

Diese Leseprobe haben Sie beim
 edv-buchversand.de heruntergeladen.
Das Buch können Sie online in unserem
Shop bestellen.
[Hier zum Shop](#)

Auf einen Blick

TEIL I	
Aufbau einer Testinfrastruktur	45
TEIL II	
Testorientiertes ABAP-Design	185
TEIL III	
Agile Neuentwicklung	367
TEIL IV	
Agile Methoden	415
TEIL V	
Testorientierte ABAP-Werkzeuge	505

Inhalt

Vorwort	21
Einleitung	23

1 Einführung 31

1.1 Nachhaltigkeit: Entwicklung und Wartung mit einer Testinfrastruktur ...	32
1.2 Agilität: Praktiken des Agile Software Engineerings	37
1.3 Effizienz: Einhaltung von Designprinzipien	38
1.4 Kommunikation: Diagramme als Gesprächsgrundlage	42

TEIL I Aufbau einer Testinfrastruktur

2 Beispielanwendung für diesen Buchteil 47

2.1 Stammdatenmanagement	48
2.1.1 Änderungsantrag	48
2.1.2 Definition eines Stammdatenmodells	49
2.2 Benutzeroberfläche	50
2.3 Backend	51

3 Codebasierte Verbesserung eines Tests 55

3.1 Einführung in den Testcode	55
3.1.1 Erste Definition einer Testklasse	56
3.1.2 Erste Implementierung der Testklasse	57
3.1.3 ABAP-Unit-Protokoll	64
3.1.4 Diagramme	65
3.2 Allgemeine Clean-Code-Prinzipien	67
3.2.1 Verbesserungsprozess	67
3.2.2 Verbesserung der Definition der Testklasse	69

3.2.3	Verbesserung der Implementierung der Testklasse	70
3.2.4	Diagramme	74
3.3	Testorientierte Clean-Code-Prinzipien	76
3.3.1	Verbesserungsprozess	76
3.3.2	Verbesserung der Implementierung der Testklasse	77
3.3.3	Diagramme	81
4	Designbasierte Verbesserung des Tests	85
4.1	Basisklasse von Testklassen	85
4.1.1	Verbesserungsprozess	86
4.1.2	Extraktion einer Basisklasse aus einer Testklasse	86
4.1.3	Diagramme	88
4.2	Verwendung von Hilfsklassen durch Testklassen	91
4.2.1	Verbesserungsprozess	91
4.2.2	Entwurfsmuster für Klassen	91
4.2.3	Definition und Verwendung der Testdatenklasse	92
4.2.4	Diagramme	95
5	Robuster Integrationstest	99
5.1	Verbesserungsprozess	99
5.2	Unabhängigkeit von Testmethoden	100
5.2.1	Testdatencontainer	101
5.2.2	Singleton-Entwurfsmuster	102
5.3	Wiederholbarkeit von Testmethoden	105
5.3.1	Automatischer Testaufbau	105
5.3.2	Hilfsmethoden der Testklasse	106
5.3.3	Diagramme	110
6	Minimierung von Abhängigkeiten	113
6.1	Vereinfachung einer Verwendung	114
6.1.1	Verbesserungsprozess	115
6.1.2	Vereinfachung der produktiven API	116

6.1.3	Zugriff auf den Testdatencontainer	117
6.1.4	Delegierende Testklasse	117
6.1.5	Diagramme	119
6.2	Abgrenzung einer Verwendung	121
6.2.1	Verbesserungsprozess	122
6.2.2	Geschachtelte Interfaces	122
6.2.3	Diagramme	123
6.3	Unabhängigkeit von einer Erzeugung	125
6.3.1	Verbesserungsprozess	125
6.3.2	Definition einer Factory	126
6.3.3	Implementierung einer Factory	127
6.3.4	Verwendung einer Factory	128
6.3.5	Diagramme	128
6.4	Unabhängigkeit von einer Erweiterung	131
6.4.1	Verbesserungsprozess	131
6.4.2	Abstrakte Oberklasse aller Entitäten	131
6.4.3	Konkrete Unterklasse für Flugverbindungen	133
7	Isolierter Komponententest	137
7.1	Transformation des Integrationstests	138
7.1.1	Verbesserungsprozess	139
7.1.2	Isolierung mit einem Governance-API-Double	139
7.1.3	Konfiguration des Governance-API-Doubles	144
7.1.4	Diagramme	147
7.2	Skalierung mit einer Testsprache	150
7.2.1	Verbesserungsprozess	150
7.2.2	Zustand einer Entität	151
7.2.3	Testmethode: Ungesicherte Änderungen	153
7.2.4	Testmethode: Änderungen zu einem neuen Entitätstyp	155
7.2.5	Testmethode: Enthaltene Änderungen	156
7.3	Testgetriebene Entwicklung	157
7.3.1	Testklasse für das Hervorheben von Löschungen	158
7.3.2	Diagramme	159
7.4	Liskov-Substitution-Prinzip	161
7.4.1	Beispielhafte Verletzung	161
7.4.2	Beispielhafte Erfüllung	163

8	Redesign mit Unit-Tests	165
8.1	Objektorientierte API für Entitäten	167
8.1.1	Entitätstypklasse	167
8.1.2	Selektionsklasse	167
8.1.3	Entitätsmengenklasse	169
8.1.4	Entity-API-Klasse	169
8.2	Hervorheben von Änderungen als eigenständige Einheit	170
8.2.1	Highlight-Changes-Interface	170
8.2.2	Highlight-Changes-Klasse	171
8.2.3	Berechnungsmethode für Entitätsknoten	172
8.2.4	Berechnungsmethode für Entitätsbäume	174
8.2.5	Verwendung der Highlight-Changes-Klasse	175
8.2.6	Objektorientiertes Programmiermodell	176
8.3	Tests für die Highlight-Changes-Klasse	177
8.3.1	Unit-Tests	178
8.3.2	Komponententests	180
8.4	Zusammenfassung des ersten Teils	182

TEIL II Testorientiertes ABAP-Design

9	Design von Methoden	187
9.1	Regeln zur Implementierung einer Methode	188
9.1.1	Kommentierung	188
9.1.2	Formatierung	190
9.2	Regeln für die Signatur einer Methode	196
9.2.1	Einfaches Beispiel für die Verbesserung einer Methodensignatur	196
9.2.2	Tipps und Tricks für eine kompakte Methodensignatur	197
9.2.3	Anspruchsvolles Beispiel für die Verbesserung einer Methodensignatur	199

10	Design von Klassen	209
10.1	Erzeugung eines Objekts durch seine Klasse	210
10.1.1	Erzeugungsmethoden	210
10.1.2	Testbarkeit des Konstruktors	211
10.2	Erzeugung eines Objekts durch eine Factory-Klasse	213
10.2.1	Erzeugung ohne Factory	213
10.2.2	Erzeugung mit konkreter Factory	214
10.2.3	Erzeugung mit abstrakter Factory	215
10.3	Abhängigkeitsarten zwischen Klassen	217
10.3.1	Innere Abhängigkeit	218
10.3.2	Äußere Abhängigkeit	219
10.4	Interfaces einer Klasse	220
10.4.1	Berechtigungs-spezifische Interfaces	220
10.4.2	Rollenspezifische Interfaces	221
10.4.3	Interface-Prinzipien	222
10.5	Abstraktionsebenen innerhalb einer Klasse	222
10.5.1	Entitätsmengenklasse mit Tabellen	224
10.5.2	Entitätsmengenklasse mit Objekten	226
10.5.3	Effizienz	227
10.6	Katalog-Entwurfsmuster	229
10.7	Innerer Zusammenhalt	231
10.7.1	Komponentengraph	232
10.7.2	Horizontale Zerlegung einer Klasse	233
10.7.3	Vertikale Zerlegung einer Klasse	234
10.7.4	Beispiel für die Zerlegung einer Klasse	235
11	Design von Paketen	237
11.1	Paketkonzept	237
11.1.1	Sichtbarkeit	239
11.1.2	Paketschnittstellen	240
11.1.3	Verwendungserklärungen	242
11.2	Produktpakete	244
11.3	Testpakete	246

12 Testfälle	249
12.1 Testdesign	250
12.1.1 Parameterorientierte Testverfahren	250
12.1.2 Zustandsorientiertes Testverfahren	253
12.2 Testpyramide	253
12.2.1 Weiterentwicklung von Bestandscode	253
12.2.2 Neuentwicklung	254
12.3 Testabdeckung	256
12.3.1 Zuständigkeiten der Testtypen	257
12.3.2 Schrittweise Vervollständigung der Testabdeckung	257
13 Testdoubles	259
13.1 Vorteile von Testdoubles	259
13.1.1 Abhängigkeiten eines Tests	260
13.1.2 Überprüfung des Verhaltens einer Methode	261
13.2 Spezifikation von Testdoubles	263
13.2.1 Testdoubletypen	264
13.2.2 Testdoubles verwenden	268
13.3 Design von Testdoubles	269
13.3.1 Interface- und Unterklassendoubles	270
13.3.2 Proxydoubles	271
13.3.3 Änderungen des Produktdesigns zugunsten der Testbarkeit	273
13.4 Injektion von Testdoubles	275
13.4.1 Von der CUT-Produktklasse angebotene Injektionsmechanismen	275
13.4.2 Von der CUT-Klasse ermöglichte Injektionsmechanismen	277
13.4.3 Von einem DOC-Singleton ermöglichte Injektionsmechanismen	280
13.4.4 Von einer statischen Factory ermöglichte Injektionsmechanismen	283
13.4.5 Von einer Singleton-Factory ermöglichte Injektionsmechanismen	284
13.4.6 Von einer abstrakten Factory ermöglichte Injektionsmechanismen	285

14 Globale Testdoubles	289
14.1 Testdouble für eine DOM-Methode	290
14.1.1 Doubleklasse mit vielen Erzeugungsmethoden	292
14.1.2 Doubleklasse mit einer Erzeugungsmethode	294
14.2 Testdouble für zwei DOM-Methoden	295
14.2.1 Doubleklasse für mehrere DOM-Methoden	295
14.2.2 Definition und Implementierung der Testklasse	298
14.3 Kombination zweier Methodendoubles	301
14.3.1 Starker Zusammenhalt einer Doubleklasse	301
14.3.2 Decorator-Doubleklasse	303
14.4 Globalisierung von Testdoubles	307
14.5 Design globaler Testdoubles	310
14.6 Anpassung globaler Testdoubles	313
14.6.1 Job-Interface	313
14.6.2 Globale Jobdoubleklasse	314
14.6.3 Testklasse für eine Jobverwenderklasse	316
15 Testklassen	319
15.1 Lokale Testklassen	321
15.2 Entwurfsmuster	324
15.2.1 Gegeben-Wenn-Dann-Entwurfsmuster	324
15.2.2 Testklassen-Entwurfsmuster	325
15.3 Testklassenhierarchien	327
15.4 Globale Testklassen	331
15.5 Namenskonventionen für Testcode	333
16 Testdaten	335
16.1 Testdatencontainer	335
16.1.1 Verwendung von Testdatencontainern in ABAP Unit	335
16.1.2 Vor- und Nachteile von Testdatencontainern	336
16.2 Testdatenobjekte	343

17 Testinfrastruktur 355

17.1 Anwendungsszenarien für eine globale Testinfrastruktur	357
17.1.1 Komponententests	357
17.1.2 Integrationstests	359
17.2 Entwicklungsprozesse mithilfe einer Testinfrastruktur	362
17.2.1 Entwicklung einer API für Legacy-Komponenten	362
17.2.2 Teamübergreifende Testinfrastruktur für ein Produkt	364
17.3 Zusammenfassung des zweiten Teils	365

TEIL III Agile Neuentwicklung**18 Planung und Vorarbeit** 369

18.1 Spezifikation der Beispielanwendung	370
18.2 Architektur und Design der Beispielanwendung	372
18.2.1 Paketstruktur	372
18.2.2 Datenmodell	374
18.2.3 Objektmodell	375
18.3 Teststrategie für die Beispielanwendung	377
18.3.1 Akzeptanztests	377
18.3.2 Prozesskomponententests	378
18.3.3 Modellkomponententests	379
18.3.4 Zugriffsintegrationstests	379
18.4 Vorbereitung der testgetriebenen Entwicklung	380
18.4.1 Skelett aller Komponenten	380
18.4.2 Skelett der Modellkomponente	383
18.4.3 Externe Factory eines Pakets	384
18.4.4 Interne Factory eines Pakets	386
18.4.5 API eines Pakets	387
18.4.6 Datenklasse in einem Paket	388
18.4.7 Basisklasse für die Komponententests eines Pakets	392

19 Testgetriebene Entwicklung 393

19.1 Akzeptanztestgetriebene Entwicklung	393
19.1.1 Basisklasse für die Akzeptanztests	394
19.1.2 Akzeptanztest (Gegeben- und Wenn-Phase)	395
19.2 Komponententestgetriebene Entwicklung	396
19.2.1 Komponententestklasse für den Erzeugungsprozess	397
19.2.2 Globales Double für den Vertragsmanager	398
19.2.3 Entwicklungstaktik	400
19.3 Unit-Test-getriebene Entwicklung	401
19.4 Objektbasierte Überprüfung	403
19.4.1 Akzeptanztest (Dann-Phase)	403
19.4.2 Objektbasierter Vergleich	404
19.4.3 Refactoring des Akzeptanztests	406
19.5 Testen des Änderns und Sicherns eines Vertrags	407
19.5.1 Komponententest	410
19.5.2 Integrationstest	412
19.6 Zusammenfassung des dritten Teils	413

TEIL IV Agile Methoden**20 Scrum** 417

20.1 Artefakte	417
20.1.1 Produkt-Backlog	418
20.1.2 Ready-Kriterien	418
20.1.3 Done-Kriterien	419
20.1.4 Sprint-Backlog	421
20.1.5 Sprint-Burndown-Chart	422
20.1.6 Release-Burndown-Chart	422
20.2 Rollen	423
20.2.1 Product Owner	423
20.2.2 Scrum Master	423
20.2.3 Team	424

20.3 Meetings	424
20.3.1 Backlog-Grooming	424
20.3.2 Sprintplanung – Teil 1	425
20.3.3 Sprintplanung – Teil 2	425
20.3.4 Daily Scrum	426
20.3.5 Sprint-Review	426
20.3.6 Sprint-Retrospektive	427
20.3.7 Zyklus	427
20.4 Eigenschaften	429
21 Agile Software Engineering	433
<hr/>	
21.1 Refactoring	434
21.1.1 Durchführung	434
21.1.2 Vor- und Nachteile	434
21.1.3 Abgrenzungen	435
21.2 Testgetriebene Entwicklung	437
21.2.1 Durchführung	437
21.2.2 Vor- und Nachteile	439
21.2.3 Alternativen	440
21.3 Paarweise Programmierung	441
21.3.1 Durchführung	442
21.3.2 Vor- und Nachteile	444
21.3.3 Alternativen	446
21.4 Walking Skeleton	447
21.4.1 Durchführung	447
21.4.2 Vor- und Nachteile	452
21.5 Gemeinsame Codeverantwortung	453
21.5.1 Durchführung	455
21.5.2 Vor- und Nachteile	456
21.5.3 Alternativen	457
21.6 Kontinuierliche Integration	459
21.6.1 Kontinuierliche Integration mit ABAP	459
21.6.2 Quasilokale Änderung	460

22 Lean-Entwicklungsmodell	463
<hr/>	
22.1 Grundlagen	463
22.2 Umsetzung der Lean-Prinzipien mit Agile Software Engineering	465
22.2.1 Refactoring	465
22.2.2 Testgetriebene Entwicklung	465
22.2.3 Paarweise Programmierung	466
22.2.4 Walking Skeleton	466
22.2.5 Gemeinsame Codeverantwortung	467
22.3 Testinfrastruktur	467
22.4 Agile Coaching	468
23 Entwicklung der Teams	469
<hr/>	
23.1 Nachhaltige Ausbildung	469
23.2 Lücken in der Ausbildung	473
23.2.1 Keine Unit-Tests	473
23.2.2 Lange Methoden	474
23.2.3 Ausbildungsplan	474
23.3 Agile Coaching	476
23.3.1 Durchführung	476
23.3.2 Vor- und Nachteile	481
23.4 Netzwerk für Agile Coaching	483
24 Entwicklung des Backlogs	485
<hr/>	
24.1 Design Thinking	486
24.1.1 Verstehen	488
24.1.2 Beobachten	489
24.1.3 Sichtweise definieren	490
24.1.4 Ideen finden	491
24.1.5 Prototypen entwickeln	491
24.1.6 Testen	491
24.2 User Story Mapping	492
24.2.1 Aufbau einer User Story Map	492
24.2.2 Überprüfung einer User Story Map	495

25 Entwicklung des Produkts	497
25.1 Nachhaltige Entwicklung	498
25.1.1 Erhaltung der Produktivität	498
25.1.2 Verbesserung der Produktivität	499
25.2 Entwicklungsstrategien für Legacy Code	501
25.2.1 Charakterisierung	501
25.2.2 Transformation	501
25.3 Entwicklungsstrategien für neuen Code	502
25.4 Zusammenfassung des vierten Teils	504

TEIL V Testorientierte ABAP-Werkzeuge

26 ABAP Unit	507
26.1 Klasse CL_ABAP_UNIT_ASSERT	507
26.1.1 Methoden	508
26.1.2 Parameter	508
26.1.3 Bedingungen	509
26.2 Ausführung von Tests	512
26.2.1 Statische Grundlagen der Testausführung	512
26.2.2 Dynamische Grundlagen der Testausführung	513
26.2.3 Testbeziehungen	516
26.3 Entwicklungsobjekte	518
27 ABAP Development Tools	521
27.1 Einführung	521
27.1.1 SAP-Hilfe	522
27.1.2 Projekt anlegen	523
27.1.3 Produktklasse anlegen	523
27.2 Testgetriebene Entwicklung mit den ABAP Development Tools	525
27.2.1 Berechtigungsprüfung als Test-Stub	526
27.2.2 Berechtigungsprüfung als Test-Spy	534
27.2.3 Berechtigungsprüfung als Mock-Objekt	540

28 ABAP-Werkzeuge zur Testisolierung	545
28.1 Beispielklasse	545
28.2 Open SQL Test Double Framework	548
28.3 Test Seams	551
28.4 ABAP Test Double Framework	553
28.5 Zusammenfassung des fünften Teils und des gesamten Buches	557
Anhang	559
A Namenskonventionen	561
B Literaturverzeichnis	563
C Der Autor	564
Index	565