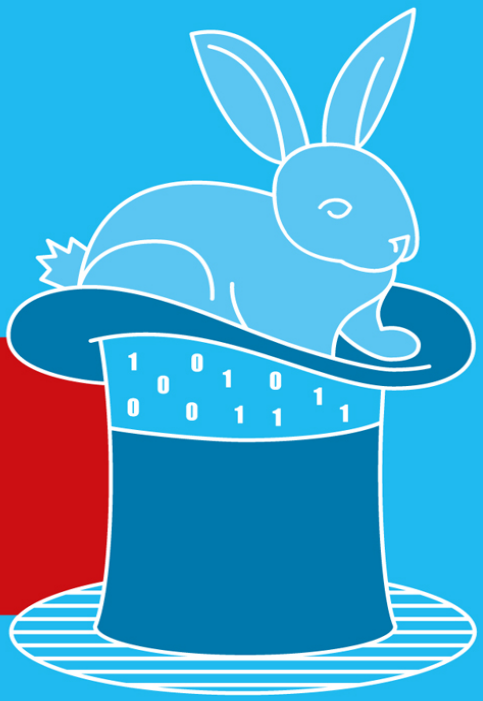
david **THOMAS**
andrew **HUNT**

Jubiläums-ausgabe



2. Auflage

DER PRAGMATISCHE PROGRAMMIERER

Ihr Weg zur Meisterschaft

Listings zum Buch unter:
plus.hanser-fachbuch.de

HANSER

HANSER

David Thomas
Andrew Hunt

Der Pragmatische Programmierer

Ihr Weg zur Meisterschaft

Jubiläumsausgabe

übersetzt von Jürgen Dubau

Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial.

Geben Sie auf **plus.hanser-fachbuch.de** einfach diesen Code ein:

plus-4h16t-r2ag5

Titel der Originalausgabe: „The Pragmatic Programmer: Your Journey To Mastery“, 20th Anniversary Edition

Authorized translation from the English language edition, entitled THE PRAGMATIC PROGRAMMER: YOUR JOURNEY TO MASTERY, 20TH ANNIVERSARY EDITION, 2nd Edition by DAVID THOMAS; ANDREW HUNT. Published by Pearson Education, Inc., publishing as Addison-Wesley Professional, Copyright © 2020.

GERMAN language edition published by Carl Hanser Verlag München, Copyright © 2021

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

Lektorat: Brigitte Bauer-Schiewek
Copy editing: Petra Kienle, Fürstenfeldbruck
Fachlektorat: Steffen Gemkow, Dresden
Layout: Manuela Treindl, Fürth
Umschlagdesign: Marc Müller-Bremer, München, *www.rebranding.de*
Umschlagrealisation: Max Kostopoulos
Titelmotiv: © Max Kostopoulos
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Print-ISBN: 978-3-446-46384-4
E-Book-ISBN: 978-3-446-46632-6
E-Pub-ISBN: 978-3-446-46633-3

Für Juliet und Elli,
Zachary und Elizabeth,
Henry und Stuart

Inhalt

Titelei

Impressum

Inhalt

Geleitwort

Vorwort zur zweiten Ausgabe

Aus der Einleitung zur ersten Ausgabe

1 Eine Pragmatische Philosophie

- Topic 1: Es ist Ihr Leben
- Topic 2: Der Hund hat meinen Quelltext gefressen

Topic 3: Software-Entropie

Topic 4: Steinsuppe und gekochte Frösche

Topic 5: Gut ist gut genug

Topic 6: Ihr Wissensportfolio

Topic 7: Kommuniziere!

2 Ein Pragmatisches Vorgehen

Topic 8: Die Essenz des guten Designs

Topic 9: Das Übel der Wiederholung

Topic 10: Orthogonalität

Topic 11: Umkehrbarkeit

Topic 12: Leuchtspurmunition

Topic 13: Prototypen und Post-it-Zettel

Topic 14: Fachsprachen

Topic 15: Abschätzen

3 Das Handwerkszeug

Topic 16: Die Kraft von Klartext

Topic 17: Kommandospiele

Topic 18: Profi-Editor

Topic 19: Versionsverwaltung

Topic 20: Fehlersuche

Topic 21: Textbearbeitung

Topic 22: Entwickler-Journale

4 Pragmatisch paranoid

Topic 23: Design by Contract

Topic 24: Tote Programme lügen nicht

Topic 25: Abgesichert programmieren

Topic 26: Wie man Ressourcen balanciert

Topic 27: Nicht schneller als die Scheinwerfer

5 Biegen oder Zerbrechen

Topic 28: Entkopplung

Topic 29: Jonglieren mit der realen Welt

Topic 30: Transformierende Programmierung

Topic 31: Erbschaftssteuer

Topic 32: Konfiguration

6 Concurrency

Alles ist nebenläufig

Topic 33: Auflösen der zeitlichen Kopplung

Topic 34: Shared State ist ein falscher Zustand

Topic 35: Aktoren und Prozesse

Topic 36: Blackboards

7 Beim Implementieren

Topic 37: Hören Sie auf Ihr Reptiliengehirn

Topic 38: Programmieren mit dem Zufall

Topic 39: Geschwindigkeit von Algorithmen

Topic 40: Refaktorisieren

Topic 41: Testen fürs Entwickeln

Topic 42: Property-Based Testing

Topic 43: Sicher bleiben da draußen

Topic 44: Dinge benennen

8 Vor dem Projekt

Topic 45: Die Anforderungsgrube

Topic 46: Unlösbare Rätsel

Topic 47: Zusammenarbeit

Topic 48: Die Essenz der Agilität

9 Pragmatische Projekte

Topic 49: Pragmatische Teams

Topic 50: Kokosnüsse bringen’s nicht

Topic 51: Pragmatic Starter Kit

Topic 52: Erfreuen Sie die Anwender

Topic 53: Stolz und Vorurteil

Nachwort

Der moralische Kompass

Stellen Sie sich Ihre Zukunft vor, die Sie sich wünschen

Anhang A: Literaturverzeichnis

Anhang B: Lösungen zu den Übungen

Die Autoren

Geleitwort

Ich weiß noch, als Dave und Andy das erste Mal über die Neuauflage dieses Buchs twitterten. Das war eine große Meldung. Ich sah zu, wie die Entwicklerwelt mit Begeisterung reagierte. Mein Feed kam gar nicht zur Ruhe. Nach zwanzig Jahren ist *Der Pragmatische Programmierer* heute genauso relevant wie damals.

Es sagt viel aus, dass ein Buch mit einer solchen Geschichte solch eine Reaktion bewirkt. Ich hatte das Privileg, für dieses Vorwort ein unveröffentlichtes Exemplar zu lesen, und ich verstand, warum es solches Aufsehen erregte. Es ist zwar ein technisches Buch, aber es als solches zu bezeichnen, erweist ihm einen schlechten Dienst. Technische Bücher schüchtern oft ein. Sie sind vollgestopft mit großen Worten, obskuren Begriffen, überfrachteten Beispielen, bei denen man sich unabsichtlich dumm fühlt. Je erfahrener ein Autor ist, desto leichter vergisst er, wie es ist, neue Konzepte zu lernen oder ein Anfänger zu sein.

Trotz ihrer jahrzehntelangen Programmiererfahrung haben Dave und Andy die schwierige Herausforderung des Schreibens mit der gleichen Begeisterung gemeistert wie Menschen, die diese Lektionen gerade erst gelernt haben. Sie reden nicht von oben herab mit Ihnen. Sie setzen nicht voraus, dass Sie Experte sind.

Sie gehen nicht einmal davon aus, dass Sie die erste Ausgabe gelesen haben. Sie nehmen Sie so, wie Sie sind: als Programmierer, die einfach nur besser werden wollen. Dazu haben sie dieses Buch verfasst, um Ihnen dabei zu helfen, dorthin zu gelangen, jeweils einen machbaren Schritt nach dem nächsten.

Fairerweise muss man sagen, dass sie dies bereits vorher gemacht hatten. Die ursprüngliche Veröffentlichung war voller konkreter Beispiele, neuer Ideen und praktischer Tipps zum Aufbau Ihrer Entwickler-Muskeln und zur Entwicklung Ihres Entwickler-Hirns, die auch heute noch gelten. Aber in dieser aktualisierten Ausgabe werden zwei Sachen verbessert.

Die erste ist offensichtlich: Einige ältere Verweise oder veraltete Beispiele entfallen und werden durch neue, moderne Inhalte ersetzt. Sie werden keine Beispiele für Schleifeninvarianten oder Build Machines finden. Dave und Andy sorgen mit ihren eindringlichen Inhalten dafür, dass die Lektionen immer noch wirksam sind und es keine Ablenkung durch veraltete Beispiele gibt. Hier werden alte Ideen wie DRY (*Don't Repeat Yourself*, dt. „Wiederholen Sie sich nicht") entstaubt und bekommen einen frischen Anstrich, der sie wirklich zum Glänzen bringt.

Aber die zweite Sache macht diese Veröffentlichung wirklich spannend. Nach der ersten Ausgabe hatten beide Gelegenheit, darüber nachzudenken, was sie zu sagen versuchten, was sie ihren Lesern mitgeben wollten und wie das aufgenommen wurde. Sie erhielten Rückmeldungen zu diesen Lektionen. So konnten sie erkennen, wo es knirschte, was verfeinert werden musste, was missverstanden wurde. In den zwanzig Jahren, in denen dieses Buch seinen Weg durch Hände und Herzen von Programmierern auf der ganzen Welt gefunden hat, studierten

Dave und Andy diese Reaktionen und formulierten neue Ideen und neue Konzepte.

Sie haben die Bedeutung der Handlungsfähigkeit kennengelernt und erkannt, dass Entwickler vermutlich mehr davon besitzen als die meisten anderen Fachleute. Sie beginnen dieses Buch mit der einfachen und doch tiefgründigen Botschaft: „Es ist Ihr Leben." Das erinnert uns an unsere Macht bei unserer Code-Basis, in unseren Jobs, in unseren Karrieren. Es gibt den Ton für alles andere im Buch an – dass es mehr ist als nur ein weiteres technisches Buch voller Code-Beispiele.

Dieses Buch sticht wirklich aus den Regalen der Fachbücher heraus, weil hier Leute verstehen, was es bedeutet, ein Programmierer zu sein. Bei der Programmierung geht es darum, zu versuchen, die Zukunft weniger schmerzhaft zu gestalten. Es geht darum, Dinge für unsere Teamkollegen einfacher zu machen. Es geht darum, Fehler zu machen und wieder auf die Beine zu kommen. Es geht darum, gute Gewohnheiten zu etablieren. Es geht darum, die eigenen Instrumente zu verstehen. Coding ist nur ein Teil der Welt des Programmiererdaseins und dieses Buch untersucht diese Welt.

Ich verbringe viel Zeit damit, über diese Reise als Entwickler nachzudenken. Mit Coding bin ich nicht aufgewachsen, habe es nicht am College studiert. Ich habe meine Teenagerjahre nicht damit verbracht, an der Technik herumzubasteln. Ich bin mit Mitte zwanzig in die Welt der Programmierung eingetreten und musste lernen, was es heißt, Programmierer zu sein. Diese Gemeinschaft unterscheidet sich sehr von anderen, denen ich angehörte. Hier engagiert man sich auf einzigartige Weise für das Lernen und die praktische Anwendung, die sowohl erfrischend als auch einschüchternd ist.

Für mich war es, als würde ich in eine neue Stadt ziehen. Ich musste die Nachbarn kennen-lernen, Geschäfte suchen, die besten Cafés finden. Es dauerte, bis mir die Gegend vertraut war und ich die effizientesten Routen kannte, Straßen mit hohem Verkehrsaufkommen vermeiden konnte und wusste, wann die Rush Hour ist. Für das andere Wetter brauchte ich auch neue Klamotten.

Die erste Zeit in einer neuen Stadt kann beängstigend sein. Wie schön, freundliche, sachkundige Nachbarn zu haben, die dort schon eine Weile leben? Wer zeigt einem die Nachbarschaft mit den Cafés und so? Jemand, der Kultur und Puls der Stadt versteht, sodass man sich nicht nur zu Hause fühlt, sondern zur Gemeinschaft beiträgt? Dave und Andy sind solche Nachbarn.

Als relativer Neuling wird man leicht überwältigt – nicht vom Akt des Programmierens, sondern von dem Prozess, ein Programmierer zu werden. Hier muss ein kompletter Mentalitätswandel stattfinden – Gewohnheiten, Verhaltensweisen und Erwartungen müssen sich ändern. Der Prozess, ein besserer Programmierer zu werden, geschieht nicht nur, weil man gelernt hat, wie man programmiert; er muss mit Absicht und bewusster Praxis durchgeführt werden. Dieses Buch ist eine Handreichung, damit Sie ein besserer Programmierer, eine bessere Programmiererin werden.

Aber täuschen Sie sich nicht – es sagt Ihnen nicht, wie Entwickeln geht. Es ist nicht in dieser Weise philosophisch oder wertend. Hier erfahren Sie schlicht und einfach, was Pragmatische Programmierer sind, wie sie arbeiten und wie sie an Code herangehen. Die Autoren überlassen es Ihnen zu entscheiden, ob Sie einer werden wollen. Wenn Sie das Gefühl haben, dass das nichts für Sie ist, werden sie es Ihnen nicht übelnehmen. Aber

wenn Sie entscheiden, dass dies Ihr Weg ist, sind das freundliche Nachbarn, die Ihnen den Weg zeigen.

Saron Jitbarek

Gründer und CEO von CodeNewbie

Host von Command Line Heroes

Vorwort zur zweiten Ausgabe

In den 1990er-Jahren arbeiteten wir mit Unternehmen, deren Projekte Probleme bereiteten. Wir ertappten uns dabei, wie wir jedem das Gleiche sagten: Vielleicht sollten Sie das testen, bevor Sie es ausliefern? Warum geht der Build für diesen Code nur auf dem Rechner von Maria? Warum hat niemand die Anwender gefragt?

Um bei neuen Kunden Zeit zu sparen, begannen wir mit unseren Notizen. Und aus diesen Notizen wurde *Der Pragmatische Programmierer*. Zu unserer Überraschung schien das Buch den Nerv der Zeit zu treffen und es ist in den letzten 20 Jahren nach wie vor beliebt.

Aber 20 Jahre sind in Bezug auf Software viele Lebensalter. Nehmen Sie einen Entwickler von 1999 und setzen Sie ihn heute in ein Team und er würde sich sehr abkämpfen, in dieser fremden neuen Welt klarzukommen. Aber die Welt der 1990er-Jahre ist dem heutigen Entwickler ebenso fremd. Die Verweise des Buchs auf Dinge wie CORBA, CASE-Tools und indexbasierte Schleifen waren bestenfalls kurios und wahrscheinlich eher verwirrend.

Gleichzeitig haben 20 Jahre keinerlei Auswirkungen auf den gesunden Menschenverstand gehabt. Die Technologie mag sich geändert haben, aber die Menschen eben nicht. Praktiken und

Ansätze, die damals eine gute Idee waren, sind auch heute noch gut. Diese Aspekte des Buchs sind gut gealtert.

Als es also an der Zeit war, zum 20. Jahrestag diese Ausgabe zu erstellen, mussten wir eine Entscheidung treffen. Wir könnten die Technologien, auf die wir verweisen, durchgehen, aktualisieren und dann Feierabend machen. Oder wir könnten die Annahmen, die den von uns empfohlenen Praktiken zugrunde liegen, im Lichte weiterer zwei Jahrzehnte Erfahrung noch einmal überprüfen.

Am Ende haben wir beides getan.

Daher ist dieses Buch so etwas wie ein Schiff des Theseus.[1] Etwa ein Drittel der Themen in diesem Buch sind brandneu. Von den übrigen wurden die meisten teilweise oder vollständig umgeschrieben. Unsere Absicht war es, die Dinge klarer, relevanter und hoffentlich etwas zeitloser zu machen.

Wir mussten einige schwierige Entscheidungen treffen. Wir haben den „Ressourcen“-Anhang weggelassen, sowohl weil es unmöglich wäre, auf dem neuesten Stand zu bleiben, als auch weil es einfacher ist, das Gewünschte zu finden. Angesichts des derzeitigen Überflusses an parallel arbeitender Hardware und des Mangels an guten Möglichkeiten, damit umzugehen, haben wir Themen im Zusammenhang mit Concurrency neu organisiert und verfasst. Ergänzt wird das durch Inhalte, die die sich verändernden Einstellungen und Umgebungen reflektieren: von der agilen Bewegung, die wir mit ins Leben gerufen haben, über eine steigende Akzeptanz funktionaler Programmiersprachen bis hin zum wachsenden Bedürfnis nach Berücksichtigung von Privatsphäre und Sicherheit.

Interessanterweise gab es zwischen uns jedoch wesentlich weniger Diskussionen über den Inhalt dieser Ausgabe als beim Verfassen der ersten. Wir waren beide der Meinung, dass die wichtigen Sachen leichter zu erkennen sind.

Das Buch, das Sie in Händen halten, ist jedenfalls das Ergebnis. Bitte genießen Sie es. Vielleicht eignen Sie sich ein paar neue Praktiken an. Vielleicht beschließen Sie, dass einige der von uns vorgeschlagenen Dinge falsch sind. Lassen Sie sich auf Ihr Handwerk ein. Sagen Sie uns Ihre Meinung.

Denken Sie aber vor allem daran, dass es Spaß machen soll.

Aufbau des Buchs

Dieses Buch ist mehr als eine Sammlung von Tipps. Jedes Thema ist in sich abgeschlossen und behandelt einen bestimmten Aspekt. Zahlreiche Querverweise sollen helfen, jedes Thema in Zusammenhänge einzuordnen. Sie können das Buch mit seinen *Topics* in beliebiger Reihenfolge lesen.

Gelegentlich werden Sie auf einen Kasten mit der Beschriftung *Tipp n* stoßen (so wie *Tipp 1*, Kümmern Sie sich um Ihr Können). Tipps betonen einerseits entscheidende Stellen im Text, andererseits sind sie auch für sich alleine gültig – wir selbst wenden sie täglich an. Eine Zusammenfassung aller Tipps finden Sie auf dem Beihefter im Buch.

An geeigneten Stellen haben wir Übungen und weiterführende Aufgaben eingefügt. Während die Übungen relativ einfache Lösungen haben, sind die Aufgaben ziemlich offen gestellt. Um Ihnen eine Vorstellung von unserer Denkweise zu geben, haben

wir unsere Lösungen zu den Übungen im Anhang zusammengetragen. Nur wenige Übungen haben eine einzige richtige Lösung. Die Aufgaben sind eher als Grundlage für Diskussionen oder Ausarbeitungen in Programmierkursen für Fortgeschrittene gedacht.

Am Ende finden Sie auch eine kurze Bibliografie, in der die Bücher und Artikel aufgeführt sind, auf die wir ausdrücklich verweisen.

Schall und Rauch

> *„Wenn ich ein Wort gebrauche“, sagte Humpty Dumpty in leicht verächtlichem Ton, „so bedeutet es das, was ich will, dass es bedeutet – nicht mehr und nicht weniger.“*
>
> Lewis Carroll, „Alice hinter den Spiegeln“

Quer über das Buch verteilt finden sich Fachjargonschnipsel – entweder vollkommen vernünftige Wörter, die für eine technische Bedeutung missbraucht wurden, oder schreckliche Wortschöpfungen, denen von Informatikern ohne Gefühl für Sprache eine Bedeutung zugewiesen wurde. Wir versuchen, solche Fachbegriffe zu definieren oder zumindest ihre Bedeutung zu beschreiben, wenn wir sie zum ersten Mal verwenden. Dennoch sind uns sicher einige durch die Lappen gegangen und andere wie Objekt und Relationale Datenbank sind so gebräuchlich, dass eine Definition nur langweilen würde. Wenn Sie auf ein unbekanntes Wort stoßen, überspringen Sie es bitte nicht. Nehmen Sie sich die Zeit, es im Internet oder einem Informatiklehrbuch nachzuschlagen, und schicken Sie uns eine

E-Mail, damit wir in der nächsten Auflage eine Definition angeben können.

Wir haben uns entschieden, Rache an den Informatikern zu nehmen. Manchmal gibt es durchaus gute Fachwörter für Konzepte, doch wir haben beschlossen, diese Wörter zu ignorieren. Warum? Weil der vorhandene Fachjargon in der Regel auf einen bestimmten Problembereich oder auf eine bestimmte Entwicklungsphase beschränkt ist. Eine der Grundphilosophien dieses Buchs ist jedoch, dass die meisten von uns vorgeschlagenen Techniken universell sind: Modularität zum Beispiel betrifft Quelltext, Entwurf und Teamorganisation. Wenn wir gebräuchliche Fachwörter in einem breiteren Kontext verwenden wollten, erschien es verwirrend, und wir konnten den ganzen Ballast der ursprünglichen Bedeutung nicht loswerden. In diesen Fällen haben wir zum Verfall der Sprache beigetragen und unsere eigenen Wörter erfunden.

Quelltexte und andere Quellen

Die meisten Beispiele in diesem Buch sind kompilierbaren Quelltexten entnommen, die Sie von der Plus.Hanser-Webseite herunterladen können:

Die Quelltexte zu diesem Buch

Geben Sie auf *plus.hanser-fachbuch.de* einfach diesen Code ein:

```
plus-4h16t-r2ag5
```

Dann steht Ihnen der Download zur Verfügung.

Hinweis: In diesem Buch stehen die Namen der entsprechenden Dateien jeweils vor den Codes, die diesen Quelltexten entnommen wurden.

Außerdem haben wir eine eigene Website zum Pragmatischen Programmierer:

Die Website zum Pragmatischen Programmierer

Auf unserer Website

https://pragprog.com/titles/tpp20

finden Sie auch Links zu nützlichen Quellen zusammen mit Aktualisierungen des Buches und Neuigkeiten zu weiteren Aktivitäten der Pragmatischen Programmierer.

Danksagungen für die zweite Ausgabe

Wir haben in den letzten 20 Jahren buchstäblich Tausende von interessanten Gesprächen über die Programmierung geführt, Menschen auf Konferenzen, bei Kursen und manchmal sogar im

Flugzeug getroffen. Jedes einzelne hat unser Verständnis des Entwicklungsprozesses erweitert und zu den Aktualisierungen in dieser Ausgabe beigetragen. Wir danken Ihnen allen (und sagen Sie uns immer wieder, wenn wir uns irren).

Dank geht auch an die Teilnehmer am Beta-Prozess des Buchs. Ihre Fragen und Kommentare haben uns geholfen, die Dinge besser zu erklären.

Bevor wir zur Beta-Version übergingen, baten wir einige Leute um Kommentare zu diesem Buch. Dank an VM (Vicky) Brasseur, Jeff Langr und Kim Shrier für eure detaillierten Kommentare und an José Valim und Nick Cuthbert für eure technischen Überprüfungen. Wir bedanken uns bei Ron Jeffries, dass wir das Sudoku-Beispiel verwenden durften.

Vielen Dank an die Leute von Pearson, die bereit waren, dass wir dieses Buch auf unsere Weise verwirklichen durften.

Ein besonderer Dank gilt der unentbehrlichen Janet Furlow, die alles meistert, was sie angeht, und uns bei der Stange hält.

Und schließlich Applaus für all die Pragmatischen Programmierer da draußen, die in den letzten zwanzig Jahren die Programmierung für alle besser gemacht haben! Wir freuen uns auf zwanzig weitere Jahre.

---

[1] Wenn im Laufe der Jahre jede Komponente eines Schiffs ersetzt wird, wenn sie ausfällt, ist das daraus resultierende Schiff dann immer noch das gleiche Schiff?

Aus der Einleitung zur ersten Ausgabe

Dieses Buch wird Ihnen helfen, ein besserer Programmierer zu werden.

Egal ob Sie alleine programmieren, in einem großen Projektteam oder als Berater mit mehreren Kunden gleichzeitig arbeiten, dieses Buch wird Ihnen helfen, Ihre Arbeit besser zu machen. Dies ist kein Buch über Theorien – wir konzentrieren uns auf pragmatische Themen und darauf, die eigenen Erfahrungen zu nutzen, um fundierte Entscheidungen zu treffen. Das Wort *pragmatisch* kommt vom lateinischen *pragmaticus* – „geschäftskundig“ –, was wiederum vom griechischen πραγματικός abgeleitet ist und etwa „tüchtig“ bedeutet.

In diesem Buch geht es ums Tun.

Programmieren ist ein Handwerk. Vereinfacht gesagt geht es darum, einen Computer dazu zu bringen, das zu tun, was man von ihm erwartet (oder was die Anwender von ihm erwarten). Als Programmierer sind Sie teils Zuhörer, teils Ratgeber, teils Dolmetscher und teils Diktator. Sie versuchen, schwer fassbare Anforderungen festzuhalten und sie so auszudrücken, dass eine

bloße Maschine ihnen gerecht werden kann. Sie versuchen, Ihre Arbeit zu dokumentieren, sodass andere sie verstehen können, und Sie versuchen, Ihre Programme so zu konstruieren, dass andere darauf aufbauen können. Nicht zuletzt tun Sie das alles im Wettlauf gegen die Zeit. Sie arbeiten jeden Tag an kleinen Wundern.

Das ist eine schwierige Aufgabe.

Von vielen Seiten wird Hilfe angeboten. Werkzeughersteller werben mit den Wundern, die ihre Produkte vollbringen können. Methoden-Gurus versprechen garantierten Erfolg mit ihrer Lehre. Jeder behauptet, seine Programmiersprache sei die beste, und jedes Betriebssystem ist angeblich die Antwort auf alle erdenklichen Übel.

Natürlich ist nichts davon wahr. Es gibt keine einfachen Antworten und nicht *die* beste Lösung, sei es ein Werkzeug, eine Programmiersprache oder ein Betriebssystem. Ob ein System besser als ein anderes ist, hängt immer von den gegebenen Umständen ab.

An dieser Stelle ist Pragmatismus notwendig. Sie sollten nicht mit einer bestimmten Technologie verheiratet sein, sondern einen ausreichend weiten Horizont und breite Erfahrung haben, um in konkreten Situationen gute Lösungen auszuwählen. Ihr Horizont beruht auf dem Verständnis der EDV-Grundlagen und Ihre Erfahrung entspringt einer breiten Basis praktischer Projekte. Gemeinsam machen Theorie und Praxis stark.

Sie passen Ihr Vorgehen den aktuellen Gegebenheiten und der Umgebung an. Sie gewichten die Bedeutung aller Faktoren, die auf ein Projekt wirken. Sie bieten angemessene Lösungen, die auf Erfahrung beruhen. Das ist für Sie ein kontinuierlicher Prozess,

der jedes Projekt begleitet. Pragmatische Programmierer erledigen ihre Arbeit und sie machen es gut.

Wer sollte dieses Buch lesen?

Dieses Buch richtet sich an Leserinnen und Leser, die erfolgreichere und produktivere Programmiererinnen und Programmierer werden wollen. Vielleicht sind Sie frustriert, weil Sie glauben, Ihr wahres Potenzial nicht zu erreichen, oder Sie sehen, dass Kollegen mithilfe von Werkzeugen produktiver sind als Sie. Möglicherweise setzen Sie ältere Technologien ein und wollen erfahren, wie neue Ideen bei Ihrer Arbeit eingesetzt werden können.

Wir behaupten weder, wir hätten Antworten auf alle (oder auch nur die meisten) Fragen, noch, dass unsere Ideen in allen Situationen anwendbar wären. Aber wenn Sie unserer Herangehensweise folgen, werden Sie schnell Erfahrung sammeln, Ihre Produktivität wird steigen und Sie werden ein besseres Verständnis für Software-Entwicklung im Ganzen bekommen. Nicht zuletzt werden Sie bessere Software schreiben.

Was macht einen Pragmatischen Programmierer aus?

Jeder Software-Entwickler hat seine individuellen Stärken und Schwächen, Vorlieben und Abneigungen. Im Laufe der Zeit baut sich jeder seine persönliche Arbeitsumgebung auf, die die

Individualität genauso widerspiegelt wie Hobbys, Kleidung oder Haarschnitt. Pragmatische Programmierer haben aber viele der folgenden Eigenschaften gemeinsam.

Neues früh aufgreifen und anwenden (early adopter/fast adapter). Sie haben ein untrügliches Gespür für Technologien und Techniken und probieren gerne Dinge aus. Sie erfassen Neues schnell und bauen es in Ihr übriges Wissen ein. Ihr Selbstvertrauen wächst mit Ihrer Erfahrung.

Neugierde. Sie stellen Fragen. *„Das ist gut – wie haben Sie das gemacht? Hatten Sie Probleme mit dieser Bibliothek? Ich habe von Quantencomputern gehört, was ist da dran? Wie sind symbolische Links implementiert?“* Wie Eichhörnchen sammeln sie kleine Wissensbrocken, von denen jeder auch Entscheidungen beeinflussen kann, die sie erst Jahre später treffen.

Kritisches Denken. Sie nehmen nichts als gegeben hin, ohne die Hintergründe zu kennen. Wenn ein Kollege behauptet: „Das machen wir immer so“, oder ein Verkäufer die Lösung aller Probleme verspricht, werden Sie misstrauisch.

Realismus. Sie gehen Problemen ganz auf den Grund, um sie vollständig zu verstehen. Dadurch bekommen Sie ein gutes Gefühl dafür, wie kompliziert das Problem ist und wie viel Zeit die Lösung erfordert. Ein eingehendes Verständnis davon, dass ein Prozess schwierig sein oder länger dauern könnte, gibt Ihnen das nötige Durchhaltevermögen für Ihre Arbeit.

Hans Dampf in allen Gassen. Sie sind ständig bemüht, sich in eine Vielzahl von Technologien und Umgebungen einzuarbeiten. Auch wenn Sie im Moment hochspezialisiert

arbeiten, sind Sie immer bereit für neue Aufgaben und Herausforderungen.

Die grundlegendsten Eigenschaften haben wir uns bis zum Schluss aufgehoben. Sie gelten für alle Pragmatischen Programmierer und daher formulieren wir sie als Tipp:

Tipp 1

Kümmern Sie sich um Ihr Können.

Wir sind der Meinung, dass Software-Entwicklung sinnlos ist, wenn man sich nicht darum kümmert, es gut zu machen.

Tipp 2

Denken Sie über Ihre Arbeit nach.

Wenn Sie ein Pragmatischer Programmierer sein wollen, denken Sie bei allem, was Sie machen, über Ihr Tun nach. Es geht nicht um ein einmaliges Reflektieren Ihrer gegenwärtigen Arbeit, sondern um ein ständiges, kritisches Bewerten jeder einzelnen Entscheidung. Schalten Sie nie auf Autopilot. Überdenken und kritisieren Sie Ihre eigene Arbeit immer in Echtzeit. Das alte Firmenmotto von IBM, *THINK!*, ist das Mantra des Pragmatischen Programmierers.

Wenn Ihnen das anstrengend erscheint, zeigt das Ihren *Realismus*. Es wird einiges Ihrer kostbaren Zeit in Anspruch nehmen – Zeit, die wahrscheinlich sowieso schon viel zu knapp

ist. Die Belohnung dafür ist eine engere Beziehung zu der Arbeit, die Sie gerne tun, das Gefühl, immer mehr Fachgebiete zu beherrschen, und die Freude an kontinuierlicher, persönlicher Entwicklung. Auf lange Sicht wird sich Ihre Investition auszahlen. Sie und Ihr Team arbeiten effektiver, schreiben leichter wartbaren Quelltext und Sie verbringen weniger Zeit in Besprechungen.

Einzelne Pragmatiker, große Teams

Manche Leute glauben, dass es in großen Teams und komplexen Projekten keinen Platz für Individualität gibt. Sie behaupten: „Software-Entwicklung ist eine Ingenieursdisziplin, die zusammenbricht, wenn einzelne Team-Mitglieder selbst Entscheidungen treffen."

Wir widersprechen.

Software-Entwicklung *sollte* eine Ingenieursdisziplin sein. Das schließt aber individuelles, handwerkliches Können nicht aus. Denken Sie an die großen Kathedralen des Mittelalters. Jede davon verschlang, verteilt über viele Jahrzehnte, Tausende von Personen-Jahren Arbeit. Gesammelte Erfahrungen wurden an die nächste Generation von Baumeistern weitergegeben, die mit dem Erreichten die Baukunst vorantrieben. Die Zimmermänner, Steinmetze, Bildhauer und Glasbläser waren alles Handwerker, die bauliche Anforderungen interpretierten, um ein Ganzes zu schaffen, das den rein mechanischen Aspekt der Konstruktion übertrifft. Der Glaube an ihren individuellen Beitrag stützte das Bauvorhaben: *Wir, die bloße Steine behauen, müssen uns immer Kathedralen vorstellen.*