkte im deutschsprachigen Raum geworden ist. Im Folgenden werden jeweils die wesentlichen Aspekte von damals skizziert, soweit sie für das Gesamtverständnis notwendig sind. Das Vorgängerwerk zu kennen, ist daher keine Voraussetzung.

Zudem gibt es in einigen Abschnitten technische Deep Dives. Sie dienen nur der Vertiefung und sind für das Gesamtverständnis ebenfalls nicht zwingend. Überblättern Sie diese ruhig.

Um das Werk aktuell zu halten sind, sämtliche Arbeitsmaterialien und Checklisten digital unter https://XXXXX.com abrufbar.

PROLOGUE

November 2022

Ich erinnere mich an jenen kalten Abend im November. Es war spät, der Arbeitstag längst zerfasert in E-Mails, Slack-Pings und ziellosem Scrollen. Ein Freund schickte einen Link, lakonisch: „Hast du das gesehen?“ Die URL führte zu chat.openai.com.

Chatbots waren so 90er, trotzdem klickte ich mich zur Website und tippte skeptisch: „Erkläre Blockchain in drei Sätzen.“ Die Antwort erschien. Präzise. Strukturiert. Irritierend kompetent.

Ich erhöhte den Einsatz. „Beurteile, wie gut die drei wichtigsten Thesen meines Buches ›Transformationale Produkte‹ von 2017 gealtert sind.“ Das Ergebnis? Ich hätte es in dem Moment nicht besser artikulieren können. Ich machte verwirrt weiter, obwohl ich das Ergebnis bereits ahnte: „Ein Haiku über Artificial Intelligence.“ Es skandierte korrekt. Na klar.

Na klar? Davor hätte ich jeden ausgelacht, der so etwas prophezeit hätte. Jetzt starrte ich auf den Bildschirm wie jemand, der gerade bemerkt hatte, dass hinter dem leuchtenden Panel vor mir eine außerirdische Intelligenz erwacht war. Es fühlte sich an wie ein Riss in der Realität.

Dieselbe Szene wiederholte sich in den folgenden Tagen und Wochen millionenfach. Büros, Küchen, U-Bahnen. Menschen sahen auf ihre Screens und spürten: Ihre Vorstellungen davon, was Computer können, lösten sich gerade auf.

Fünf Tage nach dem Start erreichte ChatGPT eine Million Nutzer. Zwei Monate später durchbrach OpenAI die 100-Millionen-Schallmauer. Keine Technologie und kein Medium verbreiteten sich in der Geschichte schneller.

Das war der ChatGPT-Moment. Der Augenblick, in dem etwas vollkommen Unerwartetes die Labortür aufschlägt und die Welt in neues Licht taucht.

Aber ChatGPT sollte erst der Anfang sein.

Drei Jahre später

Wieder ein später Abend. Wieder saß ich vor dem Bildschirm. Diesmal war es kein Link von einem Freund – ich hatte Claude Code mit dem neuen Opus-4.5-Modell upgedatet, das Anthropic gerade veröffentlicht hatte.

Ich war immer noch in der Desillusionphase des Vibe Codings gefangen. Das ganze Jahr hatten wir die neuen AI Coding Tools bereits im Einsatz. Es begann mit kleinen Prototypen einzelner Funktionen und wuchs bis hin zu MVPs, die sogar in die Produktion gingen. Aber die Arbeit mit den Tools war langsam, fehleranfällig und glücksspielhaft. Die Entropie wuchs mit der Zeit und die Codebasis war irgendwann kaum noch zu managen.

Es schien eine Sackgasse zu sein.

Ich war müde. Ich öffnete ein Terminalfenster und startete Claude Code. Mein Prompt: unsere LLM-Analytics-Plattform mit einer sechsstelligen Anzahl an Programmzeilen komplett aufzuräumen. Robuster, sicherer – ohne das Nutzerverhalten zu ändern. Ein lustlos-aggressiver Brute-Force-Prompt. In den Monaten davor hätte er jede Codebasis zuverlässig geschreddert.

Am nächsten Morgen hatte ich meine abendliche Aktion schon fast vergessen, startete die Analytics-Plattform, die sich an diesem Morgen ein klein wenig flüssiger anfühlte als noch am Tag zuvor. Erst nach einer Stunde fiel mein Blick auf das von vielen Fenstern verdeckte Terminalprogramm. Claude Code meldete, dass es in der Nacht die Codebasis um 40 Prozent reduziert, große Module sauber aufgeräumt und neu strukturiert, Dutzende kleinerer und mittlerer Security-Themen identifiziert und gefixt sowie die Dokumentation upgedatet hatte. Natürlich hätte es auch alles getestet und, da alles fehlerfrei lief, bereits deployed.

Boom. Ich hatte den ganzen Morgen bereits an einer völlig neuen Codebasis gearbeitet.

Ich war ehrlich geschockt. Was passiert hier gerade? Ich dachte an jenen Novemberabend 2022. Damals hatte ich mit den LLMs eine künstliche Intelligenz erlebt, die *denken* mehr oder weniger gut simulieren konnte. Jetzt erlebte ich eine, die *liefern* konnte. Das war keine Simulation mehr. Der Code war tatsächlich neu geschrieben und lief einwandfrei. Entwicklerwochen an Programmierleistung wurden in einem einzigen Prompt verdichtet, der lediglich ein paar Millionen Token an Compute-Leistung triggerte.

Das war der Moment, in dem ich verstand: ChatGPT war erst der Pilot. Jetzt beginnt die eigentliche Serie. Das Skript, das zeigt, wie wir digitale Produkte entwickeln, muss neu verfasst werden.

Davon handelt dieses Buch.

Second Order Effects
Warum dieser Moment anders ist

“Almost all new code written at OpenAI today is written by Codex. We’re at the point where AI writes the code, and humans verify it works.“

— Sam Altman, OpenAI DevDay (Oktober 2025)

Technologische Brüche sind nichts Neues. Das iPhone 2007. Das World Wide Web Mitte der Neunziger. Personal Computer in den Achtzigern. Jeder dieser Momente veränderte Branchen und ließ neue Industrien aus dem Nichts entstehen. Jede Generation veränderte, wie Software entwickelt wurde: Hochsprachen ersetzten Assembler. Objektorientierung ersetzte die prozedurale Programmierung. Agile ersetzte den Wasserfall. DevOps automatisierte den Betrieb. Low-Code versprach einfache Anwendungen für alle.

Aber all diese Umbrüche betrafen Werkzeuge, Methoden und Abstraktionsebenen. Die geistige Kärrnerarbeit – ein Problem vollständig zu durchdringen und in Code zu übersetzen – blieb die Aufgabe des Menschen an der Tastatur. Der Aufwand, Software zu schreiben, hat sich in den letzten Jahrzehnten nicht verringert.

Das ändert sich gerade radikal, weil AI diese geistige Tätigkeit selbst übernimmt. Die Maschine übersetzt jetzt Probleme in Code. Nicht immer fehlerfrei. Nicht jederzeit zuverlässig. Aber gut genug und jeden Monat besser, um die Frage zu stellen: wie sieht eine Welt aus, in der AI das Problem der Software-Entwicklung gelöst hat?

STILLE REVOLUTION

Mit den agentischen AI-Coding-Tools wie Claude Code werden Large Language Modelle (LLMs) zu Produktivitätsmonstern. Sie generieren mittlerweile Code, der direkt einsatzbereit ist.

Diese Erfahrung wiederholt sich gerade rund um den Globus. Entwickler berichten, dass sie innerhalb weniger Wochen von manuellem Coding auf 80 Prozent agentischem AI-Coding umgestiegen sind. Die größte Veränderung ihrer Arbeitsweise, die sie je erlebt haben.

Die breite Öffentlichkeit ahnt von dieser Revolution wenig. Wahr ist aber auch: Die Kluft zwischen dem, was in den AI-zentrierten Softwareunternehmen und den „normalen" IT-Abteilungen passiert, wächst täglich.

Die Ironie: Wer programmiert, programmiert jetzt auf Englisch. Oder eben Deutsch. Nicht in Python, nicht in JavaScript – in natürlicher Sprache. Das kratzt am Ego jener Profession, die bisher programmiert hat. Aber der Hebel ist zu groß, um ihn zu ignorieren.

Die Antwort darauf, wie eine Welt aussieht, in der das Problem der Softwareentwicklung gelöst ist, lautet: Es wird eine radikal neue Welt sein. Dazu muss ich einmal ausholen, warum Softwareentwicklung bisher überhaupt ein Problem war.

Digitale Produkte

2017 schrieb ich ein Buch über transformationale Produkte, das zum Klassiker der digitalen Produktentwicklung wurde. Meine Beobachtung war damals: Die Digitalisierung transformiert nicht nur menschliches Verhalten und das, was Unternehmen tun – sie verändert auch, was Unternehmen sind. Ich verdichtete die Methoden erfolgreicher Silicon-Valley-Unternehmen mit meinen eigenen Erfahrungen in der Entwicklung digitaler Produkte für Startups und mit den digitalen Schnellbooten deutscher Konzerne seit Mitte der 1990er Jahre.

Meine Kernthesen waren:

These 1: Produkte werden zu Diensten. Die Grenze zwischen Produkt und Service bröckelt. Ein Auto ist kein Auto mehr – es ist Mobilität als Dienst. Eine Software ist keine Software mehr – sie ist ein sich ständig wandelndes Erlebnis.

These 2: Dienste werden zu Plattformen. Wer nur verkauft, verliert. Wer Ökosysteme baut, gewinnt. Amazon, Apple, Google – sie alle haben begriffen, dass der eigentliche Wert in der Steuerung von Netzwerkeffekten liegt, nicht im Einzelprodukt.
These 3: Die S-Kurve ist das Schicksal. Jede Technik durchläuft denselben Zyklus: langsamer Beginn, sprunghaftes Wachstum, Sättigung. Wer die Kurve lesen kann, sieht die Zukunft. Wer sie übersieht, wird von ihr überrollt.

These 4: Experiences schlagen Features. Gewohnheitsprägende Produkte – jene, die einen sich selbst verstärkenden Kreislauf aus Nutzung, Belohnung und Gewohnheit schaffen – dominieren die Feature-Inflation.

Das Buch enthielt auch ein umfangreiches Playbook für die Entwicklung digitaler Produkte als Blaupause für all jene, die sich erst ab Mitte der 2010er Jahre auf den Weg machten, ihr Kerngeschäft und ihre Produkte konsequent für eine digitale Welt auszurichten.

Eine Dekade später stellt sich die Frage, die kaum jemand laut ausspricht: Wann hat Ihr letztes Produkt-Release tatsächlich einen messbaren Unterschied gemacht? Nicht „Wir haben geliefert". Nicht „Das Projekt wurde abgeschlossen". Oder gar „Das Ticket ist geschlossen". Stattdessen: Wann hat ein digitales Produkt Ihres Unternehmens zuletzt das Nutzerverhalten verändert, Marktanteile gewonnen oder einen Geschäftsbereich transformiert?

Wenn Sie jetzt nachdenken müssen, sind Sie nicht allein.

Ein durchschnittliches Feature braucht in deutschen Unternehmen 14 bis 18 Monate von der ersten Idee bis zur Livestellung. Die Entwicklungskosten pro Feature sind in den letzten fünf Jahren deutlich gestiegen. Branchenstudien zeigen regelmäßig: Von zehn gestarteten Produktinitiativen erreichen nur zwei bis drei ihr ursprüngliches Ziel. Und Nutzungsdatenanalysen

(etwa von Pendo oder Amplitude) belegen, dass die Mehrheit der Features nur von einem Bruchteil der Nutzer regelmäßig genutzt wird.

Softwareentwicklung ist hart. Und statt alle Energie in die Kreation digitaler Produkte zu investieren, wurde die meiste Energie darauf gelegt, überhaupt „zu liefern" und — mit zunehmenden Schwierigkeiten im Geschäft — immer günstiger zu entwickeln. Das Ergebnis kann man heute im niedrigen Digitalisierungsgrad deutscher Unternehmen und Behörden betrauern.

VERSCHWUNDENER FLASCHENHALS

Digitale Produkte entstehen in crossfunktionalen Teams. Das war mein Mantra – und es stimmt immer noch auf gewisse Weise. Aber das war ein unzulässiger Shortcut. Ich unterschätzte massiv die Schwierigkeit, erprobte Vorgehensmodelle im Kleinen auf große Unternehmenseinheiten zu skalieren.

Der Flaschenhals war immer die Softwareentwicklung. Nicht die Ideen. Nicht das Design. Nicht das Marketing. Sondern die Fähigkeit, Ideen in funktionierenden Code zu übersetzen. Die Entwickler kamen nicht hinterher (wir werden später sehen: Es lag nicht an ihnen). Das Backlog wuchs schneller als die Lieferfähigkeit.

Was taten die Unternehmen? Sie versuchten, den Flaschenhals zu weiten.

Agile versprach: kleinere Pakete, kürzere Zyklen, weniger Verschwendung. Es half – ein bisschen. Aber der Engpass blieb.

SAFe versprach: Skalierung. Wenn ein Team nicht reicht, nehmen wir zehn. Koordination durch Frameworks. Das Ergebnis? Mehr Koordination, mehr Meetings, mehr Overhead – und dieselbe Time-to-Market. Der Flaschenhals wurde nicht geweitet. Er wurde mit Bürokratie ummantelt.

Offshoring versprach: mehr Entwickler, weniger Kosten. Das Ergebnis? Mehr Entwickler, mehr Kommunikationsprobleme, mehr Qualitätsprobleme – und: längere Time-to-Market. Der Flaschenhals wurde nicht geweitet. Er wurde verlagert.

Der Flaschenhals der Softwareentwicklung, der Engpass, um den wir zwanzig Jahre lang unsere gesamte Industrie organisiert hatten, existiert nicht mehr. In der alten Welt gab es einen natürlichen Filter: die Kosten der Umsetzung. Nicht jede Idee konnte verwirklicht werden, weil das Bauen teuer war. Der Filter war auch ein Schutz. Er schützte Organisationen (und einen selbst) vor schlechten Ideen und zwang zu priorisieren. Er sortierte schlechte Ideen aus, bevor sie zu schlechtem Code wurden.

Mit AI hat sich die Gleichung verändert. Was früher der Engpass war gibt es jetzt im Überfluss. Der neue Engpass ist die Fähigkeit zu entscheiden, *welcher* Code geschrieben werden soll. Der Intent – die präzise Formulierung dessen, was wir wollen und warum – wird zum kritischen Pfad. Ein einzelner Product Engineer mit den richtigen Tools liefert heute, wofür früher ein Team Wochen brauchte. Die Produktionskapazität ist explodiert.

Das ist keine inkrementelle Veränderung. Das ist ein Strukturbruch.

Dieses Buch handelt davon, was das bedeutet.

- Was bedeutet es für Teams, wenn der Engpass nicht mehr die Entwicklung ist?
- Was bedeutet es für Prozesse, wenn Releases in Stunden statt in Wochen möglich sind?
- Was bedeutet es für die Entwicklung digitaler Produkte, wenn sich die Randbedingungen dramatisch verändern?

Wenn Sie jahrelang gegen eine Wand gerannt sind und die Wand plötzlich verschwindet – dann stürzen Sie. Oder Sie lernen zu fliegen.

DAS JEVONS-PARADOXON

1865 beobachtete der britische Ökonom William Stanley Jevons einen seltsamen Widerspruch: James Watts' verbesserte Dampfmaschine senkte den Kohleverbrauch pro Energieeinheit um 75 Prozent. Die Erwartung: weniger Kohleverbrauch. Die Wirklichkeit: Der Verbrauch stieg sprunghaft. Die höhere Leistung machte neue Anwendungen erst wirtschaftlich und überkompensierte die Einsparungen beim Kohleverbrauch.

Das Muster wiederholt sich in jeder Technologiewelle:

Als Computing billig wurde, rechneten wir nicht einfach schneller. Die erste Generation von Computern war darauf optimiert, mathematische Berechnungen effizienter durchzuführen – etwa für Artillerietabellen, Versicherungsmathematik und wissenschaftliche Simulationen. Das war der First-Order-Effect: dieselben Aufgaben schneller erledigt. Aber dann entstanden Spreadsheets – nicht als schnellere Buchhaltung, sondern als völlig neues Werkzeug für „Was-wäre-wenn"-Analysen. Es entstanden relationale Datenbanken, die nicht einfach Karteikarten digitalisierten, sondern völlig neue Abfragemöglichkeiten schufen. Das waren Second-Order-Effects: neue Kategorien, die zuvor nicht existiert hatten.

Als Speicher billig wurde, speicherten wir nicht einfach nur mehr Dateien. Plötzlich wurde es wirtschaftlich sinnvoll, jede Suchanfrage zu speichern, jedes Nutzerverhalten zu protokollieren, jede Interaktion zu archivieren. Google wurde möglich – nicht als bessere Suchmaschine, sondern als Unternehmen, dessen Geschäftsmodell auf der Analyse von Milliarden Datenpunkten basiert. Facebook, das gesamte AdTech-Ökosystem, datengetriebene Geschäftsmodelle jeder Art – sie alle existieren nur, weil Speicher so billig wurde, dass das Sammeln und Analysieren von Daten im industriellen Maßstab wirtschaftlich sinnvoll wurde.

Als Bandbreite billig wurde, kommunizierten wir nicht einfach schneller. Es entstanden die Cloud als dominantes Computing-Paradigma, Software-as-a-Service als Geschäftsmodell und Streaming als Ersatz für Besitz. Netflix konnte nur existieren, weil Bandbreite billig genug wurde, dass das Streamen von HD-Video an Millionen von Haushalten gleichzeitig wirtschaftlich tragbar wurde.

Jetzt erleben wir das Jevons-Paradoxon der Softwareentwicklung:

AI macht die Codeerzeugung um Größenordnungen günstiger. Die naive Erwartung: Wir brauchen weniger Entwickler. Die Wirklichkeit: Wir starten viel mehr Projekte. Eine Flut an Software entsteht.

Die Folge klingt paradox:

- Es wird *leichter*, Software zu entwickeln.
- Es wird *schwerer*, sich zu differenzieren.

Wenn jeder bauen kann, wird das *Was* wichtiger als das *Wie*. Der Funke – das, was ein Produkt transformational macht – wird zum entscheidenden Faktor. Nicht die Fähigkeit, Code zu schreiben. Es geht um die Fähigkeit, den richtigen Code zu schreiben.

Das neue Paradoxon: Softwareentwicklung wird leichter – und zugleich *auf andere Weise schwer*.

DIE UNBEQUEME FRAGE

Es gibt eine Frage, die in den Diskussionen über AI und Softwareentwicklung systematisch vermieden wird: Wer verliert?

Jede Technologiewelle hat Gewinner und Verlierer. Die Gewinner sind sichtbar, sie stehen auf Konferenzbühnen. Die Verlierer verschwinden still.

Wer gewinnt? Dieselben, die bei jeder technologischen Disruption gewinnen: kleine, schnelle, autonome Teams mit klarer Ownership. Menschen, die Entscheidungen treffen, nicht Menschen, die Entscheidungen koordinieren.

Wer verliert? Die Koordinationsklasse. Menschen, deren Jobs nur existieren, um die Komplexität zu managen, die von anderen Menschen erzeugt wird, deren Jobs darin bestehen, Komplexität zu managen. Eine Bürokratie, die sich selbst rechtfertigt – Schicht um Schicht, bis niemand mehr weiß, wo die eigentliche Arbeit aufhört und die Verwaltung der Arbeit anfängt. AI entzieht dieser Pyramide gerade das Fundament.

WARUM DIE MEISTEN PROGNOSEN FALSCH SIND

Die Technologiebranche ist voller Prognosen zu AI und Produktivität. „Entwickler werden 40% produktiver". „Wir werden 30% weniger Entwickler brauchen". Diese Zahlen klingen präzise. Sie sind präzise falsch.

Sie sind falsch, weil sie First-Order-Denken anwenden: Wenn X effizienter wird, brauchen wir weniger X. Das Jevons-Paradoxon zeigt: Das Gegenteil passiert. Effizienz führt zu mehr Nutzung, nicht zu weniger. Aber sie sind auch aus einem tieferen Grund falsch: Sie messen das Falsche. „40% produktiver" – gemessen woran? Lines of Code? Features pro Sprint? Pull Requests pro Woche? All diese Metriken messen Output, nicht Outcome. Sie messen, wie viel produziert wird, nicht, ob das Produzierte einen Unterschied macht.

In einer Welt, in der Code billig ist, ist „mehr Code" keine Errungenschaft. Es ist eine Gefahr. Mehr Code bedeutet mehr Komplexität, mehr Wartung, mehr potenzielle Fehler. Die relevante Frage ist nicht „Wie viel

Code produzieren wir?“ sondern „Wie viel Wirkung erzielen wir pro Einheit Komplexität?“.

Diese Frage stellt fast niemand. Weil sie unbequem ist. Weil sie bedeutet, dass die Produktivität neu definiert werden muss. Weil sie bedeutet, dass viele der aktuellen Metriken überflüssig werden.

ZWEI PARADIGMENWECHSEL

Die Produktentwicklung steht vor zwei radikalen Umbrüchen – und beide ereignen sich exakt jetzt.

① Paradigmenwechsel 1: Agile ist in der Krise

Die agilen Methoden, an denen wir über zwei Dekaden gefeilt haben, liefern nicht mehr. Agil zu denken ist nicht verfehlt. Aber was Unternehmen daraus machten, bewahrte vom ursprünglichen Geist nur noch die Begriffe.

Das nächste Kapitel seziert diese Krise. Hier genügt die Diagnose: Die starken Produktunternehmen fragen nicht: „Wie skalieren wir Agilität?“ Sie fragen: „Wie bleiben wir klein genug, um keine skalierte Agilität zu brauchen?“

② Paradigmenwechsel 2: Intent ist der neue Code

Der zweite Bruch betrifft die Softwareentwicklung selbst. Wenn AI-Werkzeuge den kompletten Routine-Code erzeugen und die Umsetzung vom Engpass zur Massenware wird – was bleibt als menschliche Kernkompetenz? Die Antwort: der Intent.

Ein Einwand liegt auf der Hand: War Intent nicht schon immer wichtig? Jeder gute Produktmanager, jede gute Unternehmerin musste schon immer präzise formulieren, was erreicht werden soll.

Der Einwand ist richtig. Und geht doch am Punkt vorbei.

Der Unterschied ist nicht, dass Intent *neu* wäre. Der Unterschied ist, dass Intent jetzt direkt produktiv wird. Früher war Intent der Anfang einer langen Kette: Intent wurde zur Spezifikation, Spezifikation zur Architektur, Architektur zu Code, Code zu Test. Jeder Übersetzungsschritt dauerte Wochen. Jeder Schritt verlor Wissen. Am Ende stand Code, der mit dem ursprünglichen Intent oft nur noch entfernt verwandt war.

Heute kann Intent, wenn er scharf genug gefasst ist, *unmittelbar* in laufenden Code übersetzt werden. Die Kette zurrt zusammen. Was früher zehn Schritte brauchte, braucht jetzt nur einen. Was früher Wochen dauerte, dauert Stunden.

Das ändert die Wirtschaftlichkeit von Grund auf. Früher war Intent billig und Code teuer. Jetzt ist Code billig und Intent teuer. Wenn Output nichts mehr kostet, wird der Input zum knappen Gut.

INPUT WIRD WICHTIGER ALS OUTPUT

In einer Welt, in der der Engpass der Softwareentwicklung nicht mehr existiert, wird es eine kambrische Explosion digitaler Produkte geben. Die Aspekte, die erfolgreiche digitale Produkte ausmachen, bleiben die gleichen. Doch wie diese Produkte gebaut werden, verändert sich von Grund auf. Wie genau, weiß noch niemand. In den letzten drei Jahren seit dem Siegeszug der generischen AI-Modelle haben wir aber Leitplanken entwickelt, die uns sehr geholfen haben, in dieser neuen Welt erfolgreiche Produkte zu bauen. Es ist weniger ein Framework. Eher eine Haltung. Der Dreiklang lautet Soul – System – Speed:

Soul ist die Fähigkeit, über digitale Produkte zu denken und zu fühlen – den Funken, der ein Produkt von einer Commodity unterscheidet. Die notwendigen Fähigkeiten: Judgment, Cultural Fluency, Meaning-Setting Narratives.

System sichert Produktionsqualität durch Infrastruktur – Architecture, Testability, Observability, Security, Performance, Reliability –, was den Unterschied zwischen Prototyp und Produktion ausmacht.

Speed entsteht durch Pipeline-Verdichtung – nicht durch bessere Prozesse. Die Pipeline wird radikal reduziert: Perceive → Prompt → Produce → Pitch:

- **Perceive**: Einen Insight wahrnehmen, bevor man formuliert
- **Prompt**: Die AI mit klarem Intent beauftragen
- **Produce**: Direkt produzieren, keine Prototypen
- **Pitch**: Pitchen, was man gebaut hat – nicht was man bauen will

Speed entsteht nicht durch bessere Prozesse. Sie entsteht durch das Streichen von Prozessen.

NEUBAUTEN STATT PFLASTER

Die meisten Organisationen reagieren auf diese Brüche, als handele es sich um kleine Veränderungen. Sie kleben ein AI-Pflaster auf den alten Prozess. Sie schicken Mitarbeiter ins AI-Bootcamp. Sie streuen AI-Kompetenz in Stellenanzeigen.

Die Erschütterungen, die wir erleben, lassen sich nicht durch kosmetische Anpassungen abfedern. Sie gehen an die Substanz, ja, und verlangen oft Neubauten. Die gefährlichste Illusion ist die Illusion der Kontinuität: dass man weitermachen kann wie bisher, nur etwas schneller, etwas digitaler, etwas AI-gestützter. Diese Illusion ist bequem – führt aber aufs Abstellgleis. Nicht alle Rollen, die es heute gibt, werden es morgen noch geben. Nicht alle Karrierewege, die gestern sicher waren, sind es morgen noch.

Das ist kein Grund zur Panik. Es ist ein Grund zum Handeln. Wer heute anfängt, Intent-Formulierung, Nutzerverständnis und strategisches Denken zu entwickeln, wird morgen gefragt sein. Wer wartet, wird von den Ereignissen überrollt.

WAS DIESES BUCH LIEFERT

2017 beschrieb ich die Umwälzung durch digitale Produkte. Heute beschreibe ich die Umwälzung der Produktentwicklung selbst. Ein neues Betriebsmodell für eine neue Wirklichkeit:

Soul – System – Speed. Der Dreiklang als Antwort auf das Jevons-Paradoxon: Soul liefert den Funken, System die Zuverlässigkeit, Speed die Beschleunigung.

Die Intent-to-Production Pipeline. Der klassische Genehmigungsablauf – ob Wasserfall oder SAFe-Bürokratie – ist tot. Er verwechselt Dokumentation mit Fortschritt. Die Pipeline ersetzt ihn: Perceive, Prompt, Produce, Pitch. Speed in der Praxis.

Product Teams. Keine parallel skalierten Teams mehr, keine Abhängigkeiten. Keine Übergaben. Keine Silos. Kleine, selbständige Einheiten, die Verantwortung von Anfang bis Ende tragen.

Outcome-Governance. Schluss mit Feature-Zählen. Schluss mit Velocity-Messung. Schluss mit Output-Theater. Steuerung, die nur eine Frage stellt: Haben wir für den Nutzer und das Business etwas erreicht, das messbar ist?

Dieses Buch beschreibt, was jetzt zu tun ist:

- **Teil I: Der neue Kontext** kartiert das veränderte Gelände durch AI. Kapitel 2 zerlegt, warum Agilität in der Krise steckt. Kapitel 3 untersucht die Umwälzung in der Softwareentwicklung.

- **Teil II: Das neue Betriebsmodell** liefert die Leitplanken für diese neue Wirklichkeit. Kapitel 4 führt Soul – System – Speed als neue Prinzipien ein. Kapitel 5 beschreibt die Intent-to-Production-Pipeline.

- **Teil III: Umsetzung** macht es greifbar. Kapitel 6 führt die Product Teams ein. Kapitel 7 markiert die Risiken und Fehlmuster – den Weg, auf dem diese Umwälzung scheitern kann. Der Epilog versucht einen einen Ausblick auf das zu geben, was kommt, und erweitert den Horizont noch einmal.

WAS BLEIBT

Nicht alles ändert sich. Manche Prinzipien sind zeitlos:

Nutzerzentrierung. Produkte gehören dem Nutzer. Nicht der produzierenden Organisation. Nicht den Prozesse. Das war 2016 richtig. Das ist 2026 richtig. Das wird 2036 richtig sein.

Versuchen und Lernen. Niemand weiß vorher, was funktioniert. Die einzige Methode, es herauszufinden: ausprobieren, messen, lernen. Das Tempo der Versuche ändert sich jetzt. Der Grundsatz nicht.

Menschen vor Prozessen. Menschen bauen Produkte. Nicht Regelwerke. Das beste Framework in den Händen eines unmotivierten Teams erzeugt nur Rauschen.

Outcome vor Output. Was zählt, ist nicht, was wir liefern. Es zählt, was wir erreichen. Features zu zählen ist leicht. Wirkung zu messen ist schwer. Aber nur Wirkung zählt.

Diese Grundsätze bilden das Fundament, auf dem das neue Betriebsmodell steht. Sie sind der rote Faden zwischen 2017 und heute.

FÜR WEN IST DIESES BUCH?

Dieses Buch richtet sich an alle, die noch was vorhaben:

- **Gründerinnen und Geschäftsführer**, die verstehen wollen, wie sich ihre Produktorganisation wandeln muss – oder überflüssig wird
- **Produktverantwortliche**, die begreifen wollen, was AI mit ihrer Rolle macht, bevor die Rolle sie überholt

- **Tech-Leads**, die neue Formen für ihre Teams suchen, weil die alten nicht mehr funktionieren
- **Beratungen und Agenturen**, die ihre Auftraggeber in diese neue Wirklichkeit begleiten wollen, statt sie mit Regelwerken von gestern einzuschläfern

Wenn Sie nur einen Gedanken aus diesem Buch mitnehmen, dann diesen:

Kernbotschaft: Der Flaschenhals in der Entwicklung digitaler Produkte – das Schreiben von Code – ist verschwunden. Agentische AI hat ihn beseitigt. Aber Engpässe verschwinden nicht; sie verschieben sich. Der neue Engpass ist Intent: zu wissen, *was* gebaut werden soll und *warum*. Die Erfahrung lehrt, dass diese Verschiebung zu Second-Order-Effekten führt, die wesentlich größer sind als die Effizienzgewinne, die man zu Beginn prognostiziert hat.

ZUSAMMENFASSUNG: WARUM JETZT ALLES ANDERS IST

Sechs Erkenntnisse:

1. **Zwei Wendepunkte:** ChatGPT (November 2022) zeigte eine AI, die das Denken simulieren konnte. Die neue Generation agentischer AI-Coding-Tools kann bauen. Der eigentliche Strukturbruch.

2. **Das Jevons-Paradoxon:** Bauen wird leichter – sich voneinander zu unterscheiden sehr viel schwerer.

3. **Die zwei Paradigmenwechsel:** Agile steckt in der Krise – SAFe löst das falsche Problem. Intent ist der neue Code – die Übersetzungskette von Idee zu Code schrumpft auf eine Konversation.

4. **Soul – System – Speed:** Der Dreiklang als Antwort.

5. **Die 4P-Pipeline:** Perceive → Prompt → Produce → Pitch.

6. **Was bleibt:** Nutzerzentrierung, Versuchen und Lernen, Menschen vor Prozessen, Outcome vor Output. Die Technologie ändert sich. Die Grundsätze nicht.

Im folgenden Kapitel: Warum Agile in der Krise steckt – und warum die Antwort anders lautet als „mehr Agile".

Los geht's.

Das Update des Beststellers
„Transformationale Produkte“

IT’s OVER

VOM ENDE DER IT, WIE WIR SIE KENNEN, UND WARUM JETZT ENDLICH ALLES GUT WIRD MIT DER DIGITALISIERUNG.

Matthias Schrader

ÜBER DEN AUTOR

Matthias Schrader gehört zu den digitalen Pionieren in Deutschland. Mitte der 1990er-Jahre gründete er SinnerSchrader und entwickelte E-Commerce-Lösungen für Start-ups, deren Produkte in kürzester Zeit börsenreif wurden.

1999 ging SinnerSchrader selbst an die Börse und gehörte zu den wenigen Unternehmen, die den Neuen Markt nicht nur überlebten, sondern sogar aus dieser Zeit gestärkt hervorgingen. 2006 gründete Matthias Schrader die NEXT Conference, die sich innerhalb weniger Jahre als führende Konferenz für die digitale Transformation in Deutschland etablierte. In der Folge unterstützte SinnerSchrader mit über 500 Beratern, Designern und Software-Entwicklern hauptsächlich DAX-Konzerne bei der Entwicklung digitaler Produkte.

2017 übernahm die weltweite Management- und Technologieberatung Accenture das Unternehmen für einen dreistelligen Millionenbetrag. In der Folge war der Autor bis Ende 2022 für das Geschäft von Accenture Interactive/Song in Deutschland, Österreich und der Schweiz verantwortlich. 2024 gründete er gemeinsam mit zahlreichen Weggefährten die AI-native Beratung und Agentur OH-SO Digital in Hamburg, Berlin, München und Prag.

Inhaltsverzeichnis

User Manual

Dieser Text ist auf Denglisch verfasst. Im digitalen Kontext haben wir grundsätzlich mit vielen englischsprachigen Fachbegriffen zu tun und eine strikte Eindeutschung wäre für viele Teile des Fachpublikums unverdaulich gewesen. Wir haben uns bemüht, die zentralen Termini in einem Glossar zusammenzustellen. Bei der ersten Verwendung des Begriffs erscheint er unterstrichen.

Das vorliegende Buch macht an verschiedenen Stellen Rückgriff auf das Buch „Transformationale Produkte“, welches der Autor im Jahr 2017 veröffentlicht hat und zum Standardwerk für die Entwicklung digitaler Produkte im deutschsprachigen Raum geworden ist. Im Folgenden werden jeweils die wesentlichen Aspekte von damals skizziert, soweit sie für das Gesamtverständnis notwendig sind. Das Vorgängerwerk zu kennen, ist daher keine Voraussetzung.

Zudem gibt es in einigen Abschnitten technische Deep Dives. Sie dienen nur der Vertiefung und sind für das Gesamtverständnis ebenfalls nicht zwingend. Überblättern Sie diese ruhig.

Um das Werk aktuell zu halten sind, sämtliche Arbeitsmaterialien und Checklisten digital unter https://XXXXX.com abrufbar.

PROLOGUE

November 2022

Ich erinnere mich an jenen kalten Abend im November. Es war spät, der Arbeitstag längst zerfasert in E-Mails, Slack-Pings und ziellosem Scrollen. Ein Freund schickte einen Link, lakonisch: „Hast du das gesehen?“ Die URL führte zu chat.openai.com.

Chatbots waren so 90er, trotzdem klickte ich mich zur Website und tippte skeptisch: „Erkläre Blockchain in drei Sätzen.“ Die Antwort erschien. Präzise. Strukturiert. Irritierend kompetent.

Ich erhöhte den Einsatz. „Beurteile, wie gut die drei wichtigsten Thesen meines Buches ›Transformationale Produkte‹ von 2017 gealtert sind.“ Das Ergebnis? Ich hätte es in dem Moment nicht besser artikulieren können. Ich machte verwirrt weiter, obwohl ich das Ergebnis bereits ahnte: „Ein Haiku über Artificial Intelligence.“ Es skandierte korrekt. Na klar.

Na klar? Davor hätte ich jeden ausgelacht, der so etwas prophezeit hätte. Jetzt starrte ich auf den Bildschirm wie jemand, der gerade bemerkt hatte, dass hinter dem leuchtenden Panel vor mir eine außerirdische Intelligenz erwacht war. Es fühlte sich an wie ein Riss in der Realität.

Dieselbe Szene wiederholte sich in den folgenden Tagen und Wochen millionenfach. Büros, Küchen, U-Bahnen. Menschen sahen auf ihre Screens und spürten: Ihre Vorstellungen davon, was Computer können, lösten sich gerade auf.

Fünf Tage nach dem Start erreichte ChatGPT eine Million Nutzer. Zwei Monate später durchbrach OpenAI die 100-Millionen-Schallmauer. Keine Technologie und kein Medium verbreiteten sich in der Geschichte schneller.

Das war der ChatGPT-Moment. Der Augenblick, in dem etwas vollkommen Unerwartetes die Labortür aufschlägt und die Welt in neues Licht taucht.

Aber ChatGPT sollte erst der Anfang sein.

Drei Jahre später

Wieder ein später Abend. Wieder saß ich vor dem Bildschirm. Diesmal war es kein Link von einem Freund – ich hatte Claude Code mit dem neuen Opus-4.5-Modell upgedatet, das Anthropic gerade veröffentlicht hatte.

Ich war immer noch in der Desillusionphase des Vibe Codings gefangen. Das ganze Jahr hatten wir die neuen AI Coding Tools bereits im Einsatz. Es begann mit kleinen Prototypen einzelner Funktionen und wuchs bis hin zu MVPs, die sogar in die Produktion gingen. Aber die Arbeit mit den Tools war langsam, fehleranfällig und glücksspielhaft. Die Entropie wuchs mit der Zeit und die Codebasis war irgendwann kaum noch zu managen.

Es schien eine Sackgasse zu sein.

Ich war müde. Ich öffnete ein Terminalfenster und startete Claude Code. Mein Prompt: unsere LLM-Analytics-Plattform mit einer sechsstelligen Anzahl an Programmzeilen komplett aufzuräumen. Robuster, sicherer – ohne das Nutzerverhalten zu ändern. Ein lustlos-aggressiver Brute-Force-Prompt. In den Monaten davor hätte er jede Codebasis zuverlässig geschreddert.

Am nächsten Morgen hatte ich meine abendliche Aktion schon fast vergessen, startete die Analytics-Plattform, die sich an diesem Morgen ein klein wenig flüssiger anfühlte als noch am Tag zuvor. Erst nach einer Stunde fiel mein Blick auf das von vielen Fenstern verdeckte Terminalprogramm. Claude Code meldete, dass es in der Nacht die Codebasis um 40 Prozent reduziert, große Module sauber aufgeräumt und neu strukturiert, Dutzende kleinerer und mittlerer Security-Themen identifiziert und gefixt sowie die Dokumentation upgedatet hatte. Natürlich hätte es auch alles getestet und, da alles fehlerfrei lief, bereits deployed.

Boom. Ich hatte den ganzen Morgen bereits an einer völlig neuen Codebasis gearbeitet.

Ich war ehrlich geschockt. Was passiert hier gerade? Ich dachte an jenen Novemberabend 2022. Damals hatte ich mit den LLMs eine künstliche Intelligenz erlebt, die *denken* mehr oder weniger gut simulieren konnte. Jetzt erlebte ich eine, die *liefern* konnte. Das war keine Simulation mehr. Der Code war tatsächlich neu geschrieben und lief einwandfrei. Entwicklerwochen an Programmierleistung wurden in einem einzigen Prompt verdichtet, der lediglich ein paar Millionen Token an Compute-Leistung triggerte.

Das war der Moment, in dem ich verstand: ChatGPT war erst der Pilot. Jetzt beginnt die eigentliche Serie. Das Skript, das zeigt, wie wir digitale Produkte entwickeln, muss neu verfasst werden.

Davon handelt dieses Buch.

Second Order Effects
Warum dieser Moment anders ist

“Almost all new code written at OpenAI today is written by Codex. We’re at the point where AI writes the code, and humans verify it works.“

— Sam Altman, OpenAI DevDay (Oktober 2025)

Technologische Brüche sind nichts Neues. Das iPhone 2007. Das World Wide Web Mitte der Neunziger. Personal Computer in den Achtzigern. Jeder dieser Momente veränderte Branchen und ließ neue Industrien aus dem Nichts entstehen. Jede Generation veränderte, wie Software entwickelt wurde: Hochsprachen ersetzten Assembler. Objektorientierung ersetzte die prozedurale Programmierung. Agile ersetzte den Wasserfall. DevOps automatisierte den Betrieb. Low-Code versprach einfache Anwendungen für alle.

Aber all diese Umbrüche betrafen Werkzeuge, Methoden und Abstraktionsebenen. Die geistige Kärrnerarbeit – ein Problem vollständig zu durchdringen und in Code zu übersetzen – blieb die Aufgabe des Menschen an der Tastatur. Der Aufwand, Software zu schreiben, hat sich in den letzten Jahrzehnten nicht verringert.

Das ändert sich gerade radikal, weil AI diese geistige Tätigkeit selbst übernimmt. Die Maschine übersetzt jetzt Probleme in Code. Nicht immer fehlerfrei. Nicht jederzeit zuverlässig. Aber gut genug und jeden Monat besser, um die Frage zu stellen: wie sieht eine Welt aus, in der AI das Problem der Software-Entwicklung gelöst hat?

STILLE REVOLUTION

Mit den agentischen AI-Coding-Tools wie Claude Code werden Large Language Modelle (LLMs) zu Produktivitätsmonstern. Sie generieren mittlerweile Code, der direkt einsatzbereit ist.

Diese Erfahrung wiederholt sich gerade rund um den Globus. Entwickler berichten, dass sie innerhalb weniger Wochen von manuellem Coding auf 80 Prozent agentischem AI-Coding umgestiegen sind. Die größte Veränderung ihrer Arbeitsweise, die sie je erlebt haben.

Die breite Öffentlichkeit ahnt von dieser Revolution wenig. Wahr ist aber auch: Die Kluft zwischen dem, was in den AI-zentrierten Softwareunternehmen und den „normalen" IT-Abteilungen passiert, wächst täglich.

Die Ironie: Wer programmiert, programmiert jetzt auf Englisch. Oder eben Deutsch. Nicht in Python, nicht in JavaScript – in natürlicher Sprache. Das kratzt am Ego jener Profession, die bisher programmiert hat. Aber der Hebel ist zu groß, um ihn zu ignorieren.

Die Antwort darauf, wie eine Welt aussieht, in der das Problem der Softwareentwicklung gelöst ist, lautet: Es wird eine radikal neue Welt sein. Dazu muss ich einmal ausholen, warum Softwareentwicklung bisher überhaupt ein Problem war.

Digitale Produkte

2017 schrieb ich ein Buch über transformationale Produkte, das zum Klassiker der digitalen Produktentwicklung wurde. Meine Beobachtung war damals: Die Digitalisierung transformiert nicht nur menschliches Verhalten und das, was Unternehmen tun – sie verändert auch, was Unternehmen sind. Ich verdichtete die Methoden erfolgreicher Silicon-Valley-Unternehmen mit meinen eigenen Erfahrungen in der Entwicklung digitaler Produkte für Startups und mit den digitalen Schnellbooten deutscher Konzerne seit Mitte der 1990er Jahre.

Meine Kernthesen waren:

These 1: Produkte werden zu Diensten. Die Grenze zwischen Produkt und Service bröckelt. Ein Auto ist kein Auto mehr – es ist Mobilität als Dienst. Eine Software ist keine Software mehr – sie ist ein sich ständig wandelndes Erlebnis.

These 2: Dienste werden zu Plattformen. Wer nur verkauft, verliert. Wer Ökosysteme baut, gewinnt. Amazon, Apple, Google – sie alle haben begriffen, dass der eigentliche Wert in der Steuerung von Netzwerkeffekten liegt, nicht im Einzelprodukt.
These 3: Die S-Kurve ist das Schicksal. Jede Technik durchläuft denselben Zyklus: langsamer Beginn, sprunghaftes Wachstum, Sättigung. Wer die Kurve lesen kann, sieht die Zukunft. Wer sie übersieht, wird von ihr überrollt.

These 4: Experiences schlagen Features. Gewohnheitsprägende Produkte – jene, die einen sich selbst verstärkenden Kreislauf aus Nutzung, Belohnung und Gewohnheit schaffen – dominieren die Feature-Inflation.

Das Buch enthielt auch ein umfangreiches Playbook für die Entwicklung digitaler Produkte als Blaupause für all jene, die sich erst ab Mitte der 2010er Jahre auf den Weg machten, ihr Kerngeschäft und ihre Produkte konsequent für eine digitale Welt auszurichten.

Eine Dekade später stellt sich die Frage, die kaum jemand laut ausspricht: Wann hat Ihr letztes Produkt-Release tatsächlich einen messbaren Unterschied gemacht? Nicht „Wir haben geliefert". Nicht „Das Projekt wurde abgeschlossen". Oder gar „Das Ticket ist geschlossen". Stattdessen: Wann hat ein digitales Produkt Ihres Unternehmens zuletzt das Nutzerverhalten verändert, Marktanteile gewonnen oder einen Geschäftsbereich transformiert?

Wenn Sie jetzt nachdenken müssen, sind Sie nicht allein.

Ein durchschnittliches Feature braucht in deutschen Unternehmen 14 bis 18 Monate von der ersten Idee bis zur Livestellung. Die Entwicklungskosten pro Feature sind in den letzten fünf Jahren deutlich gestiegen. Branchenstudien zeigen regelmäßig: Von zehn gestarteten Produktinitiativen erreichen nur zwei bis drei ihr ursprüngliches Ziel. Und Nutzungsdatenanalysen

(etwa von Pendo oder Amplitude) belegen, dass die Mehrheit der Features nur von einem Bruchteil der Nutzer regelmäßig genutzt wird.

Softwareentwicklung ist hart. Und statt alle Energie in die Kreation digitaler Produkte zu investieren, wurde die meiste Energie darauf gelegt, überhaupt „zu liefern" und — mit zunehmenden Schwierigkeiten im Geschäft — immer günstiger zu entwickeln. Das Ergebnis kann man heute im niedrigen Digitalisierungsgrad deutscher Unternehmen und Behörden betrauern.

VERSCHWUNDENER FLASCHENHALS

Digitale Produkte entstehen in crossfunktionalen Teams. Das war mein Mantra – und es stimmt immer noch auf gewisse Weise. Aber das war ein unzulässiger Shortcut. Ich unterschätzte massiv die Schwierigkeit, erprobte Vorgehensmodelle im Kleinen auf große Unternehmenseinheiten zu skalieren.

Der Flaschenhals war immer die Softwareentwicklung. Nicht die Ideen. Nicht das Design. Nicht das Marketing. Sondern die Fähigkeit, Ideen in funktionierenden Code zu übersetzen. Die Entwickler kamen nicht hinterher (wir werden später sehen: Es lag nicht an ihnen). Das Backlog wuchs schneller als die Lieferfähigkeit.

Was taten die Unternehmen? Sie versuchten, den Flaschenhals zu weiten.

Agile versprach: kleinere Pakete, kürzere Zyklen, weniger Verschwendung. Es half – ein bisschen. Aber der Engpass blieb.

SAFe versprach: Skalierung. Wenn ein Team nicht reicht, nehmen wir zehn. Koordination durch Frameworks. Das Ergebnis? Mehr Koordination, mehr Meetings, mehr Overhead – und dieselbe Time-to-Market. Der Flaschenhals wurde nicht geweitet. Er wurde mit Bürokratie ummantelt.

Offshoring versprach: mehr Entwickler, weniger Kosten. Das Ergebnis? Mehr Entwickler, mehr Kommunikationsprobleme, mehr Qualitätsprobleme – und: längere Time-to-Market. Der Flaschenhals wurde nicht geweitet. Er wurde verlagert.

Der Flaschenhals der Softwareentwicklung, der Engpass, um den wir zwanzig Jahre lang unsere gesamte Industrie organisiert hatten, existiert nicht mehr. In der alten Welt gab es einen natürlichen Filter: die Kosten der Umsetzung. Nicht jede Idee konnte verwirklicht werden, weil das Bauen teuer war. Der Filter war auch ein Schutz. Er schützte Organisationen (und einen selbst) vor schlechten Ideen und zwang zu priorisieren. Er sortierte schlechte Ideen aus, bevor sie zu schlechtem Code wurden.

Mit AI hat sich die Gleichung verändert. Was früher der Engpass war gibt es jetzt im Überfluss. Der neue Engpass ist die Fähigkeit zu entscheiden, *welcher* Code geschrieben werden soll. Der Intent – die präzise Formulierung dessen, was wir wollen und warum – wird zum kritischen Pfad. Ein einzelner Product Engineer mit den richtigen Tools liefert heute, wofür früher ein Team Wochen brauchte. Die Produktionskapazität ist explodiert.

Das ist keine inkrementelle Veränderung. Das ist ein Strukturbruch.

Dieses Buch handelt davon, was das bedeutet.

- Was bedeutet es für Teams, wenn der Engpass nicht mehr die Entwicklung ist?
- Was bedeutet es für Prozesse, wenn Releases in Stunden statt in Wochen möglich sind?
- Was bedeutet es für die Entwicklung digitaler Produkte, wenn sich die Randbedingungen dramatisch verändern?

Wenn Sie jahrelang gegen eine Wand gerannt sind und die Wand plötzlich verschwindet – dann stürzen Sie. Oder Sie lernen zu fliegen.

DAS JEVONS-PARADOXON

1865 beobachtete der britische Ökonom William Stanley Jevons einen seltsamen Widerspruch: James Watts' verbesserte Dampfmaschine senkte den Kohleverbrauch pro Energieeinheit um 75 Prozent. Die Erwartung: weniger Kohleverbrauch. Die Wirklichkeit: Der Verbrauch stieg sprunghaft. Die höhere Leistung machte neue Anwendungen erst wirtschaftlich und überkompensierte die Einsparungen beim Kohleverbrauch.

Das Muster wiederholt sich in jeder Technologiewelle:

Als Computing billig wurde, rechneten wir nicht einfach schneller. Die erste Generation von Computern war darauf optimiert, mathematische Berechnungen effizienter durchzuführen – etwa für Artillerietabellen, Versicherungsmathematik und wissenschaftliche Simulationen. Das war der First-Order-Effect: dieselben Aufgaben schneller erledigt. Aber dann entstanden Spreadsheets – nicht als schnellere Buchhaltung, sondern als völlig neues Werkzeug für „Was-wäre-wenn"-Analysen. Es entstanden relationale Datenbanken, die nicht einfach Karteikarten digitalisierten, sondern völlig neue Abfragemöglichkeiten schufen. Das waren Second-Order-Effects: neue Kategorien, die zuvor nicht existiert hatten.

Als Speicher billig wurde, speicherten wir nicht einfach nur mehr Dateien. Plötzlich wurde es wirtschaftlich sinnvoll, jede Suchanfrage zu speichern, jedes Nutzerverhalten zu protokollieren, jede Interaktion zu archivieren. Google wurde möglich – nicht als bessere Suchmaschine, sondern als Unternehmen, dessen Geschäftsmodell auf der Analyse von Milliarden Datenpunkten basiert. Facebook, das gesamte AdTech-Ökosystem, datengetriebene Geschäftsmodelle jeder Art – sie alle existieren nur, weil Speicher so billig wurde, dass das Sammeln und Analysieren von Daten im industriellen Maßstab wirtschaftlich sinnvoll wurde.

Als Bandbreite billig wurde, kommunizierten wir nicht einfach schneller. Es entstanden die Cloud als dominantes Computing-Paradigma, Software-as-a-Service als Geschäftsmodell und Streaming als Ersatz für Besitz. Netflix konnte nur existieren, weil Bandbreite billig genug wurde, dass das Streamen von HD-Video an Millionen von Haushalten gleichzeitig wirtschaftlich tragbar wurde.

Jetzt erleben wir das Jevons-Paradoxon der Softwareentwicklung:

AI macht die Codeerzeugung um Größenordnungen günstiger. Die naive Erwartung: Wir brauchen weniger Entwickler. Die Wirklichkeit: Wir starten viel mehr Projekte. Eine Flut an Software entsteht.

Die Folge klingt paradox:

- Es wird *leichter*, Software zu entwickeln.
- Es wird *schwerer*, sich zu differenzieren.

Wenn jeder bauen kann, wird das *Was* wichtiger als das *Wie*. Der Funke – das, was ein Produkt transformational macht – wird zum entscheidenden Faktor. Nicht die Fähigkeit, Code zu schreiben. Es geht um die Fähigkeit, den richtigen Code zu schreiben.

Das neue Paradoxon: Softwareentwicklung wird leichter – und zugleich *auf andere Weise schwer*.

DIE UNBEQUEME FRAGE

Es gibt eine Frage, die in den Diskussionen über AI und Softwareentwicklung systematisch vermieden wird: Wer verliert?

Jede Technologiewelle hat Gewinner und Verlierer. Die Gewinner sind sichtbar, sie stehen auf Konferenzbühnen. Die Verlierer verschwinden still.

Wer gewinnt? Dieselben, die bei jeder technologischen Disruption gewinnen: kleine, schnelle, autonome Teams mit klarer Ownership. Menschen, die Entscheidungen treffen, nicht Menschen, die Entscheidungen koordinieren.

Wer verliert? Die Koordinationsklasse. Menschen, deren Jobs nur existieren, um die Komplexität zu managen, die von anderen Menschen erzeugt wird, deren Jobs darin bestehen, Komplexität zu managen. Eine Bürokratie, die sich selbst rechtfertigt – Schicht um Schicht, bis niemand mehr weiß, wo die eigentliche Arbeit aufhört und die Verwaltung der Arbeit anfängt. AI entzieht dieser Pyramide gerade das Fundament.

WARUM DIE MEISTEN PROGNOSEN FALSCH SIND

Die Technologiebranche ist voller Prognosen zu AI und Produktivität. „Entwickler werden 40% produktiver". „Wir werden 30% weniger Entwickler brauchen". Diese Zahlen klingen präzise. Sie sind präzise falsch.

Sie sind falsch, weil sie First-Order-Denken anwenden: Wenn X effizienter wird, brauchen wir weniger X. Das Jevons-Paradoxon zeigt: Das Gegenteil passiert. Effizienz führt zu mehr Nutzung, nicht zu weniger. Aber sie sind auch aus einem tieferen Grund falsch: Sie messen das Falsche. „40% produktiver" – gemessen woran? Lines of Code? Features pro Sprint? Pull Requests pro Woche? All diese Metriken messen Output, nicht Outcome. Sie messen, wie viel produziert wird, nicht, ob das Produzierte einen Unterschied macht.

In einer Welt, in der Code billig ist, ist „mehr Code" keine Errungenschaft. Es ist eine Gefahr. Mehr Code bedeutet mehr Komplexität, mehr Wartung, mehr potenzielle Fehler. Die relevante Frage ist nicht „Wie viel

Code produzieren wir?“ sondern „Wie viel Wirkung erzielen wir pro Einheit Komplexität?“.

Diese Frage stellt fast niemand. Weil sie unbequem ist. Weil sie bedeutet, dass die Produktivität neu definiert werden muss. Weil sie bedeutet, dass viele der aktuellen Metriken überflüssig werden.

ZWEI PARADIGMENWECHSEL

Die Produktentwicklung steht vor zwei radikalen Umbrüchen – und beide ereignen sich exakt jetzt.

① Paradigmenwechsel 1: Agile ist in der Krise

Die agilen Methoden, an denen wir über zwei Dekaden gefeilt haben, liefern nicht mehr. Agil zu denken ist nicht verfehlt. Aber was Unternehmen daraus machten, bewahrte vom ursprünglichen Geist nur noch die Begriffe.

Das nächste Kapitel seziert diese Krise. Hier genügt die Diagnose: Die starken Produktunternehmen fragen nicht: „Wie skalieren wir Agilität?“ Sie fragen: „Wie bleiben wir klein genug, um keine skalierte Agilität zu brauchen?“

② Paradigmenwechsel 2: Intent ist der neue Code

Der zweite Bruch betrifft die Softwareentwicklung selbst. Wenn AI-Werkzeuge den kompletten Routine-Code erzeugen und die Umsetzung vom Engpass zur Massenware wird – was bleibt als menschliche Kernkompetenz? Die Antwort: der Intent.

Ein Einwand liegt auf der Hand: War Intent nicht schon immer wichtig? Jeder gute Produktmanager, jede gute Unternehmerin musste schon immer präzise formulieren, was erreicht werden soll.

Der Einwand ist richtig. Und geht doch am Punkt vorbei.

Der Unterschied ist nicht, dass Intent *neu* wäre. Der Unterschied ist, dass Intent jetzt direkt produktiv wird. Früher war Intent der Anfang einer langen Kette: Intent wurde zur Spezifikation, Spezifikation zur Architektur, Architektur zu Code, Code zu Test. Jeder Übersetzungsschritt dauerte Wochen. Jeder Schritt verlor Wissen. Am Ende stand Code, der mit dem ursprünglichen Intent oft nur noch entfernt verwandt war.

Heute kann Intent, wenn er scharf genug gefasst ist, *unmittelbar* in laufenden Code übersetzt werden. Die Kette zurrt zusammen. Was früher zehn Schritte brauchte, braucht jetzt nur einen. Was früher Wochen dauerte, dauert Stunden.

Das ändert die Wirtschaftlichkeit von Grund auf. Früher war Intent billig und Code teuer. Jetzt ist Code billig und Intent teuer. Wenn Output nichts mehr kostet, wird der Input zum knappen Gut.

INPUT WIRD WICHTIGER ALS OUTPUT

In einer Welt, in der der Engpass der Softwareentwicklung nicht mehr existiert, wird es eine kambrische Explosion digitaler Produkte geben. Die Aspekte, die erfolgreiche digitale Produkte ausmachen, bleiben die gleichen. Doch wie diese Produkte gebaut werden, verändert sich von Grund auf. Wie genau, weiß noch niemand. In den letzten drei Jahren seit dem Siegeszug der generischen AI-Modelle haben wir aber Leitplanken entwickelt, die uns sehr geholfen haben, in dieser neuen Welt erfolgreiche Produkte zu bauen. Es ist weniger ein Framework. Eher eine Haltung. Der Dreiklang lautet Soul – System – Speed:

Soul ist die Fähigkeit, über digitale Produkte zu denken und zu fühlen – den Funken, der ein Produkt von einer Commodity unterscheidet. Die notwendigen Fähigkeiten: Judgment, Cultural Fluency, Meaning-Setting Narratives.

System sichert Produktionsqualität durch Infrastruktur – Architecture, Testability, Observability, Security, Performance, Reliability –, was den Unterschied zwischen Prototyp und Produktion ausmacht.

Speed entsteht durch Pipeline-Verdichtung – nicht durch bessere Prozesse. Die Pipeline wird radikal reduziert: Perceive → Prompt → Produce → Pitch:

- **Perceive**: Einen Insight wahrnehmen, bevor man formuliert
- **Prompt**: Die AI mit klarem Intent beauftragen
- **Produce**: Direkt produzieren, keine Prototypen
- **Pitch**: Pitchen, was man gebaut hat – nicht was man bauen will

Speed entsteht nicht durch bessere Prozesse. Sie entsteht durch das Streichen von Prozessen.

NEUBAUTEN STATT PFLASTER

Die meisten Organisationen reagieren auf diese Brüche, als handele es sich um kleine Veränderungen. Sie kleben ein AI-Pflaster auf den alten Prozess. Sie schicken Mitarbeiter ins AI-Bootcamp. Sie streuen AI-Kompetenz in Stellenanzeigen.

Die Erschütterungen, die wir erleben, lassen sich nicht durch kosmetische Anpassungen abfedern. Sie gehen an die Substanz, ja, und verlangen oft Neubauten. Die gefährlichste Illusion ist die Illusion der Kontinuität: dass man weitermachen kann wie bisher, nur etwas schneller, etwas digitaler, etwas AI-gestützter. Diese Illusion ist bequem – führt aber aufs Abstellgleis. Nicht alle Rollen, die es heute gibt, werden es morgen noch geben. Nicht alle Karrierewege, die gestern sicher waren, sind es morgen noch.

Das ist kein Grund zur Panik. Es ist ein Grund zum Handeln. Wer heute anfängt, Intent-Formulierung, Nutzerverständnis und strategisches Denken zu entwickeln, wird morgen gefragt sein. Wer wartet, wird von den Ereignissen überrollt.

WAS DIESES BUCH LIEFERT

2017 beschrieb ich die Umwälzung durch digitale Produkte. Heute beschreibe ich die Umwälzung der Produktentwicklung selbst. Ein neues Betriebsmodell für eine neue Wirklichkeit:

Soul – System – Speed. Der Dreiklang als Antwort auf das Jevons-Paradoxon: Soul liefert den Funken, System die Zuverlässigkeit, Speed die Beschleunigung.

Die Intent-to-Production Pipeline. Der klassische Genehmigungsablauf – ob Wasserfall oder SAFe-Bürokratie – ist tot. Er verwechselt Dokumentation mit Fortschritt. Die Pipeline ersetzt ihn: Perceive, Prompt, Produce, Pitch. Speed in der Praxis.

Product Teams. Keine parallel skalierten Teams mehr, keine Abhängigkeiten. Keine Übergaben. Keine Silos. Kleine, selbständige Einheiten, die Verantwortung von Anfang bis Ende tragen.

Outcome-Governance. Schluss mit Feature-Zählen. Schluss mit Velocity-Messung. Schluss mit Output-Theater. Steuerung, die nur eine Frage stellt: Haben wir für den Nutzer und das Business etwas erreicht, das messbar ist?

Dieses Buch beschreibt, was jetzt zu tun ist:

- **Teil I: Der neue Kontext** kartiert das veränderte Gelände durch AI. Kapitel 2 zerlegt, warum Agilität in der Krise steckt. Kapitel 3 untersucht die Umwälzung in der Softwareentwicklung.

- **Teil II: Das neue Betriebsmodell** liefert die Leitplanken für diese neue Wirklichkeit. Kapitel 4 führt Soul – System – Speed als neue Prinzipien ein. Kapitel 5 beschreibt die Intent-to-Production-Pipeline.

- **Teil III: Umsetzung** macht es greifbar. Kapitel 6 führt die Product Teams ein. Kapitel 7 markiert die Risiken und Fehlmuster – den Weg, auf dem diese Umwälzung scheitern kann. Der Epilog versucht einen einen Ausblick auf das zu geben, was kommt, und erweitert den Horizont noch einmal.

WAS BLEIBT

Nicht alles ändert sich. Manche Prinzipien sind zeitlos:

Nutzerzentrierung. Produkte gehören dem Nutzer. Nicht der produzierenden Organisation. Nicht den Prozesse. Das war 2016 richtig. Das ist 2026 richtig. Das wird 2036 richtig sein.

Versuchen und Lernen. Niemand weiß vorher, was funktioniert. Die einzige Methode, es herauszufinden: ausprobieren, messen, lernen. Das Tempo der Versuche ändert sich jetzt. Der Grundsatz nicht.

Menschen vor Prozessen. Menschen bauen Produkte. Nicht Regelwerke. Das beste Framework in den Händen eines unmotivierten Teams erzeugt nur Rauschen.

Outcome vor Output. Was zählt, ist nicht, was wir liefern. Es zählt, was wir erreichen. Features zu zählen ist leicht. Wirkung zu messen ist schwer. Aber nur Wirkung zählt.

Diese Grundsätze bilden das Fundament, auf dem das neue Betriebsmodell steht. Sie sind der rote Faden zwischen 2017 und heute.

FÜR WEN IST DIESES BUCH?

Dieses Buch richtet sich an alle, die noch was vorhaben:

- **Gründerinnen und Geschäftsführer**, die verstehen wollen, wie sich ihre Produktorganisation wandeln muss – oder überflüssig wird
- **Produktverantwortliche**, die begreifen wollen, was AI mit ihrer Rolle macht, bevor die Rolle sie überholt

- **Tech-Leads**, die neue Formen für ihre Teams suchen, weil die alten nicht mehr funktionieren
- **Beratungen und Agenturen**, die ihre Auftraggeber in diese neue Wirklichkeit begleiten wollen, statt sie mit Regelwerken von gestern einzuschläfern

Wenn Sie nur einen Gedanken aus diesem Buch mitnehmen, dann diesen:

Kernbotschaft: Der Flaschenhals in der Entwicklung digitaler Produkte – das Schreiben von Code – ist verschwunden. Agentische AI hat ihn beseitigt. Aber Engpässe verschwinden nicht; sie verschieben sich. Der neue Engpass ist Intent: zu wissen, *was* gebaut werden soll und *warum*. Die Erfahrung lehrt, dass diese Verschiebung zu Second-Order-Effekten führt, die wesentlich größer sind als die Effizienzgewinne, die man zu Beginn prognostiziert hat.

ZUSAMMENFASSUNG: WARUM JETZT ALLES ANDERS IST

Sechs Erkenntnisse:

1. **Zwei Wendepunkte:** ChatGPT (November 2022) zeigte eine AI, die das Denken simulieren konnte. Die neue Generation agentischer AI-Coding-Tools kann bauen. Der eigentliche Strukturbruch.

2. **Das Jevons-Paradoxon:** Bauen wird leichter – sich voneinander zu unterscheiden sehr viel schwerer.

3. **Die zwei Paradigmenwechsel:** Agile steckt in der Krise – SAFe löst das falsche Problem. Intent ist der neue Code – die Übersetzungskette von Idee zu Code schrumpft auf eine Konversation.

4. **Soul – System – Speed:** Der Dreiklang als Antwort.

5. **Die 4P-Pipeline:** Perceive → Prompt → Produce → Pitch.

6. **Was bleibt:** Nutzerzentrierung, Versuchen und Lernen, Menschen vor Prozessen, Outcome vor Output. Die Technologie ändert sich. Die Grundsätze nicht.

Im folgenden Kapitel: Warum Agile in der Krise steckt – und warum die Antwort anders lautet als „mehr Agile".

Los geht's.

Das Update des Beststellers
„Transformationale Produkte“

CODE CRASH

VOM ENDE DER IT, WIE WIR SIE KENNEN, UND WARUM JETZT ENDLICH ALLES GUT WIRD MIT DER DIGITALISIERUNG.

Matthias Schrader

ÜBER DEN AUTOR

Matthias Schrader gehört zu den digitalen Pionieren in Deutschland. Mitte der 1990er-Jahre gründete er SinnerSchrader und entwickelte E-Commerce-Lösungen für Start-ups, deren Produkte in kürzester Zeit börsenreif wurden.

1999 ging SinnerSchrader selbst an die Börse und gehörte zu den wenigen Unternehmen, die den Neuen Markt nicht nur überlebten, sondern sogar aus dieser Zeit gestärkt hervorgingen. 2006 gründete Matthias Schrader die NEXT Conference, die sich innerhalb weniger Jahre als führende Konferenz für die digitale Transformation in Deutschland etablierte. In der Folge unterstützte SinnerSchrader mit über 500 Beratern, Designern und Software-Entwicklern hauptsächlich DAX-Konzerne bei der Entwicklung digitaler Produkte.

2017 übernahm die weltweite Management- und Technologieberatung Accenture das Unternehmen für einen dreistelligen Millionenbetrag. In der Folge war der Autor bis Ende 2022 für das Geschäft von Accenture Interactive/Song in Deutschland, Österreich und der Schweiz verantwortlich. 2024 gründete er gemeinsam mit zahlreichen Weggefährten die AI-native Beratung und Agentur OH-SO Digital in Hamburg, Berlin, München und Prag.

Inhaltsverzeichnis

User Manual

Dieser Text ist auf Denglisch verfasst. Im digitalen Kontext haben wir grundsätzlich mit vielen englischsprachigen Fachbegriffen zu tun und eine strikte Eindeutschung wäre für viele Teile des Fachpublikums unverdaulich gewesen. Wir haben uns bemüht, die zentralen Termini in einem Glossar zusammenzustellen. Bei der ersten Verwendung des Begriffs erscheint er unterstrichen.

Das vorliegende Buch macht an verschiedenen Stellen Rückgriff auf das Buch „Transformationale Produkte“, welches der Autor im Jahr 2017 veröffentlicht hat und zum Standardwerk für die Entwicklung digitaler Produkte im deutschsprachigen Raum geworden ist. Im Folgenden werden jeweils die wesentlichen Aspekte von damals skizziert, soweit sie für das Gesamtverständnis notwendig sind. Das Vorgängerwerk zu kennen, ist daher keine Voraussetzung.

Zudem gibt es in einigen Abschnitten technische Deep Dives. Sie dienen nur der Vertiefung und sind für das Gesamtverständnis ebenfalls nicht zwingend. Überblättern Sie diese ruhig.

Um das Werk aktuell zu halten sind, sämtliche Arbeitsmaterialien und Checklisten digital unter https://XXXXX.com abrufbar.

PROLOGUE

November 2022

Ich erinnere mich an jenen kalten Abend im November. Es war spät, der Arbeitstag längst zerfasert in E-Mails, Slack-Pings und ziellosem Scrollen. Ein Freund schickte einen Link, lakonisch: „Hast du das gesehen?“ Die URL führte zu chat.openai.com.

Chatbots waren so 90er, trotzdem klickte ich mich zur Website und tippte skeptisch: „Erkläre Blockchain in drei Sätzen.“ Die Antwort erschien. Präzise. Strukturiert. Irritierend kompetent.

Ich erhöhte den Einsatz. „Beurteile, wie gut die drei wichtigsten Thesen meines Buches ›Transformationale Produkte‹ von 2017 gealtert sind.“ Das Ergebnis? Ich hätte es in dem Moment nicht besser artikulieren können. Ich machte verwirrt weiter, obwohl ich das Ergebnis bereits ahnte: „Ein Haiku über Artificial Intelligence.“ Es skandierte korrekt. Na klar.

Na klar? Davor hätte ich jeden ausgelacht, der so etwas prophezeit hätte. Jetzt starrte ich auf den Bildschirm wie jemand, der gerade bemerkt hatte, dass hinter dem leuchtenden Panel vor mir eine außerirdische Intelligenz erwacht war. Es fühlte sich an wie ein Riss in der Realität.

Dieselbe Szene wiederholte sich in den folgenden Tagen und Wochen millionenfach. Büros, Küchen, U-Bahnen. Menschen sahen auf ihre Screens und spürten: Ihre Vorstellungen davon, was Computer können, lösten sich gerade auf.

Fünf Tage nach dem Start erreichte ChatGPT eine Million Nutzer. Zwei Monate später durchbrach OpenAI die 100-Millionen-Schallmauer. Keine Technologie und kein Medium verbreiteten sich in der Geschichte schneller.

Das war der ChatGPT-Moment. Der Augenblick, in dem etwas vollkommen Unerwartetes die Labortür aufschlägt und die Welt in neues Licht taucht.

Aber ChatGPT sollte erst der Anfang sein.

Drei Jahre später

Wieder ein später Abend. Wieder saß ich vor dem Bildschirm. Diesmal war es kein Link von einem Freund – ich hatte Claude Code mit dem neuen Opus-4.5-Modell upgedatet, das Anthropic gerade veröffentlicht hatte.

Ich war immer noch in der Desillusionphase des Vibe Codings gefangen. Das ganze Jahr hatten wir die neuen AI Coding Tools bereits im Einsatz. Es begann mit kleinen Prototypen einzelner Funktionen und wuchs bis hin zu MVPs, die sogar in die Produktion gingen. Aber die Arbeit mit den Tools war langsam, fehleranfällig und glücksspielhaft. Die Entropie wuchs mit der Zeit und die Codebasis war irgendwann kaum noch zu managen.

Es schien eine Sackgasse zu sein.

Ich war müde. Ich öffnete ein Terminalfenster und startete Claude Code. Mein Prompt: unsere LLM-Analytics-Plattform mit einer sechsstelligen Anzahl an Programmzeilen komplett aufzuräumen. Robuster, sicherer – ohne das Nutzerverhalten zu ändern. Ein lustlos-aggressiver Brute-Force-Prompt. In den Monaten davor hätte er jede Codebasis zuverlässig geschreddert.

Am nächsten Morgen hatte ich meine abendliche Aktion schon fast vergessen, startete die Analytics-Plattform, die sich an diesem Morgen ein klein wenig flüssiger anfühlte als noch am Tag zuvor. Erst nach einer Stunde fiel mein Blick auf das von vielen Fenstern verdeckte Terminalprogramm. Claude Code meldete, dass es in der Nacht die Codebasis um 40 Prozent reduziert, große Module sauber aufgeräumt und neu strukturiert, Dutzende kleinerer und mittlerer Security-Themen identifiziert und gefixt sowie die Dokumentation upgedatet hatte. Natürlich hätte es auch alles getestet und, da alles fehlerfrei lief, bereits deployed.

Boom. Ich hatte den ganzen Morgen bereits an einer völlig neuen Codebasis gearbeitet.

Ich war ehrlich geschockt. Was passiert hier gerade? Ich dachte an jenen Novemberabend 2022. Damals hatte ich mit den LLMs eine künstliche Intelligenz erlebt, die *denken* mehr oder weniger gut simulieren konnte. Jetzt erlebte ich eine, die *liefern* konnte. Das war keine Simulation mehr. Der Code war tatsächlich neu geschrieben und lief einwandfrei. Entwicklerwochen an Programmierleistung wurden in einem einzigen Prompt verdichtet, der lediglich ein paar Millionen Token an Compute-Leistung triggerte.

Das war der Moment, in dem ich verstand: ChatGPT war erst der Pilot. Jetzt beginnt die eigentliche Serie. Das Skript, das zeigt, wie wir digitale Produkte entwickeln, muss neu verfasst werden.

Davon handelt dieses Buch.

Second Order Effects
Warum dieser Moment anders ist

“Almost all new code written at OpenAI today is written by Codex. We’re at the point where AI writes the code, and humans verify it works.“

— Sam Altman, OpenAI DevDay (Oktober 2025)

Technologische Brüche sind nichts Neues. Das iPhone 2007. Das World Wide Web Mitte der Neunziger. Personal Computer in den Achtzigern. Jeder dieser Momente veränderte Branchen und ließ neue Industrien aus dem Nichts entstehen. Jede Generation veränderte, wie Software entwickelt wurde: Hochsprachen ersetzten Assembler. Objektorientierung ersetzte die prozedurale Programmierung. Agile ersetzte den Wasserfall. DevOps automatisierte den Betrieb. Low-Code versprach einfache Anwendungen für alle.

Aber all diese Umbrüche betrafen Werkzeuge, Methoden und Abstraktionsebenen. Die geistige Kärrnerarbeit – ein Problem vollständig zu durchdringen und in Code zu übersetzen – blieb die Aufgabe des Menschen an der Tastatur. Der Aufwand, Software zu schreiben, hat sich in den letzten Jahrzehnten nicht verringert.

Das ändert sich gerade radikal, weil AI diese geistige Tätigkeit selbst übernimmt. Die Maschine übersetzt jetzt Probleme in Code. Nicht immer fehlerfrei. Nicht jederzeit zuverlässig. Aber gut genug und jeden Monat besser, um die Frage zu stellen: wie sieht eine Welt aus, in der AI das Problem der Software-Entwicklung gelöst hat?

STILLE REVOLUTION

Mit den agentischen AI-Coding-Tools wie Claude Code werden Large Language Modelle (LLMs) zu Produktivitätsmonstern. Sie generieren mittlerweile Code, der direkt einsatzbereit ist.

Diese Erfahrung wiederholt sich gerade rund um den Globus. Entwickler berichten, dass sie innerhalb weniger Wochen von manuellem Coding auf 80 Prozent agentischem AI-Coding umgestiegen sind. Die größte Veränderung ihrer Arbeitsweise, die sie je erlebt haben.

Die breite Öffentlichkeit ahnt von dieser Revolution wenig. Wahr ist aber auch: Die Kluft zwischen dem, was in den AI-zentrierten Softwareunternehmen und den „normalen" IT-Abteilungen passiert, wächst täglich.

Die Ironie: Wer programmiert, programmiert jetzt auf Englisch. Oder eben Deutsch. Nicht in Python, nicht in JavaScript – in natürlicher Sprache. Das kratzt am Ego jener Profession, die bisher programmiert hat. Aber der Hebel ist zu groß, um ihn zu ignorieren.

Die Antwort darauf, wie eine Welt aussieht, in der das Problem der Softwareentwicklung gelöst ist, lautet: Es wird eine radikal neue Welt sein. Dazu muss ich einmal ausholen, warum Softwareentwicklung bisher überhaupt ein Problem war.

Digitale Produkte

2017 schrieb ich ein Buch über transformationale Produkte, das zum Klassiker der digitalen Produktentwicklung wurde. Meine Beobachtung war damals: Die Digitalisierung transformiert nicht nur menschliches Verhalten und das, was Unternehmen tun – sie verändert auch, was Unternehmen sind. Ich verdichtete die Methoden erfolgreicher Silicon-Valley-Unternehmen mit meinen eigenen Erfahrungen in der Entwicklung digitaler Produkte für Startups und mit den digitalen Schnellbooten deutscher Konzerne seit Mitte der 1990er Jahre.

Meine Kernthesen waren:

These 1: Produkte werden zu Diensten. Die Grenze zwischen Produkt und Service bröckelt. Ein Auto ist kein Auto mehr – es ist Mobilität als Dienst. Eine Software ist keine Software mehr – sie ist ein sich ständig wandelndes Erlebnis.

These 2: Dienste werden zu Plattformen. Wer nur verkauft, verliert. Wer Ökosysteme baut, gewinnt. Amazon, Apple, Google – sie alle haben begriffen, dass der eigentliche Wert in der Steuerung von Netzwerkeffekten liegt, nicht im Einzelprodukt.
These 3: Die S-Kurve ist das Schicksal. Jede Technik durchläuft denselben Zyklus: langsamer Beginn, sprunghaftes Wachstum, Sättigung. Wer die Kurve lesen kann, sieht die Zukunft. Wer sie übersieht, wird von ihr überrollt.

These 4: Experiences schlagen Features. Gewohnheitsprägende Produkte – jene, die einen sich selbst verstärkenden Kreislauf aus Nutzung, Belohnung und Gewohnheit schaffen – dominieren die Feature-Inflation.

Das Buch enthielt auch ein umfangreiches Playbook für die Entwicklung digitaler Produkte als Blaupause für all jene, die sich erst ab Mitte der 2010er Jahre auf den Weg machten, ihr Kerngeschäft und ihre Produkte konsequent für eine digitale Welt auszurichten.

Eine Dekade später stellt sich die Frage, die kaum jemand laut ausspricht: Wann hat Ihr letztes Produkt-Release tatsächlich einen messbaren Unterschied gemacht? Nicht „Wir haben geliefert". Nicht „Das Projekt wurde abgeschlossen". Oder gar „Das Ticket ist geschlossen". Stattdessen: Wann hat ein digitales Produkt Ihres Unternehmens zuletzt das Nutzerverhalten verändert, Marktanteile gewonnen oder einen Geschäftsbereich transformiert?

Wenn Sie jetzt nachdenken müssen, sind Sie nicht allein.

Ein durchschnittliches Feature braucht in deutschen Unternehmen 14 bis 18 Monate von der ersten Idee bis zur Livestellung. Die Entwicklungskosten pro Feature sind in den letzten fünf Jahren deutlich gestiegen. Branchenstudien zeigen regelmäßig: Von zehn gestarteten Produktinitiativen erreichen nur zwei bis drei ihr ursprüngliches Ziel. Und Nutzungsdatenanalysen

(etwa von Pendo oder Amplitude) belegen, dass die Mehrheit der Features nur von einem Bruchteil der Nutzer regelmäßig genutzt wird.

Softwareentwicklung ist hart. Und statt alle Energie in die Kreation digitaler Produkte zu investieren, wurde die meiste Energie darauf gelegt, überhaupt „zu liefern" und — mit zunehmenden Schwierigkeiten im Geschäft — immer günstiger zu entwickeln. Das Ergebnis kann man heute im niedrigen Digitalisierungsgrad deutscher Unternehmen und Behörden betrauern.

VERSCHWUNDENER FLASCHENHALS

Digitale Produkte entstehen in crossfunktionalen Teams. Das war mein Mantra – und es stimmt immer noch auf gewisse Weise. Aber das war ein unzulässiger Shortcut. Ich unterschätzte massiv die Schwierigkeit, erprobte Vorgehensmodelle im Kleinen auf große Unternehmenseinheiten zu skalieren.

Der Flaschenhals war immer die Softwareentwicklung. Nicht die Ideen. Nicht das Design. Nicht das Marketing. Sondern die Fähigkeit, Ideen in funktionierenden Code zu übersetzen. Die Entwickler kamen nicht hinterher (wir werden später sehen: Es lag nicht an ihnen). Das Backlog wuchs schneller als die Lieferfähigkeit.

Was taten die Unternehmen? Sie versuchten, den Flaschenhals zu weiten.

Agile versprach: kleinere Pakete, kürzere Zyklen, weniger Verschwendung. Es half – ein bisschen. Aber der Engpass blieb.

SAFe versprach: Skalierung. Wenn ein Team nicht reicht, nehmen wir zehn. Koordination durch Frameworks. Das Ergebnis? Mehr Koordination, mehr Meetings, mehr Overhead – und dieselbe Time-to-Market. Der Flaschenhals wurde nicht geweitet. Er wurde mit Bürokratie ummantelt.

Offshoring versprach: mehr Entwickler, weniger Kosten. Das Ergebnis? Mehr Entwickler, mehr Kommunikationsprobleme, mehr Qualitätsprobleme – und: längere Time-to-Market. Der Flaschenhals wurde nicht geweitet. Er wurde verlagert.

Der Flaschenhals der Softwareentwicklung, der Engpass, um den wir zwanzig Jahre lang unsere gesamte Industrie organisiert hatten, existiert nicht mehr. In der alten Welt gab es einen natürlichen Filter: die Kosten der Umsetzung. Nicht jede Idee konnte verwirklicht werden, weil das Bauen teuer war. Der Filter war auch ein Schutz. Er schützte Organisationen (und einen selbst) vor schlechten Ideen und zwang zu priorisieren. Er sortierte schlechte Ideen aus, bevor sie zu schlechtem Code wurden.

Mit AI hat sich die Gleichung verändert. Was früher der Engpass war gibt es jetzt im Überfluss. Der neue Engpass ist die Fähigkeit zu entscheiden, *welcher* Code geschrieben werden soll. Der Intent – die präzise Formulierung dessen, was wir wollen und warum – wird zum kritischen Pfad. Ein einzelner Product Engineer mit den richtigen Tools liefert heute, wofür früher ein Team Wochen brauchte. Die Produktionskapazität ist explodiert.

Das ist keine inkrementelle Veränderung. Das ist ein Strukturbruch.

Dieses Buch handelt davon, was das bedeutet.

- Was bedeutet es für Teams, wenn der Engpass nicht mehr die Entwicklung ist?
- Was bedeutet es für Prozesse, wenn Releases in Stunden statt in Wochen möglich sind?
- Was bedeutet es für die Entwicklung digitaler Produkte, wenn sich die Randbedingungen dramatisch verändern?

Wenn Sie jahrelang gegen eine Wand gerannt sind und die Wand plötzlich verschwindet – dann stürzen Sie. Oder Sie lernen zu fliegen.

DAS JEVONS-PARADOXON

1865 beobachtete der britische Ökonom William Stanley Jevons einen seltsamen Widerspruch: James Watts' verbesserte Dampfmaschine senkte den Kohleverbrauch pro Energieeinheit um 75 Prozent. Die Erwartung: weniger Kohleverbrauch. Die Wirklichkeit: Der Verbrauch stieg sprunghaft. Die höhere Leistung machte neue Anwendungen erst wirtschaftlich und überkompensierte die Einsparungen beim Kohleverbrauch.

Das Muster wiederholt sich in jeder Technologiewelle:

Als Computing billig wurde, rechneten wir nicht einfach schneller. Die erste Generation von Computern war darauf optimiert, mathematische Berechnungen effizienter durchzuführen – etwa für Artillerietabellen, Versicherungsmathematik und wissenschaftliche Simulationen. Das war der First-Order-Effect: dieselben Aufgaben schneller erledigt. Aber dann entstanden Spreadsheets – nicht als schnellere Buchhaltung, sondern als völlig neues Werkzeug für „Was-wäre-wenn"-Analysen. Es entstanden relationale Datenbanken, die nicht einfach Karteikarten digitalisierten, sondern völlig neue Abfragemöglichkeiten schufen. Das waren Second-Order-Effects: neue Kategorien, die zuvor nicht existiert hatten.

Als Speicher billig wurde, speicherten wir nicht einfach nur mehr Dateien. Plötzlich wurde es wirtschaftlich sinnvoll, jede Suchanfrage zu speichern, jedes Nutzerverhalten zu protokollieren, jede Interaktion zu archivieren. Google wurde möglich – nicht als bessere Suchmaschine, sondern als Unternehmen, dessen Geschäftsmodell auf der Analyse von Milliarden Datenpunkten basiert. Facebook, das gesamte AdTech-Ökosystem, datengetriebene Geschäftsmodelle jeder Art – sie alle existieren nur, weil Speicher so billig wurde, dass das Sammeln und Analysieren von Daten im industriellen Maßstab wirtschaftlich sinnvoll wurde.

Als Bandbreite billig wurde, kommunizierten wir nicht einfach schneller. Es entstanden die Cloud als dominantes Computing-Paradigma, Software-as-a-Service als Geschäftsmodell und Streaming als Ersatz für Besitz. Netflix konnte nur existieren, weil Bandbreite billig genug wurde, dass das Streamen von HD-Video an Millionen von Haushalten gleichzeitig wirtschaftlich tragbar wurde.

Jetzt erleben wir das Jevons-Paradoxon der Softwareentwicklung:

AI macht die Codeerzeugung um Größenordnungen günstiger. Die naive Erwartung: Wir brauchen weniger Entwickler. Die Wirklichkeit: Wir starten viel mehr Projekte. Eine Flut an Software entsteht.

Die Folge klingt paradox:

- Es wird *leichter*, Software zu entwickeln.
- Es wird *schwerer*, sich zu differenzieren.

Wenn jeder bauen kann, wird das *Was* wichtiger als das *Wie*. Der Funke – das, was ein Produkt transformational macht – wird zum entscheidenden Faktor. Nicht die Fähigkeit, Code zu schreiben. Es geht um die Fähigkeit, den richtigen Code zu schreiben.

Das neue Paradoxon: Softwareentwicklung wird leichter – und zugleich *auf andere Weise schwer*.

DIE UNBEQUEME FRAGE

Es gibt eine Frage, die in den Diskussionen über AI und Softwareentwicklung systematisch vermieden wird: Wer verliert?

Jede Technologiewelle hat Gewinner und Verlierer. Die Gewinner sind sichtbar, sie stehen auf Konferenzbühnen. Die Verlierer verschwinden still.

Wer gewinnt? Dieselben, die bei jeder technologischen Disruption gewinnen: kleine, schnelle, autonome Teams mit klarer Ownership. Menschen, die Entscheidungen treffen, nicht Menschen, die Entscheidungen koordinieren.

Wer verliert? Die Koordinationsklasse. Menschen, deren Jobs nur existieren, um die Komplexität zu managen, die von anderen Menschen erzeugt wird, deren Jobs darin bestehen, Komplexität zu managen. Eine Bürokratie, die sich selbst rechtfertigt – Schicht um Schicht, bis niemand mehr weiß, wo die eigentliche Arbeit aufhört und die Verwaltung der Arbeit anfängt. AI entzieht dieser Pyramide gerade das Fundament.

WARUM DIE MEISTEN PROGNOSEN FALSCH SIND

Die Technologiebranche ist voller Prognosen zu AI und Produktivität. „Entwickler werden 40% produktiver". „Wir werden 30% weniger Entwickler brauchen". Diese Zahlen klingen präzise. Sie sind präzise falsch.

Sie sind falsch, weil sie First-Order-Denken anwenden: Wenn X effizienter wird, brauchen wir weniger X. Das Jevons-Paradoxon zeigt: Das Gegenteil passiert. Effizienz führt zu mehr Nutzung, nicht zu weniger. Aber sie sind auch aus einem tieferen Grund falsch: Sie messen das Falsche. „40% produktiver" – gemessen woran? Lines of Code? Features pro Sprint? Pull Requests pro Woche? All diese Metriken messen Output, nicht Outcome. Sie messen, wie viel produziert wird, nicht, ob das Produzierte einen Unterschied macht.

In einer Welt, in der Code billig ist, ist „mehr Code" keine Errungenschaft. Es ist eine Gefahr. Mehr Code bedeutet mehr Komplexität, mehr Wartung, mehr potenzielle Fehler. Die relevante Frage ist nicht „Wie viel

Code produzieren wir?“ sondern „Wie viel Wirkung erzielen wir pro Einheit Komplexität?“.

Diese Frage stellt fast niemand. Weil sie unbequem ist. Weil sie bedeutet, dass die Produktivität neu definiert werden muss. Weil sie bedeutet, dass viele der aktuellen Metriken überflüssig werden.

ZWEI PARADIGMENWECHSEL

Die Produktentwicklung steht vor zwei radikalen Umbrüchen – und beide ereignen sich exakt jetzt.

① Paradigmenwechsel 1: Agile ist in der Krise

Die agilen Methoden, an denen wir über zwei Dekaden gefeilt haben, liefern nicht mehr. Agil zu denken ist nicht verfehlt. Aber was Unternehmen daraus machten, bewahrte vom ursprünglichen Geist nur noch die Begriffe.

Das nächste Kapitel seziert diese Krise. Hier genügt die Diagnose: Die starken Produktunternehmen fragen nicht: „Wie skalieren wir Agilität?“ Sie fragen: „Wie bleiben wir klein genug, um keine skalierte Agilität zu brauchen?“

② Paradigmenwechsel 2: Intent ist der neue Code

Der zweite Bruch betrifft die Softwareentwicklung selbst. Wenn AI-Werkzeuge den kompletten Routine-Code erzeugen und die Umsetzung vom Engpass zur Massenware wird – was bleibt als menschliche Kernkompetenz? Die Antwort: der Intent.

Ein Einwand liegt auf der Hand: War Intent nicht schon immer wichtig? Jeder gute Produktmanager, jede gute Unternehmerin musste schon immer präzise formulieren, was erreicht werden soll.

Der Einwand ist richtig. Und geht doch am Punkt vorbei.

Der Unterschied ist nicht, dass Intent *neu* wäre. Der Unterschied ist, dass Intent jetzt direkt produktiv wird. Früher war Intent der Anfang einer langen Kette: Intent wurde zur Spezifikation, Spezifikation zur Architektur, Architektur zu Code, Code zu Test. Jeder Übersetzungsschritt dauerte Wochen. Jeder Schritt verlor Wissen. Am Ende stand Code, der mit dem ursprünglichen Intent oft nur noch entfernt verwandt war.

Heute kann Intent, wenn er scharf genug gefasst ist, *unmittelbar* in laufenden Code übersetzt werden. Die Kette zurrt zusammen. Was früher zehn Schritte brauchte, braucht jetzt nur einen. Was früher Wochen dauerte, dauert Stunden.

Das ändert die Wirtschaftlichkeit von Grund auf. Früher war Intent billig und Code teuer. Jetzt ist Code billig und Intent teuer. Wenn Output nichts mehr kostet, wird der Input zum knappen Gut.

INPUT WIRD WICHTIGER ALS OUTPUT

In einer Welt, in der der Engpass der Softwareentwicklung nicht mehr existiert, wird es eine kambrische Explosion digitaler Produkte geben. Die Aspekte, die erfolgreiche digitale Produkte ausmachen, bleiben die gleichen. Doch wie diese Produkte gebaut werden, verändert sich von Grund auf. Wie genau, weiß noch niemand. In den letzten drei Jahren seit dem Siegeszug der generischen AI-Modelle haben wir aber Leitplanken entwickelt, die uns sehr geholfen haben, in dieser neuen Welt erfolgreiche Produkte zu bauen. Es ist weniger ein Framework. Eher eine Haltung. Der Dreiklang lautet Soul – System – Speed:

Soul ist die Fähigkeit, über digitale Produkte zu denken und zu fühlen – den Funken, der ein Produkt von einer Commodity unterscheidet. Die notwendigen Fähigkeiten: Judgment, Cultural Fluency, Meaning-Setting Narratives.

System sichert Produktionsqualität durch Infrastruktur – Architecture, Testability, Observability, Security, Performance, Reliability –, was den Unterschied zwischen Prototyp und Produktion ausmacht.

Speed entsteht durch Pipeline-Verdichtung – nicht durch bessere Prozesse. Die Pipeline wird radikal reduziert: Perceive → Prompt → Produce → Pitch:

- **Perceive**: Einen Insight wahrnehmen, bevor man formuliert
- **Prompt**: Die AI mit klarem Intent beauftragen
- **Produce**: Direkt produzieren, keine Prototypen
- **Pitch**: Pitchen, was man gebaut hat – nicht was man bauen will

Speed entsteht nicht durch bessere Prozesse. Sie entsteht durch das Streichen von Prozessen.

NEUBAUTEN STATT PFLASTER

Die meisten Organisationen reagieren auf diese Brüche, als handele es sich um kleine Veränderungen. Sie kleben ein AI-Pflaster auf den alten Prozess. Sie schicken Mitarbeiter ins AI-Bootcamp. Sie streuen AI-Kompetenz in Stellenanzeigen.

Die Erschütterungen, die wir erleben, lassen sich nicht durch kosmetische Anpassungen abfedern. Sie gehen an die Substanz, ja, und verlangen oft Neubauten. Die gefährlichste Illusion ist die Illusion der Kontinuität: dass man weitermachen kann wie bisher, nur etwas schneller, etwas digitaler, etwas AI-gestützter. Diese Illusion ist bequem – führt aber aufs Abstellgleis. Nicht alle Rollen, die es heute gibt, werden es morgen noch geben. Nicht alle Karrierewege, die gestern sicher waren, sind es morgen noch.

Das ist kein Grund zur Panik. Es ist ein Grund zum Handeln. Wer heute anfängt, Intent-Formulierung, Nutzerverständnis und strategisches Denken zu entwickeln, wird morgen gefragt sein. Wer wartet, wird von den Ereignissen überrollt.

WAS DIESES BUCH LIEFERT

2017 beschrieb ich die Umwälzung durch digitale Produkte. Heute beschreibe ich die Umwälzung der Produktentwicklung selbst. Ein neues Betriebsmodell für eine neue Wirklichkeit:

Soul – System – Speed. Der Dreiklang als Antwort auf das Jevons-Paradoxon: Soul liefert den Funken, System die Zuverlässigkeit, Speed die Beschleunigung.

Die Intent-to-Production Pipeline. Der klassische Genehmigungsablauf – ob Wasserfall oder SAFe-Bürokratie – ist tot. Er verwechselt Dokumentation mit Fortschritt. Die Pipeline ersetzt ihn: Perceive, Prompt, Produce, Pitch. Speed in der Praxis.

Product Teams. Keine parallel skalierten Teams mehr, keine Abhängigkeiten. Keine Übergaben. Keine Silos. Kleine, selbständige Einheiten, die Verantwortung von Anfang bis Ende tragen.

Outcome-Governance. Schluss mit Feature-Zählen. Schluss mit Velocity-Messung. Schluss mit Output-Theater. Steuerung, die nur eine Frage stellt: Haben wir für den Nutzer und das Business etwas erreicht, das messbar ist?

Dieses Buch beschreibt, was jetzt zu tun ist:

- **Teil I: Der neue Kontext** kartiert das veränderte Gelände durch AI. Kapitel 2 zerlegt, warum Agilität in der Krise steckt. Kapitel 3 untersucht die Umwälzung in der Softwareentwicklung.

- **Teil II: Das neue Betriebsmodell** liefert die Leitplanken für diese neue Wirklichkeit. Kapitel 4 führt Soul – System – Speed als neue Prinzipien ein. Kapitel 5 beschreibt die Intent-to-Production-Pipeline.

- **Teil III: Umsetzung** macht es greifbar. Kapitel 6 führt die Product Teams ein. Kapitel 7 markiert die Risiken und Fehlmuster – den Weg, auf dem diese Umwälzung scheitern kann. Der Epilog versucht einen einen Ausblick auf das zu geben, was kommt, und erweitert den Horizont noch einmal.

WAS BLEIBT

Nicht alles ändert sich. Manche Prinzipien sind zeitlos:

Nutzerzentrierung. Produkte gehören dem Nutzer. Nicht der produzierenden Organisation. Nicht den Prozesse. Das war 2016 richtig. Das ist 2026 richtig. Das wird 2036 richtig sein.

Versuchen und Lernen. Niemand weiß vorher, was funktioniert. Die einzige Methode, es herauszufinden: ausprobieren, messen, lernen. Das Tempo der Versuche ändert sich jetzt. Der Grundsatz nicht.

Menschen vor Prozessen. Menschen bauen Produkte. Nicht Regelwerke. Das beste Framework in den Händen eines unmotivierten Teams erzeugt nur Rauschen.

Outcome vor Output. Was zählt, ist nicht, was wir liefern. Es zählt, was wir erreichen. Features zu zählen ist leicht. Wirkung zu messen ist schwer. Aber nur Wirkung zählt.

Diese Grundsätze bilden das Fundament, auf dem das neue Betriebsmodell steht. Sie sind der rote Faden zwischen 2017 und heute.

FÜR WEN IST DIESES BUCH?

Dieses Buch richtet sich an alle, die noch was vorhaben:

- **Gründerinnen und Geschäftsführer**, die verstehen wollen, wie sich ihre Produktorganisation wandeln muss – oder überflüssig wird
- **Produktverantwortliche**, die begreifen wollen, was AI mit ihrer Rolle macht, bevor die Rolle sie überholt

- **Tech-Leads**, die neue Formen für ihre Teams suchen, weil die alten nicht mehr funktionieren
- **Beratungen und Agenturen**, die ihre Auftraggeber in diese neue Wirklichkeit begleiten wollen, statt sie mit Regelwerken von gestern einzuschläfern

Wenn Sie nur einen Gedanken aus diesem Buch mitnehmen, dann diesen:

Kernbotschaft: Der Flaschenhals in der Entwicklung digitaler Produkte – das Schreiben von Code – ist verschwunden. Agentische AI hat ihn beseitigt. Aber Engpässe verschwinden nicht; sie verschieben sich. Der neue Engpass ist Intent: zu wissen, *was* gebaut werden soll und *warum*. Die Erfahrung lehrt, dass diese Verschiebung zu Second-Order-Effekten führt, die wesentlich größer sind als die Effizienzgewinne, die man zu Beginn prognostiziert hat.

ZUSAMMENFASSUNG: WARUM JETZT ALLES ANDERS IST

Sechs Erkenntnisse:

1. **Zwei Wendepunkte:** ChatGPT (November 2022) zeigte eine AI, die das Denken simulieren konnte. Die neue Generation agentischer AI-Coding-Tools kann bauen. Der eigentliche Strukturbruch.

2. **Das Jevons-Paradoxon:** Bauen wird leichter – sich voneinander zu unterscheiden sehr viel schwerer.

3. **Die zwei Paradigmenwechsel:** Agile steckt in der Krise – SAFe löst das falsche Problem. Intent ist der neue Code – die Übersetzungskette von Idee zu Code schrumpft auf eine Konversation.

4. **Soul – System – Speed:** Der Dreiklang als Antwort.

5. **Die 4P-Pipeline:** Perceive → Prompt → Produce → Pitch.

6. **Was bleibt:** Nutzerzentrierung, Versuchen und Lernen, Menschen vor Prozessen, Outcome vor Output. Die Technologie ändert sich. Die Grundsätze nicht.

Im folgenden Kapitel: Warum Agile in der Krise steckt – und warum die Antwort anders lautet als „mehr Agile".

Los geht's.

Das Update des Beststellers

„Transformationale Produkte“

RENAISSANCE

WARUM WIR JETZT DIGITALE PRODUKTE RADIKAL ANDERS ENTWICKELN MÜSSEN

Matthias Schrader

ÜBER DEN AUTOR

Matthias Schrader gehört zu den digitalen Pionieren in Deutschland. Mitte der 1990er-Jahre gründete er SinnerSchrader und entwickelte E-Commerce-Lösungen für Start-ups, deren Produkte in kürzester Zeit börsenreif wurden.

1999 ging SinnerSchrader selbst an die Börse und gehörte zu den wenigen Unternehmen, die den Neuen Markt nicht nur überlebten, sondern sogar aus dieser Zeit gestärkt hervorgingen. 2006 gründete Matthias Schrader die NEXT Conference, die sich innerhalb weniger Jahre als führende Konferenz für die digitale Transformation in Deutschland etablierte. In der Folge unterstützte SinnerSchrader mit über 500 Beratern, Designern und Software-Entwicklern hauptsächlich DAX-Konzerne bei der Entwicklung digitaler Produkte.

2017 übernahm die weltweite Management- und Technologieberatung Accenture das Unternehmen für einen dreistelligen Millionenbetrag. In der Folge war der Autor bis Ende 2022 für das Geschäft von Accenture Interactive/Song in Deutschland, Österreich und der Schweiz verantwortlich. 2024 gründete er gemeinsam mit zahlreichen Weggefährten die AI-native Beratung und Agentur OH-SO Digital in Hamburg, Berlin, München und Prag.

Inhaltsverzeichnis

User Manual

Dieser Text ist auf Denglisch verfasst. Im digitalen Kontext haben wir grundsätzlich mit vielen englischsprachigen Fachbegriffen zu tun und eine strikte Eindeutschung wäre für viele Teile des Fachpublikums unverdaulich gewesen. Wir haben uns bemüht, die zentralen Termini in einem Glossar zusammenzustellen. Bei der ersten Verwendung des Begriffs erscheint er unterstrichen.

Das vorliegende Buch macht an verschiedenen Stellen Rückgriff auf das Buch „Transformationale Produkte“, welches der Autor im Jahr 2017 veröffentlicht hat und zum Standardwerk für die Entwicklung digitaler Produkte im deutschsprachigen Raum geworden ist. Im Folgenden werden jeweils die wesentlichen Aspekte von damals skizziert, soweit sie für das Gesamtverständnis notwendig sind. Das Vorgängerwerk zu kennen, ist daher keine Voraussetzung.

Zudem gibt es in einigen Abschnitten technische Deep Dives. Sie dienen nur der Vertiefung und sind für das Gesamtverständnis ebenfalls nicht zwingend. Überblättern Sie diese ruhig.

Um das Werk aktuell zu halten sind, sämtliche Arbeitsmaterialien und Checklisten digital unter https://XXXXX.com abrufbar.

PROLOGUE

November 2022

Ich erinnere mich an jenen kalten Abend im November. Es war spät, der Arbeitstag längst zerfasert in E-Mails, Slack-Pings und ziellosem Scrollen. Ein Freund schickte einen Link, lakonisch: „Hast du das gesehen?“ Die URL führte zu chat.openai.com.

Chatbots waren so 90er, trotzdem klickte ich mich zur Website und tippte skeptisch: „Erkläre Blockchain in drei Sätzen.“ Die Antwort erschien. Präzise. Strukturiert. Irritierend kompetent.

Ich erhöhte den Einsatz. „Beurteile, wie gut die drei wichtigsten Thesen meines Buches ›Transformationale Produkte‹ von 2017 gealtert sind.“ Das Ergebnis? Ich hätte es in dem Moment nicht besser artikulieren können. Ich machte verwirrt weiter, obwohl ich das Ergebnis bereits ahnte: „Ein Haiku über Artificial Intelligence.“ Es skandierte korrekt. Na klar.

Na klar? Davor hätte ich jeden ausgelacht, der so etwas prophezeit hätte. Jetzt starrte ich auf den Bildschirm wie jemand, der gerade bemerkt hatte, dass hinter dem leuchtenden Panel vor mir eine außerirdische Intelligenz erwacht war. Es fühlte sich an wie ein Riss in der Realität.

Dieselbe Szene wiederholte sich in den folgenden Tagen und Wochen millionenfach. Büros, Küchen, U-Bahnen. Menschen sahen auf ihre Screens und spürten: Ihre Vorstellungen davon, was Computer können, lösten sich gerade auf.

Fünf Tage nach dem Start erreichte ChatGPT eine Million Nutzer. Zwei Monate später durchbrach OpenAI die 100-Millionen-Schallmauer. Keine Technologie und kein Medium verbreiteten sich in der Geschichte schneller.

Das war der ChatGPT-Moment. Der Augenblick, in dem etwas vollkommen Unerwartetes die Labortür aufschlägt und die Welt in neues Licht taucht.

Aber ChatGPT sollte erst der Anfang sein.

Drei Jahre später

Wieder ein später Abend. Wieder saß ich vor dem Bildschirm. Diesmal war es kein Link von einem Freund – ich hatte Claude Code mit dem neuen Opus-4.5-Modell upgedatet, das Anthropic gerade veröffentlicht hatte.

Ich war immer noch in der Desillusionphase des Vibe Codings gefangen. Das ganze Jahr hatten wir die neuen AI Coding Tools bereits im Einsatz. Es begann mit kleinen Prototypen einzelner Funktionen und wuchs bis hin zu MVPs, die sogar in die Produktion gingen. Aber die Arbeit mit den Tools war langsam, fehleranfällig und glücksspielhaft. Die Entropie wuchs mit der Zeit und die Codebasis war irgendwann kaum noch zu managen.

Es schien eine Sackgasse zu sein.

Ich war müde. Ich öffnete ein Terminalfenster und startete Claude Code. Mein Prompt: unsere LLM-Analytics-Plattform mit einer sechsstelligen Anzahl an Programmzeilen komplett aufzuräumen. Robuster, sicherer – ohne das Nutzerverhalten zu ändern. Ein lustlos-aggressiver Brute-Force-Prompt. In den Monaten davor hätte er jede Codebasis zuverlässig geschreddert.

Am nächsten Morgen hatte ich meine abendliche Aktion schon fast vergessen, startete die Analytics-Plattform, die sich an diesem Morgen ein klein wenig flüssiger anfühlte als noch am Tag zuvor. Erst nach einer Stunde fiel mein Blick auf das von vielen Fenstern verdeckte Terminalprogramm. Claude Code meldete, dass es in der Nacht die Codebasis um 40 Prozent reduziert, große Module sauber aufgeräumt und neu strukturiert, Dutzende kleinerer und mittlerer Security-Themen identifiziert und gefixt sowie die Dokumentation upgedatet hatte. Natürlich hätte es auch alles getestet und, da alles fehlerfrei lief, bereits deployed.

Boom. Ich hatte den ganzen Morgen bereits an einer völlig neuen Codebasis gearbeitet.

Ich war ehrlich geschockt. Was passiert hier gerade? Ich dachte an jenen Novemberabend 2022. Damals hatte ich mit den LLMs eine künstliche Intelligenz erlebt, die *denken* mehr oder weniger gut simulieren konnte. Jetzt erlebte ich eine, die *liefern* konnte. Das war keine Simulation mehr. Der Code war tatsächlich neu geschrieben und lief einwandfrei. Entwicklerwochen an Programmierleistung wurden in einem einzigen Prompt verdichtet, der lediglich ein paar Millionen Token an Compute-Leistung triggerte.

Das war der Moment, in dem ich verstand: ChatGPT war erst der Pilot. Jetzt beginnt die eigentliche Serie. Das Skript, das zeigt, wie wir digitale Produkte entwickeln, muss neu verfasst werden.

Davon handelt dieses Buch.

Second Order Effects
Warum dieser Moment anders ist

“Almost all new code written at OpenAI today is written by Codex. We’re at the point where AI writes the code, and humans verify it works.“

— Sam Altman, OpenAI DevDay (Oktober 2025)

Technologische Brüche sind nichts Neues. Das iPhone 2007. Das World Wide Web Mitte der Neunziger. Personal Computer in den Achtzigern. Jeder dieser Momente veränderte Branchen und ließ neue Industrien aus dem Nichts entstehen. Jede Generation veränderte, wie Software entwickelt wurde: Hochsprachen ersetzten Assembler. Objektorientierung ersetzte die prozedurale Programmierung. Agile ersetzte den Wasserfall. DevOps automatisierte den Betrieb. Low-Code versprach einfache Anwendungen für alle.

Aber all diese Umbrüche betrafen Werkzeuge, Methoden und Abstraktionsebenen. Die geistige Kärrnerarbeit – ein Problem vollständig zu durchdringen und in Code zu übersetzen – blieb die Aufgabe des Menschen an der Tastatur. Der Aufwand, Software zu schreiben, hat sich in den letzten Jahrzehnten nicht verringert.

Das ändert sich gerade radikal, weil AI diese geistige Tätigkeit selbst übernimmt. Die Maschine übersetzt jetzt Probleme in Code. Nicht immer fehlerfrei. Nicht jederzeit zuverlässig. Aber gut genug und jeden Monat besser, um die Frage zu stellen: wie sieht eine Welt aus, in der AI das Problem der Software-Entwicklung gelöst hat?

STILLE REVOLUTION

Mit den agentischen AI-Coding-Tools wie Claude Code werden Large Language Modelle (LLMs) zu Produktivitätsmonstern. Sie generieren mittlerweile Code, der direkt einsatzbereit ist.

Diese Erfahrung wiederholt sich gerade rund um den Globus. Entwickler berichten, dass sie innerhalb weniger Wochen von manuellem Coding auf 80 Prozent agentischem AI-Coding umgestiegen sind. Die größte Veränderung ihrer Arbeitsweise, die sie je erlebt haben.

Die breite Öffentlichkeit ahnt von dieser Revolution wenig. Wahr ist aber auch: Die Kluft zwischen dem, was in den AI-zentrierten Softwareunternehmen und den „normalen" IT-Abteilungen passiert, wächst täglich.

Die Ironie: Wer programmiert, programmiert jetzt auf Englisch. Oder eben Deutsch. Nicht in Python, nicht in JavaScript – in natürlicher Sprache. Das kratzt am Ego jener Profession, die bisher programmiert hat. Aber der Hebel ist zu groß, um ihn zu ignorieren.

Die Antwort darauf, wie eine Welt aussieht, in der das Problem der Softwareentwicklung gelöst ist, lautet: Es wird eine radikal neue Welt sein. Dazu muss ich einmal ausholen, warum Softwareentwicklung bisher überhaupt ein Problem war.

Digitale Produkte

2017 schrieb ich ein Buch über transformationale Produkte, das zum Klassiker der digitalen Produktentwicklung wurde. Meine Beobachtung war damals: Die Digitalisierung transformiert nicht nur menschliches Verhalten und das, was Unternehmen tun – sie verändert auch, was Unternehmen sind. Ich verdichtete die Methoden erfolgreicher Silicon-Valley-Unternehmen mit meinen eigenen Erfahrungen in der Entwicklung digitaler Produkte für Startups und mit den digitalen Schnellbooten deutscher Konzerne seit Mitte der 1990er Jahre.

Meine Kernthesen waren:

These 1: Produkte werden zu Diensten. Die Grenze zwischen Produkt und Service bröckelt. Ein Auto ist kein Auto mehr – es ist Mobilität als Dienst. Eine Software ist keine Software mehr – sie ist ein sich ständig wandelndes Erlebnis.

These 2: Dienste werden zu Plattformen. Wer nur verkauft, verliert. Wer Ökosysteme baut, gewinnt. Amazon, Apple, Google – sie alle haben begriffen, dass der eigentliche Wert in der Steuerung von Netzwerkeffekten liegt, nicht im Einzelprodukt.
These 3: Die S-Kurve ist das Schicksal. Jede Technik durchläuft denselben Zyklus: langsamer Beginn, sprunghaftes Wachstum, Sättigung. Wer die Kurve lesen kann, sieht die Zukunft. Wer sie übersieht, wird von ihr überrollt.

These 4: Experiences schlagen Features. Gewohnheitsprägende Produkte – jene, die einen sich selbst verstärkenden Kreislauf aus Nutzung, Belohnung und Gewohnheit schaffen – dominieren die Feature-Inflation.

Das Buch enthielt auch ein umfangreiches Playbook für die Entwicklung digitaler Produkte als Blaupause für all jene, die sich erst ab Mitte der 2010er Jahre auf den Weg machten, ihr Kerngeschäft und ihre Produkte konsequent für eine digitale Welt auszurichten.

Eine Dekade später stellt sich die Frage, die kaum jemand laut ausspricht: Wann hat Ihr letztes Produkt-Release tatsächlich einen messbaren Unterschied gemacht? Nicht „Wir haben geliefert". Nicht „Das Projekt wurde abgeschlossen". Oder gar „Das Ticket ist geschlossen". Stattdessen: Wann hat ein digitales Produkt Ihres Unternehmens zuletzt das Nutzerverhalten verändert, Marktanteile gewonnen oder einen Geschäftsbereich transformiert?

Wenn Sie jetzt nachdenken müssen, sind Sie nicht allein.

Ein durchschnittliches Feature braucht in deutschen Unternehmen 14 bis 18 Monate von der ersten Idee bis zur Livestellung. Die Entwicklungskosten pro Feature sind in den letzten fünf Jahren deutlich gestiegen. Branchenstudien zeigen regelmäßig: Von zehn gestarteten Produktinitiativen erreichen nur zwei bis drei ihr ursprüngliches Ziel. Und Nutzungsdatenanalysen

(etwa von Pendo oder Amplitude) belegen, dass die Mehrheit der Features nur von einem Bruchteil der Nutzer regelmäßig genutzt wird.

Softwareentwicklung ist hart. Und statt alle Energie in die Kreation digitaler Produkte zu investieren, wurde die meiste Energie darauf gelegt, überhaupt „zu liefern" und — mit zunehmenden Schwierigkeiten im Geschäft — immer günstiger zu entwickeln. Das Ergebnis kann man heute im niedrigen Digitalisierungsgrad deutscher Unternehmen und Behörden betrauern.

VERSCHWUNDENER FLASCHENHALS

Digitale Produkte entstehen in crossfunktionalen Teams. Das war mein Mantra – und es stimmt immer noch auf gewisse Weise. Aber das war ein unzulässiger Shortcut. Ich unterschätzte massiv die Schwierigkeit, erprobte Vorgehensmodelle im Kleinen auf große Unternehmenseinheiten zu skalieren.

Der Flaschenhals war immer die Softwareentwicklung. Nicht die Ideen. Nicht das Design. Nicht das Marketing. Sondern die Fähigkeit, Ideen in funktionierenden Code zu übersetzen. Die Entwickler kamen nicht hinterher (wir werden später sehen: Es lag nicht an ihnen). Das Backlog wuchs schneller als die Lieferfähigkeit.

Was taten die Unternehmen? Sie versuchten, den Flaschenhals zu weiten.

Agile versprach: kleinere Pakete, kürzere Zyklen, weniger Verschwendung. Es half – ein bisschen. Aber der Engpass blieb.

SAFe versprach: Skalierung. Wenn ein Team nicht reicht, nehmen wir zehn. Koordination durch Frameworks. Das Ergebnis? Mehr Koordination, mehr Meetings, mehr Overhead – und dieselbe Time-to-Market. Der Flaschenhals wurde nicht geweitet. Er wurde mit Bürokratie ummantelt.

Offshoring versprach: mehr Entwickler, weniger Kosten. Das Ergebnis? Mehr Entwickler, mehr Kommunikationsprobleme, mehr Qualitätsprobleme – und: längere Time-to-Market. Der Flaschenhals wurde nicht geweitet. Er wurde verlagert.

Der Flaschenhals der Softwareentwicklung, der Engpass, um den wir zwanzig Jahre lang unsere gesamte Industrie organisiert hatten, existiert nicht mehr. In der alten Welt gab es einen natürlichen Filter: die Kosten der Umsetzung. Nicht jede Idee konnte verwirklicht werden, weil das Bauen teuer war. Der Filter war auch ein Schutz. Er schützte Organisationen (und einen selbst) vor schlechten Ideen und zwang zu priorisieren. Er sortierte schlechte Ideen aus, bevor sie zu schlechtem Code wurden.

Mit AI hat sich die Gleichung verändert. Was früher der Engpass war gibt es jetzt im Überfluss. Der neue Engpass ist die Fähigkeit zu entscheiden, *welcher* Code geschrieben werden soll. Der Intent – die präzise Formulierung dessen, was wir wollen und warum – wird zum kritischen Pfad. Ein einzelner Product Engineer mit den richtigen Tools liefert heute, wofür früher ein Team Wochen brauchte. Die Produktionskapazität ist explodiert.

Das ist keine inkrementelle Veränderung. Das ist ein Strukturbruch.

Dieses Buch handelt davon, was das bedeutet.

- Was bedeutet es für Teams, wenn der Engpass nicht mehr die Entwicklung ist?
- Was bedeutet es für Prozesse, wenn Releases in Stunden statt in Wochen möglich sind?
- Was bedeutet es für die Entwicklung digitaler Produkte, wenn sich die Randbedingungen dramatisch verändern?

Wenn Sie jahrelang gegen eine Wand gerannt sind und die Wand plötzlich verschwindet – dann stürzen Sie. Oder Sie lernen zu fliegen.

DAS JEVONS-PARADOXON

1865 beobachtete der britische Ökonom William Stanley Jevons einen seltsamen Widerspruch: James Watts' verbesserte Dampfmaschine senkte den Kohleverbrauch pro Energieeinheit um 75 Prozent. Die Erwartung: weniger Kohleverbrauch. Die Wirklichkeit: Der Verbrauch stieg sprunghaft. Die höhere Leistung machte neue Anwendungen erst wirtschaftlich und überkompensierte die Einsparungen beim Kohleverbrauch.

Das Muster wiederholt sich in jeder Technologiewelle:

Als Computing billig wurde, rechneten wir nicht einfach schneller. Die erste Generation von Computern war darauf optimiert, mathematische Berechnungen effizienter durchzuführen – etwa für Artillerietabellen, Versicherungsmathematik und wissenschaftliche Simulationen. Das war der First-Order-Effect: dieselben Aufgaben schneller erledigt. Aber dann entstanden Spreadsheets – nicht als schnellere Buchhaltung, sondern als völlig neues Werkzeug für „Was-wäre-wenn"-Analysen. Es entstanden relationale Datenbanken, die nicht einfach Karteikarten digitalisierten, sondern völlig neue Abfragemöglichkeiten schufen. Das waren Second-Order-Effects: neue Kategorien, die zuvor nicht existiert hatten.

Als Speicher billig wurde, speicherten wir nicht einfach nur mehr Dateien. Plötzlich wurde es wirtschaftlich sinnvoll, jede Suchanfrage zu speichern, jedes Nutzerverhalten zu protokollieren, jede Interaktion zu archivieren. Google wurde möglich – nicht als bessere Suchmaschine, sondern als Unternehmen, dessen Geschäftsmodell auf der Analyse von Milliarden Datenpunkten basiert. Facebook, das gesamte AdTech-Ökosystem, datengetriebene Geschäftsmodelle jeder Art – sie alle existieren nur, weil Speicher so billig wurde, dass das Sammeln und Analysieren von Daten im industriellen Maßstab wirtschaftlich sinnvoll wurde.

Als Bandbreite billig wurde, kommunizierten wir nicht einfach schneller. Es entstanden die Cloud als dominantes Computing-Paradigma, Software-as-a-Service als Geschäftsmodell und Streaming als Ersatz für Besitz. Netflix konnte nur existieren, weil Bandbreite billig genug wurde, dass das Streamen von HD-Video an Millionen von Haushalten gleichzeitig wirtschaftlich tragbar wurde.

Jetzt erleben wir das Jevons-Paradoxon der Softwareentwicklung:

AI macht die Codeerzeugung um Größenordnungen günstiger. Die naive Erwartung: Wir brauchen weniger Entwickler. Die Wirklichkeit: Wir starten viel mehr Projekte. Eine Flut an Software entsteht.

Die Folge klingt paradox:

- Es wird *leichter*, Software zu entwickeln.
- Es wird *schwerer*, sich zu differenzieren.

Wenn jeder bauen kann, wird das *Was* wichtiger als das *Wie*. Der Funke – das, was ein Produkt transformational macht – wird zum entscheidenden Faktor. Nicht die Fähigkeit, Code zu schreiben. Es geht um die Fähigkeit, den richtigen Code zu schreiben.

Das neue Paradoxon: Softwareentwicklung wird leichter – und zugleich *auf andere Weise schwer*.

DIE UNBEQUEME FRAGE

Es gibt eine Frage, die in den Diskussionen über AI und Softwareentwicklung systematisch vermieden wird: Wer verliert?

Jede Technologiewelle hat Gewinner und Verlierer. Die Gewinner sind sichtbar, sie stehen auf Konferenzbühnen. Die Verlierer verschwinden still.

Wer gewinnt? Dieselben, die bei jeder technologischen Disruption gewinnen: kleine, schnelle, autonome Teams mit klarer Ownership. Menschen, die Entscheidungen treffen, nicht Menschen, die Entscheidungen koordinieren.

Wer verliert? Die Koordinationsklasse. Menschen, deren Jobs nur existieren, um die Komplexität zu managen, die von anderen Menschen erzeugt wird, deren Jobs darin bestehen, Komplexität zu managen. Eine Bürokratie, die sich selbst rechtfertigt – Schicht um Schicht, bis niemand mehr weiß, wo die eigentliche Arbeit aufhört und die Verwaltung der Arbeit anfängt. AI entzieht dieser Pyramide gerade das Fundament.

WARUM DIE MEISTEN PROGNOSEN FALSCH SIND

Die Technologiebranche ist voller Prognosen zu AI und Produktivität. „Entwickler werden 40% produktiver". „Wir werden 30% weniger Entwickler brauchen". Diese Zahlen klingen präzise. Sie sind präzise falsch.

Sie sind falsch, weil sie First-Order-Denken anwenden: Wenn X effizienter wird, brauchen wir weniger X. Das Jevons-Paradoxon zeigt: Das Gegenteil passiert. Effizienz führt zu mehr Nutzung, nicht zu weniger. Aber sie sind auch aus einem tieferen Grund falsch: Sie messen das Falsche. „40% produktiver" – gemessen woran? Lines of Code? Features pro Sprint? Pull Requests pro Woche? All diese Metriken messen Output, nicht Outcome. Sie messen, wie viel produziert wird, nicht, ob das Produzierte einen Unterschied macht.

In einer Welt, in der Code billig ist, ist „mehr Code" keine Errungenschaft. Es ist eine Gefahr. Mehr Code bedeutet mehr Komplexität, mehr Wartung, mehr potenzielle Fehler. Die relevante Frage ist nicht „Wie viel

Code produzieren wir?“ sondern „Wie viel Wirkung erzielen wir pro Einheit Komplexität?“.

Diese Frage stellt fast niemand. Weil sie unbequem ist. Weil sie bedeutet, dass die Produktivität neu definiert werden muss. Weil sie bedeutet, dass viele der aktuellen Metriken überflüssig werden.

ZWEI PARADIGMENWECHSEL

Die Produktentwicklung steht vor zwei radikalen Umbrüchen – und beide ereignen sich exakt jetzt.

① Paradigmenwechsel 1: Agile ist in der Krise

Die agilen Methoden, an denen wir über zwei Dekaden gefeilt haben, liefern nicht mehr. Agil zu denken ist nicht verfehlt. Aber was Unternehmen daraus machten, bewahrte vom ursprünglichen Geist nur noch die Begriffe.

Das nächste Kapitel seziert diese Krise. Hier genügt die Diagnose: Die starken Produktunternehmen fragen nicht: „Wie skalieren wir Agilität?“ Sie fragen: „Wie bleiben wir klein genug, um keine skalierte Agilität zu brauchen?“

② Paradigmenwechsel 2: Intent ist der neue Code

Der zweite Bruch betrifft die Softwareentwicklung selbst. Wenn AI-Werkzeuge den kompletten Routine-Code erzeugen und die Umsetzung vom Engpass zur Massenware wird – was bleibt als menschliche Kernkompetenz? Die Antwort: der Intent.

Ein Einwand liegt auf der Hand: War Intent nicht schon immer wichtig? Jeder gute Produktmanager, jede gute Unternehmerin musste schon immer präzise formulieren, was erreicht werden soll.

Der Einwand ist richtig. Und geht doch am Punkt vorbei.

Der Unterschied ist nicht, dass Intent *neu* wäre. Der Unterschied ist, dass Intent jetzt direkt produktiv wird. Früher war Intent der Anfang einer langen Kette: Intent wurde zur Spezifikation, Spezifikation zur Architektur, Architektur zu Code, Code zu Test. Jeder Übersetzungsschritt dauerte Wochen. Jeder Schritt verlor Wissen. Am Ende stand Code, der mit dem ursprünglichen Intent oft nur noch entfernt verwandt war.

Heute kann Intent, wenn er scharf genug gefasst ist, *unmittelbar* in laufenden Code übersetzt werden. Die Kette zurrt zusammen. Was früher zehn Schritte brauchte, braucht jetzt nur einen. Was früher Wochen dauerte, dauert Stunden.

Das ändert die Wirtschaftlichkeit von Grund auf. Früher war Intent billig und Code teuer. Jetzt ist Code billig und Intent teuer. Wenn Output nichts mehr kostet, wird der Input zum knappen Gut.

INPUT WIRD WICHTIGER ALS OUTPUT

In einer Welt, in der der Engpass der Softwareentwicklung nicht mehr existiert, wird es eine kambrische Explosion digitaler Produkte geben. Die Aspekte, die erfolgreiche digitale Produkte ausmachen, bleiben die gleichen. Doch wie diese Produkte gebaut werden, verändert sich von Grund auf. Wie genau, weiß noch niemand. In den letzten drei Jahren seit dem Siegeszug der generischen AI-Modelle haben wir aber Leitplanken entwickelt, die uns sehr geholfen haben, in dieser neuen Welt erfolgreiche Produkte zu bauen. Es ist weniger ein Framework. Eher eine Haltung. Der Dreiklang lautet Soul – System – Speed:

Soul ist die Fähigkeit, über digitale Produkte zu denken und zu fühlen – den Funken, der ein Produkt von einer Commodity unterscheidet. Die notwendigen Fähigkeiten: Judgment, Cultural Fluency, Meaning-Setting Narratives.

System sichert Produktionsqualität durch Infrastruktur – Architecture, Testability, Observability, Security, Performance, Reliability –, was den Unterschied zwischen Prototyp und Produktion ausmacht.

Speed entsteht durch Pipeline-Verdichtung – nicht durch bessere Prozesse. Die Pipeline wird radikal reduziert: Perceive → Prompt → Produce → Pitch:

- **Perceive**: Einen Insight wahrnehmen, bevor man formuliert
- **Prompt**: Die AI mit klarem Intent beauftragen
- **Produce**: Direkt produzieren, keine Prototypen
- **Pitch**: Pitchen, was man gebaut hat – nicht was man bauen will

Speed entsteht nicht durch bessere Prozesse. Sie entsteht durch das Streichen von Prozessen.

NEUBAUTEN STATT PFLASTER

Die meisten Organisationen reagieren auf diese Brüche, als handele es sich um kleine Veränderungen. Sie kleben ein AI-Pflaster auf den alten Prozess. Sie schicken Mitarbeiter ins AI-Bootcamp. Sie streuen AI-Kompetenz in Stellenanzeigen.

Die Erschütterungen, die wir erleben, lassen sich nicht durch kosmetische Anpassungen abfedern. Sie gehen an die Substanz, ja, und verlangen oft Neubauten. Die gefährlichste Illusion ist die Illusion der Kontinuität: dass man weitermachen kann wie bisher, nur etwas schneller, etwas digitaler, etwas AI-gestützter. Diese Illusion ist bequem – führt aber aufs Abstellgleis. Nicht alle Rollen, die es heute gibt, werden es morgen noch geben. Nicht alle Karrierewege, die gestern sicher waren, sind es morgen noch.

Das ist kein Grund zur Panik. Es ist ein Grund zum Handeln. Wer heute anfängt, Intent-Formulierung, Nutzerverständnis und strategisches Denken zu entwickeln, wird morgen gefragt sein. Wer wartet, wird von den Ereignissen überrollt.

WAS DIESES BUCH LIEFERT

2017 beschrieb ich die Umwälzung durch digitale Produkte. Heute beschreibe ich die Umwälzung der Produktentwicklung selbst. Ein neues Betriebsmodell für eine neue Wirklichkeit:

Soul – System – Speed. Der Dreiklang als Antwort auf das Jevons-Paradoxon: Soul liefert den Funken, System die Zuverlässigkeit, Speed die Beschleunigung.

Die Intent-to-Production Pipeline. Der klassische Genehmigungsablauf – ob Wasserfall oder SAFe-Bürokratie – ist tot. Er verwechselt Dokumentation mit Fortschritt. Die Pipeline ersetzt ihn: Perceive, Prompt, Produce, Pitch. Speed in der Praxis.

Product Teams. Keine parallel skalierten Teams mehr, keine Abhängigkeiten. Keine Übergaben. Keine Silos. Kleine, selbständige Einheiten, die Verantwortung von Anfang bis Ende tragen.

Outcome-Governance. Schluss mit Feature-Zählen. Schluss mit Velocity-Messung. Schluss mit Output-Theater. Steuerung, die nur eine Frage stellt: Haben wir für den Nutzer und das Business etwas erreicht, das messbar ist?

Dieses Buch beschreibt, was jetzt zu tun ist:

- **Teil I: Der neue Kontext** kartiert das veränderte Gelände durch AI. Kapitel 2 zerlegt, warum Agilität in der Krise steckt. Kapitel 3 untersucht die Umwälzung in der Softwareentwicklung.

- **Teil II: Das neue Betriebsmodell** liefert die Leitplanken für diese neue Wirklichkeit. Kapitel 4 führt Soul – System – Speed als neue Prinzipien ein. Kapitel 5 beschreibt die Intent-to-Production-Pipeline.

- **Teil III: Umsetzung** macht es greifbar. Kapitel 6 führt die Product Teams ein. Kapitel 7 markiert die Risiken und Fehlmuster – den Weg, auf dem diese Umwälzung scheitern kann. Der Epilog versucht einen einen Ausblick auf das zu geben, was kommt, und erweitert den Horizont noch einmal.

WAS BLEIBT

Nicht alles ändert sich. Manche Prinzipien sind zeitlos:

Nutzerzentrierung. Produkte gehören dem Nutzer. Nicht der produzierenden Organisation. Nicht den Prozesse. Das war 2016 richtig. Das ist 2026 richtig. Das wird 2036 richtig sein.

Versuchen und Lernen. Niemand weiß vorher, was funktioniert. Die einzige Methode, es herauszufinden: ausprobieren, messen, lernen. Das Tempo der Versuche ändert sich jetzt. Der Grundsatz nicht.

Menschen vor Prozessen. Menschen bauen Produkte. Nicht Regelwerke. Das beste Framework in den Händen eines unmotivierten Teams erzeugt nur Rauschen.

Outcome vor Output. Was zählt, ist nicht, was wir liefern. Es zählt, was wir erreichen. Features zu zählen ist leicht. Wirkung zu messen ist schwer. Aber nur Wirkung zählt.

Diese Grundsätze bilden das Fundament, auf dem das neue Betriebsmodell steht. Sie sind der rote Faden zwischen 2017 und heute.

FÜR WEN IST DIESES BUCH?

Dieses Buch richtet sich an alle, die noch was vorhaben:

- **Gründerinnen und Geschäftsführer**, die verstehen wollen, wie sich ihre Produktorganisation wandeln muss – oder überflüssig wird
- **Produktverantwortliche**, die begreifen wollen, was AI mit ihrer Rolle macht, bevor die Rolle sie überholt

- **Tech-Leads**, die neue Formen für ihre Teams suchen, weil die alten nicht mehr funktionieren
- **Beratungen und Agenturen**, die ihre Auftraggeber in diese neue Wirklichkeit begleiten wollen, statt sie mit Regelwerken von gestern einzuschläfern

Wenn Sie nur einen Gedanken aus diesem Buch mitnehmen, dann diesen:

Kernbotschaft: Der Flaschenhals in der Entwicklung digitaler Produkte – das Schreiben von Code – ist verschwunden. Agentische AI hat ihn beseitigt. Aber Engpässe verschwinden nicht; sie verschieben sich. Der neue Engpass ist Intent: zu wissen, *was* gebaut werden soll und *warum*. Die Erfahrung lehrt, dass diese Verschiebung zu Second-Order-Effekten führt, die wesentlich größer sind als die Effizienzgewinne, die man zu Beginn prognostiziert hat.

ZUSAMMENFASSUNG: WARUM JETZT ALLES ANDERS IST

Sechs Erkenntnisse:

1. **Zwei Wendepunkte:** ChatGPT (November 2022) zeigte eine AI, die das Denken simulieren konnte. Die neue Generation agentischer AI-Coding-Tools kann bauen. Der eigentliche Strukturbruch.

2. **Das Jevons-Paradoxon:** Bauen wird leichter – sich voneinander zu unterscheiden sehr viel schwerer.

3. **Die zwei Paradigmenwechsel:** Agile steckt in der Krise – SAFe löst das falsche Problem. Intent ist der neue Code – die Übersetzungskette von Idee zu Code schrumpft auf eine Konversation.

4. **Soul – System – Speed:** Der Dreiklang als Antwort.

5. **Die 4P-Pipeline:** Perceive → Prompt → Produce → Pitch.

6. **Was bleibt:** Nutzerzentrierung, Versuchen und Lernen, Menschen vor Prozessen, Outcome vor Output. Die Technologie ändert sich. Die Grundsätze nicht.

Im folgenden Kapitel: Warum Agile in der Krise steckt – und warum die Antwort anders lautet als „mehr Agile".

Los geht's.