Langlebige Software-Architekturen

Technische Schulden analysieren, begrenzen und abbauen

DAS INHALTS-VERZEICHNIS

Inhaltsverzeichnis

1 Einleitung 1

1.1 Softwarearchitektur 1
1.2 Langlebigkeit 3
1.3 Technische Schulden 4
1.3.1 »Programmieren kann jeder!« 8
1.3.2 Komplexität und Größe 9
1.3.3 Die Architekturerosion steigt unbemerkt 12
1.3.4 Für Qualität bezahlen wir nicht extra! 13
1.3.5 Arten von technischen Schulden 14
1.4 Was ich mir alles anschauen durfte 15
1.5 Wer sollte dieses Buch lesen? 16
1.6 Wegweiser durch das Buch 16

2 Aufspüren von technischen Schulden 19

2.1 Begriffsbildung für Bausteine 19
2.2 Soll- und Ist-Architektur 21
2.3 Verbesserung am lebenden System 25
2.4 False Positives und generierter Code 41
2.5 Spickzettel zum Sotographen 43

3 Architektur in Programmiersprachen 45

3.1 Java-Systeme 45
3.2 C#-Systeme 50
3.3 C++-Systeme 52
3.4 ABAP-Systeme 53
3.5 PHP-Systeme 55
3.6 TypeScript-Systeme 57

4 Architekturanalyse und -verbesserung 59
4.1 Entwickler und Architektur 59
4.2 Architekturarbeit ist eine Holschuld 60
4.3 Live-Workshop zur Architekturverbesserung 61
4.4 Der Umgang mit den Vätern und Müttern 63
4.5 Technische Schulden im Lebenszyklus 65

5 Kognitive Psychologie und Architekturprinzipien 69
5.1 Modularität 70
5.1.1 Chunking 70
5.1.2 Übertragung auf Entwurfsprinzipien 72
5.1.2.1 Einheiten 73
5.1.2.2 Schnittstellen 75
5.1.2.3 Kopplung 76
5.2 Musterkonsistenz 77
5.2.1 Aufbau von Schemata 78
5.2.2 Übertragung auf Entwurfsprinzipien 80
5.3 Hierarchisierung 84
5.3.1 Bildung von Hierarchien 84
5.3.2 Übertragung auf Entwurfsprinzipien 86
5.4 Zyklen = misslungene Modularität + Muster 88
5.5 Konsequenzen für die Architekturanalyse 89

6 Architekturstile gegen technische Schulden 91
6.1 Regeln von Architekturstilen 91
6.2 Trennung von fachlichen und technischen Bausteinen 92
6.3 Schichtenarchitekturen 95
6.3.1 Technische Schichtung 95
6.3.2 Fachliche Schichtung 96
6.3.3 Infrastrukturschicht 98
6.3.4 Integration von fachlichen Schichten 100
6.4 Hexagonal, Onion und Clean Architecture 101
6.5 Microservices und Domain-Driven Design 103
6.6 Mustersprachen 106
6.6.1 WAM-Mustersprache 108
6.6.2 DDD-Mustersprache 110
6.6.3 Typische Framework-Muster 112
6.7 Langlebigkeit und Architekturstile 114

7 Muster in Softwarearchitekturen 115
7.1 Abbildung der Soll-Architektur auf die Ist-Architektur 115
7.2 Die ideale Struktur: fachlich oder technisch? 118
7.3 Schnittstellen von Bausteinen 123
7.4 Interfaces – das architektonische Allheilmittel? 128
7.4.1 Die Basistherapie 128
7.4.2 Die Nebenwirkungen 130
7.4.3 Feldstudien am lebenden Patienten 133
7.4.4 Der Kampf mit dem Monolithen 136
7.5 Der Wunsch nach Microservices 138
7.5.1 Früh übt sich 139
7.5.2 Der Knackpunkt: das Domänenmodell 141

8 Mustersprachen – der architektonische Schatz! 145
8.1 Die Schatzsuche 145
8.2 Die Ausgrabungsarbeiten 147
8.3 Aus der Schatztruhe 149
8.4 Den Goldanteil bestimmen 153
8.5 Jahresringe 154
8.6 Unklare Muster führen zu Zyklen 155

9 Chaos in Schichten – der tägliche Schmerz 159
9.1 Bewertung des Durcheinanders 161
9.1.1 Ausmaß der Unordnung 162
9.1.1.1 Architekturstile und Zyklen 164
9.1.1.2 Programmzeilen in Zyklen 165
9.1.1.3 Dependency Injection und Zyklen 167
9.1.2 Umfang und Verflochtenheit 167
9.1.3 Reichweite in der Architektur 170
9.2 Das große Wirrwarr 174
9.2.1 Der Schwarze-Loch-Effekt 176
9.2.2 Der Befreiungsschlag 178
9.2.3 Technische Schichtung als Waffe 180
9.2.4 Mustersprache als Leuchtturm 182
9.3 Uneven Modules 185

10 Modularität schärfen 189
10.1 Kohäsion von Bausteinen 190
10.2 Größen von Bausteinen 194
10.3 Größen von Klassen 194
10.4 Größe und Komplexität von Methoden 200
10.5 Lose Kopplung 203
10.6 Kopplung und Größe von Klassen 209
10.7 Wie modular sind Sie? 211

11 Modularity Maturity Index (MMI) 213
11.1 Modularität im MMI 213
11.2 Hierarchie im MMI 214
11.3 Musterkonsistenz im MMI 215
11.4 Berechnung des MMI 216
11.5 Vergleich mit dem MMI 219

12 Geschichten aus der Praxis 221
12.1 Das Java-System Alpha 222
12.2 Das C#-System Gamma 229
12.3 Das C++-System Beta 237
12.4 Das Java-System Delta 246
12.5 Das Java-System Epsilon mit C#-Satelliten 253
12.5.1 Java-Epsilon 253
12.5.2 C#-Epsilon 1 261
12.5.3 C#-Epsilon 2 264
12.6 Das ABAP-System Lambda 268

13 Fazit: der Weg zu langlebigen Architekturen 275

Anhang **279**

A Analysewerkzeuge **281**
A.1 Lattix 283
A.2 Sonargraph-Produktfamilie 284
A.3 SotoArc und Sotograph 286
A.4 Structure101 287

Literatur **291**

Index **299**