

Inhaltsverzeichnis

Vorwort	15
Einleitung	19
Über den Autor	23
Danksagung	24
Teil I Einführung	25
<hr/>	
1 Was bedeuten »Design« und »Architektur«?	29
1.1 Das Ziel?	30
1.2 Fallstudie	31
1.2.1 Die Signatur des Chaos	32
1.2.2 Die Perspektive der Unternehmensleitung	33
1.2.3 Was ist schiefgelaufen?	34
1.3 Fazit	37
2 Die Geschichte zweier Werte	39
2.1 Verhalten	39
2.2 Architektur	40
2.3 Der größere Wert	41
2.4 Das Eisenhower-Prinzip	42
2.5 Der Kampf für die Architektur	43
Teil II Die ersten Bausteine setzen: Programmierparadigmen	45
<hr/>	
3 Die Paradigmen	47
3.1 Strukturierte Programmierung	47
3.2 Objektorientierte Programmierung	48
3.3 Funktionale Programmierung	48
3.4 Denkanstöße	49
3.5 Fazit	49

4	Strukturierte Programmierung	51
4.1	Die Beweisführung	52
4.2	Eine »schädliche« Proklamation	54
4.3	Funktionale Dekomposition	55
4.4	Keine formalen Beweise	55
4.5	Wissenschaft als Rettung	55
4.6	Tests	56
4.7	Fazit	57
5	Objektorientierte Programmierung	59
5.1	Datenkapselung?	60
5.2	Vererbung?	63
5.3	Polymorphie	65
	5.3.1 Die Macht der Polymorphie	67
	5.3.2 Abhängigkeitsumkehr	68
5.4	Fazit	71
6	Funktionale Programmierung	73
6.1	Quadrierung von Integern	74
6.2	Unveränderbarkeit und Architektur	75
6.3	Unterteilung der Veränderbarkeit	76
6.4	Event Sourcing	77
6.5	Fazit	79
Teil III Designprinzipien		81
7	SRP: Das Single-Responsibility-Prinzip	85
7.1	Symptom 1: Versehentliche Duplizierung	86
7.2	Symptom 2: Merges	88
7.3	Lösungen	89
7.4	Fazit	90
8	OCP: Das Open-Closed-Prinzip	91
8.1	Ein Gedankenexperiment	92
8.2	Richtungssteuerung	95
8.3	Information Hiding	95
8.4	Fazit	96
9	LSP: Das Liskov'sche Substitutionsprinzip	97
9.1	Gesteuerte Nutzung der Vererbung	98

9.2	Das Quadrat-Rechteck-Problem	98
9.3	Das LSP und die Softwarearchitektur	99
9.4	Beispiel für einen Verstoß gegen das LSP	99
9.5	Fazit	101
10	ISP: Das Interface-Segregation-Prinzip	103
10.1	Das ISP und die Programmiersprachen	104
10.2	Das ISP und die Softwarearchitektur	105
10.3	Fazit	105
11	DIP: Das Dependency-Inversion-Prinzip	107
11.1	Stabile Abstraktionen	108
11.2	Factories	109
11.3	Konkrete Komponenten	110
11.4	Fazit	110
Teil IV Komponentenprinzipien		111
12	Komponenten	113
12.1	Eine kurze Historie der Komponenten	114
12.2	Relokatierbarkeit	116
12.3	Linker	117
12.4	Fazit	119
13	Komponentenkohäsion	121
13.1	REP: Das Reuse-Release-Equivalence-Prinzip	121
13.2	CCP: Das Common-Closure-Prinzip	123
13.2.1	Ähnlichkeiten mit dem SRP	124
13.3	CRP: Das Common-Reuse-Prinzip	124
13.3.1	Relation zum ISP	125
13.4	Das Spannungsdiagramm für die Komponentenkohäsion	125
13.5	Fazit	127
14	Komponentenkopplung	129
14.1	ADP: Das Acyclic-Dependencies-Prinzip	129
14.1.1	Der wöchentliche Build	130
14.1.2	Abhängigkeitszyklen abschaffen	131
14.1.3	Auswirkung eines Zyklus in einem Komponenten- abhängigkeitsgraphen	133
14.1.4	Den Zyklus durchbrechen	134
14.1.5	Jitters (Fluktuationen)	135

14.2	Top-down-Design.	136
14.3	SDP: Das Stable-Dependencies-Prinzip.	137
14.3.1	Stabilität.	137
14.3.2	Stabilitätsmetriken	139
14.3.3	Nicht alle Komponenten sollten stabil sein	141
14.3.4	Abstrakte Komponenten	143
14.4	SAP: Das Stable-Abstractions-Prinzip	143
14.4.1	Wo werden die übergeordneten Richtlinien hinterlegt?	143
14.4.2	Einführung in das SAP (Stable-Abstractions-Prinzip).	143
14.4.3	Bemessung der Abstraktion.	144
14.4.4	Die Hauptreihe	144
14.4.5	Die »Zone of Pain«.	145
14.4.6	Die »Zone of Uselessness«.	146
14.4.7	Die Ausschlusszonen vermeiden	147
14.4.8	Abstand von der Hauptreihe	147
14.5	Fazit	149

Teil V Softwarearchitektur 151

15	Was ist Softwarearchitektur?	153
15.1	Entwicklung	155
15.2	Deployment	155
15.3	Betrieb.	156
15.4	Instandhaltung.	157
15.5	Optionen offenhalten.	157
15.6	Geräteunabhängigkeit	159
15.7	Junk Mail	161
15.8	Physische Adressierung	162
15.9	Fazit	163
16	Unabhängigkeit	165
16.1	Use Cases	165
16.2	Betrieb.	166
16.3	Entwicklung	167
16.4	Deployment	167
16.5	Optionen offenhalten.	168
16.6	Layer entkoppeln	168
16.7	Use Cases entkoppeln	169
16.8	Entkopplungsmodi	170

16.9	Unabhängige Entwickelbarkeit	171
16.10	Unabhängige Deploybarkeit	171
16.11	Duplizierung	171
16.12	Entkopplungsmodi (zum Zweiten)	172
	16.12.1 Welcher Modus ist am besten geeignet?	173
16.13	Fazit	174
17	Grenzen: Linien ziehen	175
17.1	Ein paar traurige Geschichten	176
17.2	FitNesse	179
17.3	Welche Grenzen sollten Sie ziehen – und wann?	181
17.4	Wie verhält es sich mit der Ein- und Ausgabe?	184
17.5	Plug-in-Architektur	185
17.6	Das Plug-in-Argument	186
17.7	Fazit	187
18	Anatomie der Grenzen	189
18.1	Grenzüberschreitungen	189
18.2	Der gefürchtete Monolith	190
18.3	Deployment-Komponenten.	192
18.4	Threads	192
18.5	Lokale Prozesse	193
18.6	Services.	193
18.7	Fazit	194
19	Richtlinien und Ebenen	195
19.1	Ebene	196
19.2	Fazit	199
20	Geschäftsregeln	201
20.1	Entitäten	202
20.2	Use Cases	203
20.3	Request-and-Response-Modelle	205
20.4	Fazit	206
21	Die schreiende Softwarearchitektur	207
21.1	Das Thema einer Architektur	208
21.2	Der Zweck einer Softwarearchitektur	208
21.3	Aber was ist mit dem Web?	209
21.4	Frameworks sind Tools, keine Lebenseinstellung	209
21.5	Testfähige Architekturen	210
21.6	Fazit	210

22	Die saubere Architektur	211
22.1	Die Abhängigkeitsregel (Dependency Rule)	213
22.1.1	Entitäten	213
22.1.2	Use Cases	213
22.1.3	Schnittstellenadapter	214
22.1.4	Frameworks und Treiber	214
22.1.5	Nur vier Kreise?	215
22.1.6	Grenzen überschreiten	215
22.1.7	Welche Daten überqueren die Grenzlinien?	216
22.2	Ein typisches Beispiel	216
22.3	Fazit	217
23	Presenters und »Humble Objects«	219
23.1	Das Pattern »Humble Object«	219
23.2	Presenters und Views	220
23.3	Das Testen und die Softwarearchitektur	221
23.4	Datenbank-Gateways	221
23.5	Data Mappers	221
23.6	Service Listeners	222
23.7	Fazit	222
24	Partielle Grenzen	223
24.1	Den letzten Schritt weglassen	224
24.2	Eindimensionale Grenzen	224
24.3	Fassaden	225
24.4	Fazit	226
25	Layer und Grenzen	227
25.1	Hunt the Wumpus	228
25.2	Saubere Architektur?	229
25.3	Datenstromüberschreitungen	231
25.4	Datenströme teilen	232
25.5	Fazit	234
26	Die Komponente Main	235
26.1	Das ultimative Detail	235
26.2	Fazit	239
27	Services – große und kleine	241
27.1	Servicearchitektur?	241

27.2	Vorteile der Services?	242
27.2.1	Denkfalle: Entkopplung	242
27.2.2	Denkfalle: Unabhängige Entwickel- und Deploybarkeit	243
27.3	Das Kätzchen-Problem	243
27.4	Objekte als Rettung	245
27.5	Komponentenbasierte Services	247
27.6	Cross-Cutting Concerns	248
27.7	Fazit	248
28	Die Testgrenze	249
28.1	Tests als Systemkomponenten	249
28.2	Design für Testfähigkeit	250
28.3	Die Test-API.	251
28.3.1	Strukturelle Kopplung	251
28.3.2	Sicherheit	252
28.4	Fazit	252
29	Saubere eingebettete Architektur	253
29.1	App-Eignungstest	256
29.2	Der Flaschenhals der Zielhardware	259
29.2.1	Eine saubere eingebettete Architektur ist eine testfähige eingebettete Architektur.	260
29.2.2	Offenbaren Sie dem HAL-User keine Hardwaredetails	263
29.3	Fazit	269
Teil VI Details		271
30	Die Datenbank ist ein Detail	273
30.1	Relationale Datenbanken	274
30.2	Warum sind Datenbanksysteme so weit verbreitet?	274
30.3	Was wäre, wenn es keine Festplatten gäbe?	275
30.4	Details.	276
30.5	Und was ist mit der Performance?	276
30.6	Anekdote	277
30.7	Fazit	278
31	Das Web ist ein Detail	279
31.1	Der immerwährende Pendelausschlag	280
31.2	Quintessenz	281
31.3	Fazit	282

32	Ein Framework ist ein Detail	283
32.1	Framework-Autoren	284
32.2	Asymmetrische Ehe	284
32.3	Die Risiken	285
32.4	Die Lösung	285
32.5	Hiermit erkläre ich euch zu	286
32.6	Fazit	286
33	Fallstudie: Software für den Verkauf von Videos	287
33.1	Das Produkt	287
33.2	Use-Case-Analyse	288
33.3	Komponentenarchitektur	289
33.4	Abhängigkeitsmanagement	291
33.5	Fazit	291
34	Das fehlende Kapitel	293
34.1	Package by Layer	294
34.2	Package by Feature	295
34.3	Ports and Adapters	297
34.4	Package by Component	299
34.5	Der Teufel steckt in den Implementierungsdetails	304
34.6	Organisation vs. Kapselung	305
34.7	Andere Entkopplungsmodi	308
34.8	Fazit: Der fehlende Ratschlag	309
A	Architekturarchäologie	311
A.1	Das Buchhaltungssystem für die Gewerkschaft	312
A.2	Zurechtschneiden mit dem Laser	317
A.3	Monitoring von Aluminiumspritzguss	321
A.4	4-TEL	322
	A.4.1 Service Area Computer	326
	A.4.2 Ermittlung des Wartungsbedarfs	327
	A.4.3 Architektur	327
	A.4.4 Die große Neugestaltung	329
	A.4.5 Europa	330
	A.4.6 SAC: Fazit	330
A.5	Die Programmiersprache C	331
	A.5.1 C	332
A.6	BOSS	332

A.7	Projekt CCU	333
	A.7.1 Denkfälle: Die Planung	334
A.8	DLU/DRU	335
	A.8.1 Architektur	336
A.9	VRS	337
	A.9.1 Der Name	338
	A.9.2 Architektur	338
	A.9.3 VRS: Fazit	339
A.10	Der Elektronische Receptionist	340
	A.10.1 Der Untergang des ER	341
A.11	Craft Dispatch System	342
A.12	Clear Communications	344
	A.12.1 Die Gegebenheiten	345
	A.12.2 Uncle Bob	346
	A.12.3 Das Telefongespräch	346
A.13	ROSE	347
	A.13.1 Fortsetzung der Debatten	348
	A.13.2 ... unter anderem Namen	348
A.14	Prüfung zum eingetragenen Architekten	349
A.15	Fazit	352
B	Nachwort	353
	Stichwortverzeichnis	357