

Contents at a Glance

1	Introduction	33
PART I Building a Test Infrastructure		
2	The Sample Application	49
3	Code-Based Test Improvement	57
4	Design-Based Test Improvement	87
5	Robust Integration Test	103
6	Minimizing Dependencies	117
7	Isolated Component Test	141
8	Redesign with Unit Tests	169
PART II Test-Oriented ABAP Design		
9	Designing Methods	191
10	Designing Classes	215
11	Designing Packages	245
12	Test Cases	257
13	Test Doubles	267
14	Global Test Doubles	297
15	Test Classes	327
16	Test Data	343
17	Test Infrastructures	363
PART III Agile Development of a New Application		
18	Preparation for Test-Driven Development	377
19	Test-Driven Development	403
PART IV Agile Methodology		
20	Scrum	429
21	Agile Software Engineering	445
22	Lean Development Model	475
23	Team Development	481
24	Backlog Development	497
25	Product Development	509
PART V Test-Oriented ABAP Tools		
26	ABAP Unit	519
27	ABAP Development Tools	535
28	ABAP Tools for Test Isolation	559

Contents

Foreword	23
Preface	25

1	Introduction	33
----------	---------------------	-----------

1.1	Sustainability: Development and Maintenance with a Test Infrastructure	34
1.2	Agility: Practices of Agile Software Engineering	39
1.3	Efficiency: Compliance with Design Principles	40
1.3.1	SOLID Principles	40
1.3.2	FIRST Principles	42
1.4	Effectivity: Communication with Diagrams	44
1.5	Summary	46

PART I Building a Test Infrastructure

2	The Sample Application	49
----------	-------------------------------	-----------

2.1	Master Data Management	50
2.1.1	Change Request	50
2.1.2	Definition of a Master Data Model	51
2.2	User Interface	52
2.3	Backend	54
2.4	Summary	56

3	Code-Based Test Improvement	57
3.1	Introduction to the Test Code	57
3.1.1	Defining a Test Class	58
3.1.2	Implementing a Test Class	59
3.1.3	ABAP Unit Log	66
3.1.4	Diagrams	67
3.2	General Clean Code Principles	70
3.2.1	Improvement Process	70
3.2.2	Improving the Definition of the Test Class	72
3.2.3	Improving the Implementation of the Test Class	73
3.2.4	Diagrams	77
3.3	Test-Oriented Clean Code Principles	79
3.3.1	Improvement Process	79
3.3.2	Improving the Implementation of the Test Class	80
3.3.3	Diagrams	83
3.4	Summary	86

4	Design-Based Test Improvement	87
4.1	Base Classes of Test Classes	87
4.1.1	Improvement Process	88
4.1.2	Extracting a Base Class from a Test Class	88
4.1.3	Diagrams	90
4.2	Usage of Help Classes by Test Classes	93
4.2.1	Improvement Process	93
4.2.2	Design Patterns for Classes	93
4.2.3	Defining and Using the Test Data Class	94
4.2.4	Diagrams	97
4.3	Summary	100

5

Robust Integration Test

103

5.1	Improvement Process	103
5.2	Independence of Test Methods	104
5.2.1	Test Data Container	104
5.2.2	Singleton Design Pattern	106
5.3	Repeatability of Test Methods	108
5.3.1	Automated Test Setup	109
5.3.2	Help Methods for Test Class	110
5.3.3	Diagrams	113
5.4	Summary	116

6

Minimizing Dependencies

117

6.1	Simplification of Use	118
6.1.1	Improvement Process	119
6.1.2	Simplifying the Productive API	119
6.1.3	Access to the Test Data Container	120
6.1.4	Delegating Test Classes	121
6.1.5	Diagrams	122
6.2	Segregation of Use	125
6.2.1	Improvement Process	126
6.2.2	Nested Interfaces	126
6.2.3	Diagrams	127
6.3	Independence of Creations	129
6.3.1	Improvement Process	129
6.3.2	Defining a Factory	129
6.3.3	Implementing a Factory	130
6.3.4	Diagrams	132
6.4	Independence of Extensions	135
6.4.1	Improvement Process	135

6.4.2	Abstract Superclass for All Entities	135
6.4.3	Concrete Subclass for Flight Connections	137
6.5	Summary	139

7 Isolated Component Test 141

7.1	Transforming the Integration Test	142
7.1.1	Improvement Process	143
7.1.2	Isolation with a Governance API Test Double	143
7.1.3	Configuring the Governance API Test Double	148
7.1.4	Diagrams	151
7.2	Scaling with a Test Language	154
7.2.1	Improvement Process	154
7.2.2	State of an Entity	155
7.2.3	Test Method: Unsaved Changes	157
7.2.4	Test Method: Changes to a New Entity Type	159
7.2.5	Test Method: Contained Changes	160
7.3	Test-Driven Development	161
7.3.1	Test Class for Highlighting Deletions	162
7.3.2	Diagrams	163
7.4	Liskov Substitution Principle	165
7.4.1	Violation	165
7.4.2	Compliance	167
7.5	Summary	167

8 Redesign with Unit Tests 169

8.1	Object-Oriented API for Entities	171
8.1.1	Entity Type Class	171
8.1.2	Selection Class	171
8.1.3	Entity Set Class	173
8.1.4	Entity API Class	173

8.2	Highlighting Changes as an Independent Component	174
8.2.1	Highlight Changes Interface	175
8.2.2	Highlight Changes Class	175
8.2.3	Calculation Method for Entity Nodes	176
8.2.4	Calculation Method for Entity Trees	178
8.2.5	Using the Highlight Changes Class	179
8.3	Tests for the Highlight Changes Class	182
8.3.1	Unit Tests	182
8.3.2	Component Tests	185
8.4	Summary	187

PART II Test-Oriented ABAP Design

9 Designing Methods 191

9.1	Rules for Implementing Methods	192
9.1.1	Commenting	192
9.1.2	Formatting	194
9.2	Rules for Method Signatures	200
9.2.1	Simple Example of Improving Method Signatures	200
9.2.2	Compacting Method Signatures	202
9.2.3	Advanced Example of Improving Method Signatures	204
9.3	Summary	213

10 Designing Classes 215

10.1	Creation of an Object by Its Class	216
10.1.1	Creation Methods	216
10.1.2	Testability of the Constructor	217
10.2	Creation of an Object by a Factory Class	219
10.2.1	Creation without a Factory	219

10.2.2	Creation with a Concrete Factory	220
10.2.3	Creation with an Abstract Factory	221
10.3	Types of Dependencies between Classes	222
10.3.1	Internal Dependencies	223
10.3.2	External Dependencies	224
10.4	Interfaces of a Class	225
10.4.1	Authorization-Specific Interfaces	226
10.4.2	Role-Specific Interfaces	227
10.4.3	Interface Principles	228
10.5	Levels of Abstraction within a Class	228
10.5.1	Entity Set Class with Tables	230
10.5.2	Entity Set Class with Objects	232
10.5.3	Efficiency	233
10.6	Catalog Design Pattern	235
10.7	Cohesion	237
10.7.1	Component Graph	238
10.7.2	Horizontal Split of a Class	239
10.7.3	Vertical Split of a Class	239
10.7.4	Example of the Split of a Class	241
10.8	Summary	242

11	Designing Packages	245
-----------	---------------------------------	------------

11.1	Package Concept	245
11.1.1	Visibility	247
11.1.2	Package Interfaces	248
11.1.3	Use Accesses	250
11.2	Product Packages	251
11.3	Test Packages	253
11.3.1	Transport Layers	253
11.3.2	Shortening of Note Chains	255
11.4	Summary	255

12

Test Cases

257

12.1	Test Design	257
12.1.1	Parameter-Oriented Testing	258
12.1.2	State-Oriented Testing	261
12.2	Test Pyramid	261
12.2.1	Further Development of Legacy Code	261
12.2.2	New Development	262
12.3	Test Coverage	264
12.3.1	Responsibilities of the Test Types	265
12.3.2	Completion of Test Coverage	265
12.4	Summary	266

13

Test Doubles

267

13.1	Advantages of Test Doubles	267
13.1.1	Dependencies of a Test	267
13.1.2	Verifying the Behavior of a Method	269
13.2	Specifying Test Doubles	271
13.2.1	Test Double Types	271
13.2.2	Using Test Doubles	275
13.3	Designing Test Doubles	277
13.3.1	Interface and Subclass Doubles	278
13.3.2	Proxy Doubles	279
13.3.3	Changes to the Product Design in Favor of Testability	280
13.4	Injecting Test Doubles	282
13.4.1	Injection Mechanisms Offered by the Class under Test	282
13.4.2	Injection Mechanisms Enabled by the Class under Test	285
13.4.3	Injection Mechanisms Enabled by a Depended-on Singleton Class ...	287
13.4.4	Injection Mechanisms Enabled by a Static Factory	291
13.4.5	Injection Mechanisms Enabled by a Singleton Factory	292
13.4.6	Injection Mechanisms Enabled by an Abstract Factory	293
13.5	Summary	296

14

Global Test Doubles

297

14.1

Test Double for a Method

298

14.1.1

Double Class with Many Creation Methods

300

14.1.2

Double Class with One Creation Method

302

14.2

Test Double for Two Methods

303

14.2.1

Definition and Implementation of the Test Double Class

303

14.2.2

Definition and Implementation of the Test Class

306

14.3

Method Doubles and Their Combinations

309

14.3.1

Cohesion of a Double Class

309

14.3.2

Decorator Double Class

311

14.4

Globalizing Test Doubles

316

14.5

Designing Global Test Doubles

318

14.6

Adapting the Design of Global Test Doubles

320

14.6.1

Job Interface

321

14.6.2

Global Job Double Class

322

14.6.3

Test Class for a Job User Class

324

14.7

Summary

325

15

Test Classes

327

15.1

ABAP Unit Test Framework

327

15.2

Local Test Classes

329

15.3

Design Patterns for Test Classes

332

15.3.1

Given-When-Then Design Pattern

332

15.3.2

Test Class Design Pattern

333

15.4

Test Class Hierarchies

335

15.5

Global Test Classes

339

15.6

Summary

342

16

Test Data

343

16.1	Test Data Container	343
16.1.1	Using Test Data Containers in ABAP Unit	343
16.1.2	Testing with and without Test Data Containers	344
16.2	Test Data Objects	352
16.3	Summary	362

17

Test Infrastructures

363

17.1	Components of Test Infrastructures	363
17.2	Application Scenarios for a Global Test Infrastructure	365
17.2.1	Component Tests	365
17.2.2	Integration Tests	367
17.3	Development Processes Using a Test Infrastructure	370
17.3.1	Developing an API for Legacy Components	370
17.3.2	Cross-Team Test Infrastructure	372
17.4	Summary	373

PART III

Agile Development of a New Application

18

Preparation for Test-Driven Development

377

18.1	Specification of the Sample Application	378
18.2	Architecture and Design of the Sample Application	380
18.2.1	Package Structure	380
18.2.2	Data Model	382
18.2.3	Object Model	383
18.3	Test Strategy for the Sample Application	385
18.3.1	Acceptance Tests	386

18.3.2	Process Component Tests	386
18.3.3	Model Component Tests	387
18.3.4	Access Integration Tests	387
18.4	Skeleton of the Sample Application	388
18.4.1	All Components	388
18.4.2	Model Component	391
18.4.3	External Factory Class	392
18.4.4	Internal Factory Class	394
18.4.5	API Class	395
18.4.6	Test Data Class	397
18.4.7	Base Class for Component Tests	400
18.5	Summary	401

19

Test-Driven Development

403

19.1	Acceptance Test-Driven Development	403
19.1.1	Base Class for the Acceptance Tests	403
19.1.2	Acceptance Test (Given and When Phase)	405
19.2	Component Test-Driven Development	406
19.2.1	Component Test Class for the Creation Process	407
19.2.2	Global Double of the Contract Manager	408
19.2.3	Development Tactics	410
19.3	Unit Test-Driven Development	411
19.4	Completion and Improvement of the First Acceptance Test	414
19.4.1	Acceptance Test (Then Phase)	414
19.4.2	Object-Based Comparison	415
19.4.3	Refactoring of the Acceptance Test	417
19.5	Extension of the Acceptance Test Suite	418
19.5.1	Test Development Strategy	418
19.5.2	Component Test	422
19.5.3	Integration Test	423
19.6	Summary	425

PART IV Agile Methodology

20 Scrum	429
20.1 Artifacts	430
20.1.1 Product Backlog	430
20.1.2 Ready Criteria	431
20.1.3 Done Criteria	432
20.1.4 Sprint Backlog	433
20.1.5 Sprint Burndown Chart	434
20.1.6 Release Burndown Chart	435
20.2 Roles	436
20.2.1 Product Owner	436
20.2.2 Scrum Master	437
20.2.3 Team	437
20.3 Meetings	437
20.3.1 Backlog Grooming	438
20.3.2 Sprint Planning: Part 1	438
20.3.3 Sprint Planning: Part 2	439
20.3.4 Daily Scrum	439
20.3.5 Sprint Review	440
20.3.6 Sprint Retrospective	440
20.3.7 Scrum Cycle	440
20.4 Characteristics	442
20.5 Summary	443

21 Agile Software Engineering	445
21.1 Refactoring	445
21.1.1 Implementation	446
21.1.2 Advantages and Disadvantages	446
21.1.3 Differentiation	447

21.2	Test-Driven Development	449
21.2.1	Implementation	449
21.2.2	Advantages and Disadvantages	451
21.2.3	Alternatives	453
21.3	Pair Programming	453
21.3.1	Implementation	454
21.3.2	Advantages and Disadvantages	456
21.3.3	Alternatives	458
21.4	Walking Skeleton	459
21.4.1	Implementation	460
21.4.2	Advantages and Disadvantages	463
21.5	Shared Code Ownership	465
21.5.1	Implementation	466
21.5.2	Advantages and Disadvantages	468
21.5.3	Alternatives	468
21.6	Continuous Integration	470
21.6.1	Continuous Integration with ABAP	471
21.6.2	Local Change	471
21.7	Summary	473

22 Lean Development Model 475

22.1	Basics	475
22.2	Implementing Lean Principles with Agile Software Engineering	477
22.2.1	Refactoring	477
22.2.2	Test-Driven Development	477
22.2.3	Pair Programming	478
22.2.4	Walking Skeleton	478
22.2.5	Shared Code Ownership	479
22.3	Test Infrastructure	479
22.4	Summary	480

23	Team Development	481
23.1	Sustainable Learning	481
23.2	Learning Gaps	484
23.2.1	No Unit Tests	484
23.2.2	Long Methods	485
23.2.3	Learning Plan	486
23.3	Agile Coaching	487
23.3.1	Implementation	488
23.3.2	Advantages and Disadvantages	492
23.4	Network for Agile Coaching	494
23.5	Summary	495
24	Backlog Development	497
24.1	Design Thinking	498
24.1.1	Understand	500
24.1.2	Observe	501
24.1.3	Define Point of View	502
24.1.4	Ideate	502
24.1.5	Prototype	503
24.1.6	Test	503
24.2	User Story Mapping	503
24.2.1	Structure of a User Story Map	504
24.2.2	Review of a User Story Map	506
24.3	Summary	506
25	Product Development	509
25.1	Sustainable Development	510
25.1.1	Maintaining Productivity	510
25.1.2	Improving Productivity	511

25.2	Development Strategies for Legacy Code	513
25.2.1	Characterization	513
25.2.2	Transformation	513
25.3	Development Strategies for New Code	514
25.4	Summary	516

PART V Test-Oriented ABAP Tools

26 ABAP Unit 519

26.1	CL_ABAP_UNIT_ASSERT Class	519
26.1.1	Methods	520
26.1.2	Parameters	520
26.1.3	Constraints	521
26.2	Execution of Tests	525
26.2.1	Static Basics of Test Execution	525
26.2.2	Dynamic Basics of Test Execution	526
26.2.3	Test Relations	528
26.3	Development Objects	530
26.4	Summary	532

27 ABAP Development Tools 535

27.1	Introduction	535
27.1.1	SAP Help	535
27.1.2	Creating a Project	536
27.1.3	Creating a Product Class	537
27.2	Test-Driven Development with ABAP Development Tools	539
27.2.1	Authority Check as Test Stub	540
27.2.2	Authority Check as Test Spy	548
27.2.3	Authority Check as Mock Object	554
27.3	Summary	557

28	ABAP Tools for Test Isolation	559
-----------	--------------------------------------	-----

28.1	Sample Class	559
28.2	Open SQL Test Double Framework	562
28.3	Test Seams	565
28.4	ABAP Test Double Framework	568
28.5	Summary	571

	Appendices	573
--	-------------------	-----

A	Naming Conventions for ABAP Code	575
A.1	Classes	575
A.2	Storages	576
A.3	Methods	577
B	Bibliography	579
C	The Author	581

	Index	583
--	--------------	-----