

# Guidebook für Software Engineers

Der Begleiter für alle Karrierestufen

# DAS INHALTS- VERZEICHNIS

» Hier geht's  
direkt  
zum Buch

<b>Vorwort</b> .....	<b>23</b>
<b>Einleitung</b> .....	<b>25</b>
 <b>Teil I: Grundlagen der Entwicklerkarriere</b>	
<b>1 Karrierepfade</b> .....	<b>29</b>
Arten von Unternehmen .....	29
Big Tech, technologische Großunternehmen .....	30
Mittlere bis große Technologieunternehmen .....	30
Scale-ups .....	30
Start-ups .....	30
Traditionelle, nicht technologieorientierte Unternehmen mit technischen Abteilungen .....	31
Traditionelle, aber technologielaastige Unternehmen .....	32
Kleine, nicht risikofinanzierte Unternehmen .....	32
Öffentlicher Sektor .....	33
Non-Profit-Organisationen .....	33
Beratungsunternehmen, Outsourcing-Firmen und Entwicklungs- agenturen .....	33
Akademische Einrichtungen und Forschungslabore .....	35
Welche Art von Unternehmen ist für Ihre Karriereziele am besten geeignet? .....	35
Typische berufliche Laufbahnen im Software-Engineering .....	35
Einspuriger Karrierepfad .....	35
Zweispuriger Karrierepfad .....	36
Alle Karrierepfade sind einmalig .....	37
Häufig anzutreffende Karrierepfade .....	38

Vergütung und Branchenebenen (Tiers) von Unternehmen .....	40
Ebene 1: Lokaler Markt .....	41
Ebene 2: Lokaler Marktführer .....	41
Ebene 3: Regionaler/internationaler Marktführer .....	41
Contractors und Freelancer .....	42
Vor- und Nachteile der verschiedenen Ebenen .....	43
Cost-Center, Profit-Center .....	44
Alternative Wege, über den Karriereverlauf nachzudenken .....	46
<b>2 Die Karriere in die eigene Hand nehmen .....</b>	<b>49</b>
Sie sind für Ihre Karriere verantwortlich .....	49
Als jemand gesehen werden, der »die Dinge anpackt« .....	50
Schaffen Sie etwas! .....	50
Erledigen Sie möglichst viele Dinge mit weitreichenden Auswirkungen .....	51
Lassen Sie die Leute wissen, dass Sie etwas schaffen .....	51
Führen Sie ein Arbeitsprotokoll .....	51
Ist das Führen eines Protokolls nicht etwas seltsam? .....	53
Bitten Sie um Feedback und geben Sie Feedback .....	53
Bitten Sie um Feedback .....	54
Geben Sie anderen Feedback .....	55
Schlecht formuliertes Feedback »decodieren« .....	56
Machen Sie sich Ihren Manager zum Verbündeten .....	57
Regelmäßige persönliche Treffen mit Ihrem Manager .....	57
Berichten Sie und gehen Sie nicht davon aus, dass Ihr Manager alles weiß .....	57
Verstehen Sie die Ziele Ihres Managers .....	57
Liefen Sie, was vereinbart war, und geben Sie rechtzeitig Bescheid, wenn Sie es nicht schaffen .....	58
Schaffen Sie gegenseitiges Vertrauen mit Ihrem Manager .....	58
Sorgen Sie dafür, dass Ihre Arbeit wahrgenommen wird .....	58
Teilen Sie sich Ihre Kraft gut ein .....	59
Dehnen .....	59
Ausführen .....	60
Leerlauf .....	60
<b>3 Leistungsbeurteilungen .....</b>	<b>63</b>
Früh beginnen: Kontext erfassen und Ziele setzen .....	63
Die wichtigsten Faktoren erkennen und verstehen .....	63
Welches System zur Leistungsbeurteilung wird verwendet? .....	64
Besprechen Sie Ihre Ziele mit Ihrem Manager .....	65
Einigen Sie sich mit Ihrem Manager auf bestimmte Ziele .....	65

Die Macht der Gewohnheit .....	66
Zeichnen Sie Ihre Erfolge auf .....	66
Führen Sie ein Arbeitsprotokoll .....	66
Lassen Sie Ihren Manager von Ihren Fortschritten wissen .....	67
Etwas schaffen .....	67
Anderen helfen .....	67
Notieren Sie, was Sie tun, um anderen zu helfen .....	68
Bitten Sie hin und wieder um konkretes Feedback .....	68
Vor der Leistungsbeurteilung .....	70
Wie sehr können Sie sich auf Ihren Manager als Ihren Fürsprecher verlassen? .....	70
Finden Sie die wichtigen Fristen heraus .....	70
Feedback von Kollegen bekommen .....	71
Nehmen Sie eine Selbstscheinschätzung vor .....	71
Die Leistungsbeurteilung .....	72
Bedenken Sie, dass Leistungsbeurteilungen nur Momentaufnahmen sind .....	72
Erkennen Sie die Dynamik der Leistungsbeurteilung .....	72
Machen Sie sich nicht zu abhängig vom Ergebnis .....	73
Voreingenommenheit ist sehr real .....	73
Seien Sie offen für negatives Feedback und weisen Sie es nicht zurück .....	74
Sie und Ihr Manager sitzen im selben Boot .....	74
Konzentrieren Sie sich auf die langfristigen Ergebnisse .....	74
Vergessen Sie nicht das große Ganze .....	75
<b>4 Beförderungen .....</b>	<b>77</b>
Wie Beförderungen entschieden werden .....	77
Arten von Beförderungsprozessen .....	78
Informeller Beförderungsprozess .....	78
Ein schlanker Beförderungsprozess .....	79
Ein aufwendiger Beförderungsprozess .....	79
Hybride Modelle .....	80
»Terminal Level« .....	80
Beförderungen in Big-Tech-Unternehmen .....	81
Gehalt und Beförderungen .....	81
Beförderung: was vom Unternehmen erwartet wird .....	82
Beförderungsorientierte Softwareentwicklungen .....	84
Ratschläge für die Beförderung .....	85
Die Arbeit organisieren .....	86
Die Bedeutung Ihres Managers .....	87
Bleiben Sie realistisch .....	88

Langfristige Karriereplanung . . . . .	89
Machen Sie Ihr Selbstwertgefühl nicht von Beförderungen und Titeln abhängig . . . . .	89
Versuchen Sie nicht, Menschen aus dem Weg zu drängen . . . . .	89
Viele Investitionen in die Karriere zahlen sich erst später aus	89
Zufriedenheit sollte nicht durch Titel und Karrierestufen definiert werden . . . . .	90
<b>5 In verschiedenen Umgebungen erfolgreich sein . . . . .</b>	<b>91</b>
Produktteams und produktorientierte Engineers . . . . .	91
Produktorientierte Engineers . . . . .	92
Wie wird man ein produktorientierter Engineer? . . . . .	93
Plattformteams . . . . .	94
Vorteile der Arbeit in einem Plattformteam . . . . .	95
Nachteile der Arbeit in einem Plattformteam . . . . .	95
In einem Plattformteam erfolgreich sein . . . . .	96
»Friedenszeiten« im Vergleich mit »Kriegszeiten« . . . . .	97
In Kriegszeiten erfolgreich sein . . . . .	99
In Friedenszeiten erfolgreich sein . . . . .	100
Zwischen beiden Phasen wechseln . . . . .	100
Arten von Unternehmen . . . . .	102
Big Tech- und größere Technologieunternehmen . . . . .	102
Scale-ups in einer mittleren bis späten Phase . . . . .	102
Start-ups in der Frühphase . . . . .	103
Ansätze für den Erfolg in jeder Art von Unternehmen . . . . .	104
<b>6 Jobwechsel . . . . .</b>	<b>105</b>
Neue Möglichkeiten erkunden . . . . .	105
Aktive Jobsuche . . . . .	105
Passiv offen für neue Möglichkeiten . . . . .	106
Wenn Sie mit Ihrem Job zufrieden sind . . . . .	106
Auf Beförderung warten oder den Job wechseln? . . . . .	107
Beförderungen sind in der Regel rückwärtsgewandt . . . . .	108
Je höher Ihr Rang, desto riskanter ist eine neue Stelle . . . . .	108
Je höher die Karrierestufe, desto schwieriger die Beförderung . . . . .	109
Vorbereitung auf Bewerbungsgespräche in der Technologiebranche . . . . .	110
Erstes Screening . . . . .	110
Technisches Screening . . . . .	112
Interviews vor Ort (»On Site«) . . . . .	114
Interviewverläufe auf Staff-Ebene und darüber . . . . .	115
Herabstufung . . . . .	116
Umgang mit Herabstufung . . . . .	117
Höherstufung . . . . .	118

In eine neue Stelle einarbeiten (»Onboarding«) . . . . .	119
Für alle Unternehmen jeder Branchenebene . . . . .	119
Einarbeiten in einer kleineren Firma . . . . .	120
Einarbeiten in einem großen Unternehmen . . . . .	120
Einarbeiten in eine Rolle auf Senior-Stufe oder darüber . . . . .	120
Einarbeiten in eine Rolle auf Staff-Ebene oder darüber . . . . .	121
Denkanstöße . . . . .	122

## Teil II: Der kompetente Softwareentwickler

<b>7 Die Dinge anpacken . . . . .</b>	<b>125</b>
Konzentration auf die wichtigste Aufgabe . . . . .	125
Machen Sie es sich zur Gewohnheit, Ihre wichtigste Aufgabe immer fertigzustellen . . . . .	126
Lernen Sie, Nein zu sagen . . . . .	126
Blockaden beseitigen . . . . .	127
Blockaden erkennen . . . . .	127
Probieren Sie verschiedene Methoden aus, um die Blockaden zu beseitigen . . . . .	128
Holen Sie sich Unterstützung, um Blockaden zu beseitigen . . . . .	129
Lernen Sie, zu eskalieren, ohne Federn zu lassen . . . . .	129
Der »Hilfe! Ich bin blockiert!«-Schummelzettel . . . . .	131
Arbeit aufteilen . . . . .	134
Denken Sie in Stories, Aufgaben und Unteraufgaben . . . . .	134
Priorisieren Sie die Arbeiten, die Sie der Fertigstellung näherbringen . . . . .	135
Haben Sie keine Angst, Aufgaben hinzuzufügen, zu entfernen oder zu ändern . . . . .	136
Zeitaufwand schätzen . . . . .	136
Zeitaufwand für etwas schätzen, das Sie schon einmal getan haben . . . . .	136
Zeitaufwand für etwas schätzen, das Sie noch nicht getan haben . . . . .	137
Mentoren finden . . . . .	139
Der »Tech Tribe of Mentors« . . . . .	140
Das »Wohlwollen-Guthaben« auf einem guten Stand halten . . . . .	141
Jeder hat ein Wohlwollen-Guthaben . . . . .	141
Füllen Sie Ihr Wohlwollen-Guthaben regelmäßig wieder auf . . . . .	141
Vermeiden Sie es nach Möglichkeit, allein zu arbeiten . . . . .	142
Die Initiative ergreifen . . . . .	142
Möglichkeiten, die Initiative zu ergreifen . . . . .	143

<b>8</b>	<b>Programmieren</b>	<b>145</b>
	Programmieren üben – und zwar viel!	145
	Coden Sie regelmäßig	145
	Bitten Sie um Code-Reviews	146
	Lesen Sie so viel Code, wie Sie schreiben	147
	Coden Sie etwas mehr	148
	Lesbarer Code	149
	Was bedeutet »lesbarer Code«?	150
	Worauf Sie achten sollten	151
	Hochwertigen Code schreiben	152
	Den richtigen Abstraktionsgrad verwenden	152
	Richtig mit Fehlern umgehen	154
	Hüten Sie sich vor »unbekannten« Zuständen	155
<b>9</b>	<b>Softwareentwicklung</b>	<b>157</b>
	Eine Sprache besonders gut beherrschen	157
	Lernen Sie die Grundlagen einer Sprache	157
	Eine Ebene tiefer gehen	158
	Der Praktikant, der zum Go-Experten unseres Teams wurde	159
	Lernen Sie das »Haupt«-Framework für eine Sprache	159
	Lernen Sie eine zweite Sprache	160
	In die Breite oder in die Tiefe gehen?	161
	Debuggen	162
	Machen Sie sich mit Ihren Debugging-Werkzeugen vertraut	162
	Beobachten Sie, wie erfahrene Entwicklerinnen und	
	Entwickler debuggen	163
	Lernen Sie, ohne Werkzeuge zu debuggen	163
	Refaktorisieren	164
	Üben Sie die Refaktorisierung so oft wie möglich	164
	Die Refaktorisierungsmöglichkeiten Ihrer IDE kennen und nutzen	166
	Tests refaktorisieren	166
	Machen Sie sich die Refaktorisierung zur täglichen Gewohnheit	167
	Testen	167
<b>10</b>	<b>Werkzeuge des produktiven Entwicklers</b>	<b>169</b>
	Ihre lokale Entwicklungsumgebung	169
	Ein schneller Zyklus aus Bearbeiten → Kompilieren/Ausführen	
	→ Ausgabe	170
	IDE und Arbeitsablauf konfigurieren	171
	Häufig verwendete Werkzeuge	172
	Git	172
	Kommandozeile/Terminal	172
	Reguläre Ausdrücke	173
	SQL	173

KI-Assistenten für die Programmierung . . . . .	174
Unternehmensinterne Entwicklungswerkzeuge . . . . .	174
Ihr persönlicher »Produktivitätsschummelzettel« . . . . .	175
Möglichkeiten, schnell zu iterieren . . . . .	175
Lesen Sie bereits vorhandenen Code und verstehen Sie, was er tut . . . . .	175
Wissen, wie das Debugging der CI/CD funktioniert . . . . .	176
Wissen, wie man auf Produktions-Logs und -Dashboards zugreift . . . . .	176
Nehmen Sie nach Möglichkeit nur kleine Codeänderungen vor . . . . .	177
Automatisierte Tests schreiben und ausführen . . . . .	177
Warten Sie nicht auf Code-Reviews, sondern fordern Sie sie an . . . . .	178
Holen Sie sich regelmäßige Feedback . . . . .	178
Zusammenfassung . . . . .	179

### Teil III: Der vielseitige Senior-Engineer

Softwareentwicklung im Vergleich mit Software-Engineering . . . . .	181
Erwartungen an Senior-Engineers über Unternehmen der verschiedenen Ebenen hinweg . . . . .	183
Typische Senior-Titel . . . . .	183
Typische Erwartungen an Senior-Engineers . . . . .	184
Senior als höchste obligatorische Karrierestufe . . . . .	184

## 11 Aufgaben erfolgreich abschließen . . . . . 185

Dinge erledigen: Wahrnehmung und Realität . . . . .	185
Wahrnehmung und Realität können sich unterscheiden . . . . .	186
Kommunizieren Sie, was Sie tun . . . . .	186
Weniger versprechen, Erwartungen übertreffen und mehr kommunizieren . . . . .	187
Kommunizieren Sie Blockaden frühzeitig und machen Sie Lösungsvorschläge . . . . .	188
Ihre eigene Arbeit . . . . .	188
Effizient mit eingehenden Anfragen umgehen . . . . .	188
Wenn es erledigt ist, ist es auch »richtig« erledigt . . . . .	192
Arbeiten Sie in kurzen Iterationen . . . . .	195
Lange Arbeitsphasen . . . . .	195
Ihr Team . . . . .	197
Projekte gliedern und ihren Aufwand schätzen . . . . .	197
Dokumentieren Sie für andere . . . . .	197
Beseitigen Sie Blockaden für Ihr Team . . . . .	197
Werden Sie besser im »außerhalb-der-Box-Denken« . . . . .	198



Das große Ganze	199
Werden Sie produktorientiert	199
Das Geschäft verstehen	200
<b>12 Zusammenarbeit und Teamwork</b>	<b>201</b>
Code-Reviews	201
Der Ton macht die Musik	202
Vor der Freigabe Änderungen anfordern	202
Miteinander sprechen	203
Erbsenzähler-Kommentare	203
Neueinsteiger und Code-Reviews	203
Büro- und zeitzoneübergreifende Reviews	204
Arbeit im Tandem	204
Situationen, in denen Tandems sinnvoll sein können	205
Ansätze, wenn Sie der erfahrene Partner sind	206
Ansätze, wenn Sie weniger erfahren sind	207
Mentoring	208
Informelles Mentoring	208
Formelles Mentoring	208
Startpunkt für das Mentoring: das Vorstellungstreffen	209
Wenn Sie der Mentor sind	210
Wenn Sie der Mentee sind	212
Langfristige Vorteile des Mentorings	212
Feedback geben	213
Besser positives Feedback geben	213
Konstruktives oder negatives Feedback geben	213
Zusammenarbeit mit anderen Engineering-Teams	214
Zeichnen Sie eine Karte der Teams	215
Machen Sie sich mit anderen Teams bekannt	215
Einfluss auf andere	216
Liefen Sie hervorragende Arbeit ab	216
Lernen Sie andere Menschen kennen	216
Beteiligen Sie sich an funktionsübergreifenden Projekten	217
»Werben« Sie für Ihre Arbeit – aber ohne anzugeben	218
<b>13 Software-Engineering</b>	<b>219</b>
Sprachen, Plattformen und Fachgebiete	219
Lernen Sie eine imperative, eine deklarative und eine funktionale Sprache in ihrer ganzen Bandbreite	220
Machen Sie sich mit verschiedenen Plattformen vertraut	221
Steigern Sie Ihre Full-Stack-Fähigkeiten	221
KI-Assistenten können den Wechsel beschleunigen	222

Debugging	222
Wissen, welche Dashboards und Logging-Systeme man sich ansehen muss	223
Erleichtern Sie anderen das Debugging	223
Die Codebasis verstehen	224
Eignen Sie sich genug Wissen zur Infrastruktur an	225
Aus Störungen lernen	225
Technische Schulden	225
Technische Schulden abbauen	226
Die Anhäufung technischer Schulden verringern	227
»Gerade genug« technische Schulden	228
Dokumentation	229
Designdokumente/RFCs (Request for Comments)	229
Testpläne, Roll-out-Pläne und Migrationspläne	229
Dokumentation von Interfaces und Integrationen	230
Versionshinweise (Release Notes)	230
Dokumentation für Neueinsteiger	230
Ein »Teamhandbuch«	231
Betriebshandbuch für Engineers in Rufbereitschaft	231
Benutzerhandbücher und Leitfäden	231
Dokumentation als Aktivität mit großer Tragweite	232
Best Practices auf das gesamte Team ausdehnen	232
<b>14 Testing</b>	<b>235</b>
Unit-Tests	236
Der Senior-Engineer, der keinen einzigen Unit-Tests schrieb	236
Integrationstests	237
UI-Tests	238
Gedankenmodelle für automatisiertes Testen	238
Die Testpyramide	239
Der Test-Pokal	240
Spezialisierte Tests	241
Performancetests	241
Belastungstests	242
Chaostests	242
Snapshot-Tests	243
Tests auf Applikations-/Bundle-Größe	243
Smoke-Tests	243
Manuelle Tests und Plausibilitätstests	244
Andere Tests	244
Testing im Produktivbetrieb	245
Vor- und Nachteile des automatisierten Testens	246

<b>15 Softwarearchitektur</b>	<b>249</b>
Designdokumente, RFCs und Architekturdokumente	250
Das Ziel von RFCs	250
Vorteile	251
RFCs sichten	251
Architekturdokumente	252
Prototyping und Proof-of-Concept	252
Prototyping zum Erforschen	253
Erstellung mit der Absicht, das Ergebnis zu verwerfen	253
Domain-Driven Design	254
Umsetzbare Softwarearchitektur	256
Die geschäftlichen Ziele im Auge behalten	256
Einverständnis der Entscheidungsträger	256
Berechnen Sie aus der Entscheidungsstarre aus	257
Änderungen korrekt einführen	258
Im Software-Engineering gibt es keine endgültigen Entscheidungen	259
Überdenken Sie Ihre architektonischen Entscheidungen	259
Zusammenfassung	260

---

## Teil IV: Der pragmatische Tech-Lead

Typische Berufsbezeichnungen für Tech-Leads	263
Typische Erwartungen an Tech-Leads	263
Wo sich die Wege von IC und Manager trennen	264
<b>16 Projektmanagement</b>	<b>267</b>
Unternehmen, in denen Engineers Projekte leiten	267
Welchen Zweck hat das Projektmanagement?	268
Projekt-Kick-offs und Etappenziele	269
Das Projekt-Kick-off	270
Das Engineering-Kick-off	271
Etappenziele festlegen	272
Sicherstellen, dass Schätzungen nicht bindend sind	272
Die »Physik« von Softwareprojekten	273
Der Projektumfang wächst	274
Wenn sich der Zeitrahmen des Projekts ändert	275
Wenn weniger Menschen an einem Projekt arbeiten können	275
Änderungen bei den am Projekt beteiligten Personen	276
Alltag im Projektmanagement	276
Als Tech-Lead Entscheidungen treffen	278

Risiken und Abhängigkeiten . . . . .	279
Technologische Risiken . . . . .	279
Abhängigkeiten auf Engineering-Ebene . . . . .	280
Abhängigkeiten außerhalb der Engineering-Ebene . . . . .	280
Fehlende Entscheidungen oder kontextbezogene Abhängigkeiten . .	281
Unrealistischer Zeitrahmen . . . . .	281
Nicht genug Leute oder Bandbreite . . . . .	282
Etwas Unvorhergesehenes, das auf halber Strecke im Projekt auftritt . . . . .	282
Keine Ahnung, wie lange etwas tatsächlich dauern wird . . . . .	283
Projekte abschließen . . . . .	283
Was ist mit Projekten, die eher gescheitert als erfolgreich sind? . . . .	285
<b>17 Bereitstellung für den Produktivbetrieb . . . . .</b>	<b>287</b>
Extreme bei der Bereitstellung für den produktiven Einsatz . . . . .	287
YOLO-Shipping . . . . .	287
Sorgfältige mehrstufige Überprüfung . . . . .	288
Typische Bereitstellungsprozesse . . . . .	290
Start-ups . . . . .	290
Traditionelle Unternehmen . . . . .	291
Große Technologieunternehmen . . . . .	291
Das Hauptprodukt von Meta . . . . .	291
Prinzipien und Werkzeuge . . . . .	292
Entwicklungsumgebungen . . . . .	292
Testen und überprüfen . . . . .	292
Überwachung, Bereitschaftsdienst und Störungsmanagement . . . . .	293
Zusätzliche Kontrollebenen . . . . .	294
Separate Entwicklungsumgebungen . . . . .	294
Dynamisch verfügbare Testing-/Bereitstellungsumgebungen	294
Ein eigenes Team für die Qualitätssicherung (Quality Assurance, QA) . . . . .	295
Exploratives Testing . . . . .	295
Canary-Testing . . . . .	296
Feature-Flags und Experimente . . . . .	296
Etappenweise Bereitstellung . . . . .	297
Mandantenfähigkeit (Multi Tenancy) . . . . .	297
Automatische Rollbacks . . . . .	298
Automatische Roll-outs und Rollbacks . . . . .	298
Pragmatische Risiken eingehen . . . . .	298
Weitere Überlegungen . . . . .	300
Den richtigen Ansatz wählen . . . . .	301

<b>18 Stakeholder-Management</b>	<b>303</b>
Das wahre Ziel des Stakeholder-Managements	303
Arten von Stakeholdern	304
Stakeholder nach Art der Abhängigkeit einteilen	305
Finden Sie heraus, wer die Stakeholder sind	306
Stakeholder auf dem Laufenden halten	308
Problematische Stakeholder	309
Persönliche Gespräche	310
Seien Sie transparent und klären Sie auf	310
Bitten Sie um Unterstützung	310
Von Stakeholdern lernen	310
<b>19 Teamstruktur</b>	<b>313</b>
Rollen und Titel	313
Der Unterschied zwischen Titeln und Rollen	313
»Tech-Lead« kann ein Titel oder eine Rolle sein	314
Rollen können implizit oder explizit sein	314
Entscheiden Sie, welche Rollen explizit sind	314
Teamprozesse	315
Führen Sie nicht nur Prozesse ein, sondern entfernen Sie auch welche	316
Den Fokus des Teams steigern	317
Wehren Sie sich gegen plötzliche Fokusänderungen	317
<b>20 Teamdynamik</b>	<b>319</b>
Gesunde Teams	319
Klarheit	319
Ausführung	320
Gute Moral	320
Gesunde Kommunikation	320
Ein engagiertes Team	320
Ungesunde Teams	321
Gründe für ungesunde Teams und der Umgang damit	321
Strukturelle Gründe für Schwierigkeiten des Teams	324
Teams mit wachsenden Schwierigkeiten	325
Uausgesprochene Konflikte und Reden hinter dem Rücken anderer	325
Im Verborgenen wachsende Probleme bei der Umsetzung	325
Gute Arbeit ohne Beachtung	325
Wachsendes Abnutzungsrisiko	326
Ein Team mit vielen Senior-Engineers, denen die Herausforderungen fehlen	326

Teamdynamik verbessern .....	326
Zuerst beobachten ... .....	327
... dann die Dynamik verbessern .....	327
Beziehungen zu anderen Teams .....	329
Zusammenfassung .....	330

---

## Teil V: Vorbildliche Staff- und Principal-Engineers

Typische Erwartungen an Staff+-Engineers .....	334
Engineers auf Staff-Ebene und darüber sind Partner der EMs und PMs .....	335
<b>21 Das Geschäft verstehen .....</b>	<b>337</b>
North Stars, KPIs und OKRs .....	337
North Star .....	338
KPIs .....	338
OKRs .....	339
Werden die richtigen Dinge gemessen? .....	340
Ihr Team und Ihr Produkt .....	341
Den Produktmanager-Hut aufsetzen .....	342
Versetzen Sie sich in den Kunden hinein .....	342
Verstehen, warum Kunden Ihr Produkt nutzen .....	343
Den geschäftlichen Wert Ihres Produkts verstehen .....	344
Erstellen Sie eine SWOT-Analyse für Ihr Produkt .....	345
Ihr Unternehmen .....	345
Was ist das Geschäftsmodell? .....	345
Führen Sie persönliche Gespräche mit den Produktverantwortlichen .....	346
Sprechen Sie mit Leuten außerhalb der Engineering- und Produktbereiche .....	346
Führen Sie persönliche Gespräche mit geschäftlichen Stakeholdern .....	347
Beachten Sie die Kommunikation der Geschäftsleitung .....	348
Sprechen Sie mit Kunden und hören Sie ihnen zu .....	349
Beteiligen Sie sich an strategischen Diskussionen .....	349
Arbeiten Sie an bereichsübergreifenden Projekten .....	349
Führen Sie persönliche Gespräche mit Ihrem Manager und den übergeordneten Führungskräften .....	350
Nehmen Sie sich Zeit, um PRDs zu lesen .....	351
Schaffen Sie Gelegenheiten für zufällige Treffen .....	351
Warum treffen Engineers nur selten auf geschäftliche Entscheidungsträger? .....	352

Börsennotierte Unternehmen .....	353
Start-ups .....	354
Ihre Branche .....	354
<b>22 Zusammenarbeit .....</b>	<b>357</b>
Interne Politik .....	357
Die »falsche« Art der Politik .....	358
Problematische Wahrnehmungen .....	359
Feedback zu »schlechter« Politik geben .....	360
Andere beeinflussen .....	361
Verdienen Sie sich »Vertrauenkapital« .....	362
Stellen Sie Fragen und seien Sie ein aktiver Zuhörer .....	362
Erklären Sie Ihren Standpunkt .....	363
Ergreifen Sie in Designdiskussionen Partei und erklären Sie Ihre Beweggründe .....	363
Krempeln Sie die Ärmel hoch und machen Sie sich an die Arbeit ...	364
Machen Sie Ihre Arbeit sichtbar .....	364
Übernehmen Sie die Führung und ergreifen Sie die Initiative .....	364
Unterstützen Sie andere uneigennützig .....	365
Verbessern Sie Ihre Schreibfähigkeiten .....	365
Zusammenarbeit mit Managern .....	366
Zusammenarbeit mit Kollegen auf Staff+-Ebene .....	367
Das Netzwerk erweitern .....	368
Finden Sie einen Mentor im Unternehmen .....	368
Arbeit an teamübergreifenden Projekten .....	368
An internen Schulungen teilnehmen .....	369
Treffen mit Menschen von außerhalb Ihres Teams .....	369
Netzwerk und Einfluss gehören zusammen .....	369
Anderen helfen .....	369
Stellen Sie sich als Mentor zur Verfügung .....	370
Förderung .....	370
<b>23 Software-Engineering .....</b>	<b>373</b>
Coding, das Sie immer noch selbst erledigen .....	373
Programmieren in Bursts .....	373
Im Tandem in Bursts programmieren .....	374
Problematischen Projekten helfen .....	374
Passen Sie Ihren Stil dem Team an .....	375
Gehen Sie strategisch vor .....	375
Hilfreiche Engineering-Prozesse .....	376
Definieren Sie »fertig« .....	376
Stilregeln für das Programmieren .....	377
Code-Reviews .....	378

Post-Commit-Reviews .....	378
Automatisierte Tests und Testing im Produktionsbetrieb .....	379
Scaffolding für neue Dienste und Komponenten nutzen .....	379
Roll-out-Hygiene bei Experimenten .....	379
Dashboards zur Kontrolle der Systemgesundheit .....	380
Engineering-Praktiken für schnelles Iterieren .....	380
Continuous Integration (CI) .....	380
Continuous Deployment (CD) .....	381
Trunk-basierte Entwicklung .....	382
Feature-Flags .....	383
Monorepos .....	383
Microservices oder Monolithen .....	384
Werkzeuge, die Engineers effizienter machen .....	385
Dienstkatalog .....	385
Codesuche .....	385
Entwicklerportale .....	386
Cloudbasierte Entwicklungsumgebungen .....	386
KI-Programmierwerkzeuge .....	388
Kaufen, selbst erstellen oder übernehmen? .....	389
Compliance und Datenschutz .....	390
Regulierung .....	390
Logging .....	391
Auditierung .....	392
Sichere Entwicklung .....	392
<b>24 Zuverlässige Softwaresysteme .....</b>	<b>395</b>
Für Zuverlässigkeit zuständig sein .....	395
Logging .....	396
Verbindliche Logging-Praktiken .....	397
Ein Logging-Leitfaden, der sich bewährt hat .....	397
Nutzen Sie ein Framework, das ein »korrektes« Logging durchführt .....	398
Überwachung .....	398
50er-, 95er- und 99er-Perzentile .....	398
Zu überwachende Dinge .....	399
Geschäftliche Kennzahlen überwachen .....	400
Warnmeldungen .....	401
Dringlichkeit von Warnungen .....	402
Falscher Alarm .....	402
Statische Grenzwerte versus Anomalieerkennung .....	403
Bereitschaftsdienst .....	404
Typische Bereitschaftsdienste .....	405
Ein spezielles Bereitschaftsteam oder Rufbereitschaft für alle Teams? .....	405



Runbooks für den Bereitschaftsdienst .....	406
Vergütung für den Bereitschaftsdienst .....	407
Sollten Engineers »normale« Arbeit leisten, während sie in Bereitschaft sind? .....	408
Burn-out durch Bereitschaftsdienst .....	409
Umgang mit Störungen .....	409
Erkennen einer Störung .....	410
Beheben der Störung .....	410
Nachbereitung des Vorfalls .....	411
Resiliente Systeme schaffen .....	413
Planungsphase .....	413
Während der Programmierungsphase .....	414
Simulieren Sie Fehler und testen Sie die Systemantwort .....	414
<b>25 Softwarearchitektur .....</b>	<b>417</b>
Halten Sie die Sache so einfach wie möglich .....	418
Den Fachjargon kennen, aber nicht unnötig benutzen .....	419
Architektonische Schulden .....	420
Erstellung zusätzlicher Dienste, um schneller voranzukommen .....	420
Nicht aufgeteilte Monolithen .....	420
Nicht funktionale Probleme .....	421
Veraltete Sprachen oder Frameworks .....	421
One-Way-Door- oder Two-Way-Door-Entscheidungen .....	422
Two-Way-Door-Entscheidungen .....	422
One-Way-Door-Entscheidungen .....	422
Two-Way-Doors, die zu One-Way-Doors werden .....	424
Entscheidungen im Mittelfeld .....	424
Die Tragweite Ihrer Entscheidungen .....	425
Skalierbare Architektur .....	426
Skalierung, um ein Wachstum neuer geschäftlicher Anwendungsfälle zu ermöglichen .....	426
Skalierung, um ein Ansteigen von Datenmenge, Nutzung und Traffic zu ermöglichen .....	427
Architektonische Entscheidungen und geschäftliche Prioritäten .....	428
Architektonische Entscheidungen auf geschäftliche Ziele und Wachstum ausrichten .....	428
Verbinden Sie Änderungen an der Architektur mit geschäftlichen Initiativen .....	429
Manchmal ist gut genug besser als perfekt .....	429
Nahe genug an der praktischen Arbeit bleiben .....	429

Eigenschaften eines Softwarearchitekten .....	430
Eher theoretische Eigenschaften .....	431
Eher praktische Eigenschaften .....	432
Ist diese Einteilung in Archetypen nützlich? .....	434
Zusammenfassung .....	435

---

## Teil VI: Abschluss

<b>26 Lebenslanges Lernen .....</b>	<b>441</b>
Bleiben Sie neugierig .....	442
Stellen Sie Fragen .....	442
Bleiben Sie neugierig und bescheiden .....	443
Lernen Sie weiter .....	444
Tandems und »Shadowing« .....	444
Mentoring .....	445
Eigenständiges Lernen .....	446
Teilen Sie Ihr Wissen .....	447
Sammeln Sie Wissen an .....	447
Ihre Lernweise kann sich im Laufe der Zeit ändern .....	448
Fordern Sie sich selbst immer wieder heraus .....	448
Bleiben Sie mit der Branche auf dem Laufenden .....	450
Machen Sie auch mal Pause .....	451
<b>27 Literaturhinweise .....</b>	<b>453</b>
Zusatzkapitel .....	453
Mit der Branche Schritt halten .....	453
Bücher .....	454
Errata für dieses Buch .....	455
<b>Danksagungen .....</b>	<b>457</b>
<b>Index .....</b>	<b>459</b>