

Inhaltsverzeichnis

Vorwort.	13
Einleitung.	17
Danksagungen.	21
Über den Autor.	23
 Teil I Was ist Software Engineering?	 25
 1 Einführung.	 27
1.1 Engineering – Die praktische Anwendung von Wissenschaft.	27
1.2 Was ist Software Engineering?	28
1.3 Die Rückeroberung des »Software Engineering«	30
1.4 Wie man Fortschritte macht.	30
1.5 Die Geburt des Software Engineering	31
1.6 Paradigmenwechsel.	33
1.7 Zusammenfassung	34
 2 Was ist Engineering?	 35
2.1 Die Produktion ist nicht unser Problem	35
2.2 Konstruktionsingenieurwesen, nicht Produktionstechnik.	36
2.3 Eine Arbeitsdefinition von Engineering	42
2.4 Engineering != Code	43
2.5 Warum ist Engineering so wichtig?	45
2.6 Die Grenzen von »Handwerk«.	46
2.7 Präzision und Skalierbarkeit.	47
2.8 Komplexität handhaben	48
2.9 Reproduzierbarkeit und Messgenauigkeit.	49
2.10 Engineering, Kreativität und Handwerk	51
2.11 Warum das, was wir tun, kein Software Engineering ist	53
2.12 Kompromisse.	54
2.13 Die Illusion des Fortschritts	55
2.14 Der Weg vom Handwerk zum Engineering	56
2.15 Handwerk ist nicht genug.	57

2.16	Zeit für ein Umdenken?	58
2.17	Zusammenfassung	59
3	Grundlagen eines Engineering-Ansatzes.	61
3.1	Eine Branche im Wandel?	61
3.2	Die Bedeutung von Messungen	62
3.3	Anwendung von Stabilität und Durchsatz.	65
3.4	Die Grundlagen einer Ingenieursdisziplin für die Software- Entwicklung	67
3.5	Experten im Lernen	67
3.6	Experten im Umgang mit Komplexität	68
3.7	Zusammenfassung	70
Teil II	Für das Lernen optimieren	71
4	Iterativ arbeiten	73
4.1	Praktische Vorteile iterativen Arbeitens	75
4.2	Iteration als defensive Design-Strategie	77
4.3	Der Reiz des Plans	79
4.4	Praktische Aspekte des iterativen Arbeitens	86
4.5	Zusammenfassung	88
5	Feedback.	89
5.1	Ein praktisches Beispiel für die Wichtigkeit von Feedback	89
5.2	Feedback bei der Entwicklung	93
5.3	Feedback bei der Integration.	94
5.4	Feedback beim Design	96
5.5	Feedback zur Architektur	99
5.6	Frühzeitiges Feedback bevorzugen	101
5.7	Feedback beim Produktdesign	102
5.8	Feedback in Unternehmen und Kultur	103
5.9	Zusammenfassung	106
6	Inkrementalismus	107
6.1	Die Bedeutung von Modularität	108
6.2	Inkrementalismus im Unternehmen.	110
6.3	Werkzeuge für den Inkrementalismus	112
6.4	Die Auswirkungen von Änderungen begrenzen.	113
6.5	Inkrementelles Design	115
6.6	Zusammenfassung	118

7	Empirismus	119
7.1	In der Realität verankert	120
7.2	Trennung von Epirismus und Experiment	120
7.3	»Ich kenne diesen Bug!«	121
7.4	Selbsttäuschung vermeiden	123
7.5	Eine Realität erfinden, die zu unserer Argumentation passt	124
7.6	Von der Realität geleitet	128
7.7	Zusammenfassung	129
8	Experimentell vorgehen	131
8.1	Was bedeutet »experimentell vorgehen«?	132
8.2	Feedback.	133
8.3	Hypothese	135
8.4	Messung.	136
8.5	Kontrolle der Variablen.	137
8.6	Automatisierte Tests als Experimente	138
8.7	Einordnung der Versuchsergebnisse der Tests in den Kontext.	140
8.8	Der Umfang eines Experiments.	142
8.9	Zusammenfassung	143
Teil III Optimieren für den Umgang mit Komplexität		145
9	Modularität	147
9.1	Merkmale von Modularität	148
9.2	Die Bedeutung von gutem Design wird unterschätzt.	149
9.3	Die Bedeutung von Testbarkeit	151
9.4	Für Testbarkeit zu designen verbessert die Modularität	152
9.5	Services und Modularität	159
9.6	Deploybarkeit und Modularität.	161
9.7	Modularität auf verschiedenen Ebenen	163
9.8	Modularität menschlicher Systeme	164
9.9	Zusammenfassung	166
10	Kohäsion	167
10.1	Modularität und Kohäsion: Grundlagen des Designs.	167
10.2	Der grundlegende Abbau von Kohäsion	168
10.3	Kontext ist wichtig	171
10.4	High-Performance-Software.	175
10.5	Verbindung zur Kopplung	176
10.6	Hohe Kohäsion durch TDD	177

Inhaltsverzeichnis

10.7	Wie erreicht man gute Kohäsion bei Software?	177
10.8	Kosten von schlechter Kohäsion	180
10.9	Kohäsion in menschlichen Systemen	181
10.10	Zusammenfassung	182
11	Trennung von Zuständigkeiten	183
11.1	Dependency Injection	187
11.2	Trennung von wesentlicher und zufälliger Komplexität.	188
11.3	Bedeutung von DDD	192
11.4	Testbarkeit	194
11.5	Ports & Adapters	195
11.6	Wann sollte »Ports & Adapters« eingesetzt werden?	198
11.7	Was ist eine API?	199
11.8	Verwendung von TDD zur Förderung der Trennung von Zuständigkeiten	201
11.9	Zusammenfassung	202
12	Information Hiding und Abstraktion	203
12.1	Abstraktion oder Information Hiding	203
12.2	Was verursacht den »Big Ball of Mud«?	204
12.3	Unternehmerische und unternehmenskulturelle Probleme	204
12.4	Technische Probleme und Probleme des Designprozesses	207
12.5	Furcht vor Over-Engineering	211
12.6	Verbesserung der Abstraktion durch Testen	214
12.7	Die Macht der Abstraktion	215
12.8	Undichte Abstraktionen	217
12.9	Geeignete Abstraktionen auswählen	218
12.10	Abstraktionen in der Anwendungsdomäne	221
12.11	Abstrakte zufällige Komplexität	222
12.12	Systeme und Code von Drittanbietern isolieren	225
12.13	Immer das Verbergen von Informationen bevorzugen	226
12.14	Zusammenfassung	228
13	Kopplung handhaben	229
13.1	Kosten von Kopplung	229
13.2	Hochskalieren	230
13.3	Microservices	231
13.4	Entkopplung kann mehr Code bedeuten	233
13.5	Lose Kopplung ist nicht das Einzige, was wichtig ist	235

13.6	Lose Kopplung bevorzugen.	236
13.7	Wie unterscheidet sich dies von der Trennung von Zuständigkeiten?	238
13.8	DRY ist zu simpel	239
13.9	Asynchronität als Werkzeug für lose Kopplung	241
13.10	Für lose Kopplung designen.	243
13.11	Lose Kopplung in menschlichen Systemen.	243
13.12	Zusammenfassung	245

Teil IV Werkzeuge zur Unterstützung von Engineering in der Software-Entwicklung 247

14	Die Werkzeuge einer Ingenieursdisziplin.	249
14.1	Was ist Software-Entwicklung?	249
14.2	Testbarkeit als Werkzeug	252
14.3	Messpunkte	255
14.4	Schwierigkeiten, die Testbarkeit zu erreichen.	256
14.5	Wie man die Testbarkeit verbessert	260
14.6	Deploybarkeit.	261
14.7	Geschwindigkeit	263
14.8	Die Variablen kontrollieren	264
14.9	Continuous Delivery	266
14.10	Allgemeine Werkzeuge zur Unterstützung von Engineering	267
14.11	Zusammenfassung	268
15	Der moderne Software Engineer	269
15.1	Engineering als menschlicher Prozess	271
15.2	Digital disruptive Unternehmen	272
15.3	Ergebnisse vs. Mechanismen	274
15.4	Langlebigkeit und allgemeine Anwendbarkeit	277
15.5	Grundlagen einer Ingenieursdisziplin.	280
15.6	Zusammenfassung	281
	Stichwortverzeichnis	283