

# Inhalt

Einleitung .....	15
------------------	----

## TEIL I ABAP – gestern, heute und morgen

### **1 Qualität, Performance und Sicherheit in der aktuellen Softwareentwicklung ..... 23**

1.1	Motivation .....	24
1.1.1	Dimensionen von Qualität .....	25
1.1.2	Lebenszyklus von Anwendungen .....	27
1.1.3	Ursachen und Auswirkungen von Qualitätsproblemen .....	29
1.1.4	Qualitätsmanagement und Entwicklungsprozess .....	31
1.2	Entwicklung im Kontext von SAP-Anwendungen ....	34
1.2.1	Entwicklung in SAP-Landschaften .....	35
1.2.2	Standardkomponenten .....	38
1.2.3	Softwareaktualisierungen und Plattformänderungen .....	39
1.3	SAP-Produkte im Wandel der Zeit .....	41
1.3.1	Produkte und Releases im Überblick .....	43
1.3.2	Auswirkungen durch Mobile, Cloud und In-Memory .....	45

### **2 Anwendungsentwicklung mit ABAP in der Praxis ..... 49**

2.1	Der ABAP-Anwendungsserver im Überblick .....	50
2.1.1	Serverinfrastruktur .....	50
2.1.2	Entwicklungsinfrastruktur .....	54
2.1.3	Wichtige Elemente der ABAP- Programmierung .....	58
2.2	Modernes ABAP-Anwendungsdesign .....	64
2.2.1	Architektur-Blaupause .....	66
2.2.2	SAP-Investitionen und Umgang mit Legacy-Komponenten .....	72

2.3 Produkte und Serviceangebote in den Bereichen  
Qualität, Performance und Sicherheit ..... 73

**TEIL II Qualität**

**3 ABAP-Codequalität ..... 79**

3.1 Clean Code ..... 80  
 3.1.1 Lesbarkeit ..... 82  
 3.1.2 Wartbarkeit ..... 85  
 3.1.3 Änderbarkeit ..... 89  
 3.1.4 Erweiterbarkeit ..... 93  
 3.1.5 Testbarkeit ..... 99  
 3.2 Best Practices zur Sicherung der Qualität von  
 ABAP-Code ..... 102  
 3.2.1 ABAP Objects ..... 103  
 3.2.2 Interfaces ..... 109  
 3.2.3 Checkpoint-Gruppen ..... 116  
 3.3 Fehlerhandling – gewusst wie! ..... 120  
 3.3.1 Fehlerbehandlung mit einer  
 Ausnahmeklasse ..... 121  
 3.3.2 Protokollierung ..... 126  
 3.4 Legacy-Code ..... 130  
 3.4.1 Wie relevant ist Legacy-Code für  
 die Praxis? ..... 131  
 3.4.2 Softwaresanierung und Refactoring ..... 131

**4 Modultests mit ABAP Unit ..... 135**

4.1 Testbare Software entwickeln ..... 136  
 4.1.1 Was macht testbare Software aus? ..... 136  
 4.1.2 Wann und wie sollten Sie Unit-Tests  
 einsetzen? ..... 137  
 4.1.3 Wie sind Modultests aufgebaut? ..... 138  
 4.1.4 Welche verschiedenen Testtechniken  
 gibt es? ..... 140  
 4.2 ABAP-Unit-Tests erstellen ..... 140  
 4.2.1 Aufbau von ABAP-Unit-Testklassen ..... 141

4.2.2 Testklassen anlegen ..... 142  
 4.2.3 Testmethoden implementieren ..... 143  
 4.2.4 Beispiel ..... 144  
 4.3 ABAP-Unit-Tests durchführen ..... 147  
 4.3.1 Einzelne Objekte prüfen ..... 147  
 4.3.2 Ausführung mit Einplanung ..... 148  
 4.3.3 Ausführung mit Abdeckungsmessung ..... 149  
 4.3.4 ABAP Unit Browser ..... 150  
 4.4 Fortgeschrittene Techniken ..... 152  
 4.4.1 Globale Testklassen ..... 152  
 4.4.2 Parametrisierung von Tests ..... 153  
 4.4.3 Verwendung von Test-Doubles ..... 154  
 4.4.4 Eigene Bedingungen implementieren ..... 155

**5 Werkzeuge für die Qualitätssicherung ..... 157**

5.1 Qualitätssicherung mit dem Code Inspector ..... 157  
 5.1.1 Einsatz des Code Inspectors ..... 159  
 5.1.2 Einzelprüfungen ..... 166  
 5.1.3 Konfiguration des Code Inspectors ..... 170  
 5.2 Qualitätssicherung mit dem ABAP Test Cockpit ..... 173  
 5.2.1 ATC-Prüfungen als Entwickler  
 durchführen ..... 174  
 5.2.2 Nutzung des ABAP Test Cockpits als  
 Qualitätsmanager ..... 177  
 5.2.3 ATC-Befreiungen ..... 181  
 5.3 Qualitätssicherung mit ABAP in Eclipse ..... 183  
 5.3.1 Einführung in die Verwendung von  
 ABAP in Eclipse ..... 184  
 5.3.2 Nützliche Funktionen für Entwickler ..... 188  
 5.3.3 Verwendung von ABAP-Unit-Tests und  
 ATC in Eclipse ..... 193

**6 Prozesse und Methoden ..... 197**

6.1 Agile Vorgehensmodelle ..... 200  
 6.2 Extreme Programming (XP) ..... 205

**TEIL III Performance**

**7 Typische Performanceprobleme und Lösungen ..... 213**

- 7.1 Zugriffe auf die Datenbank ..... 213
  - 7.1.1 Verarbeitung großer Datenmengen ..... 216
  - 7.1.2 Viele Ausführungen ..... 219
  - 7.1.3 Identische Datenbankzugriffe ..... 223
  - 7.1.4 Probleme mit dem Zugriffspfad ..... 226
- 7.2 Zugriffe auf interne Tabellen ..... 231
  - 7.2.1 READ in Schleifen ..... 234
  - 7.2.2 Geschachtelte Schleifen ..... 238
  - 7.2.3 Sortieren in Schleifen ..... 243
  - 7.2.4 COLLECT ..... 245
- 7.3 Zugriffe auf entfernte Systeme ..... 246
- 7.4 Verbuchung ..... 250
- 7.5 Programmdesign ..... 255

**8 Werkzeuge für die Performanceanalyse ..... 259**

- 8.1 Statische Prüfungen ..... 259
  - 8.1.1 Code Inspector (SCI) ..... 260
- 8.2 Dynamische Prüfungen und Traces ..... 263
  - 8.2.1 Laufzeitstatistiken (STAD) ..... 263
  - 8.2.2 Performance Trace (ST05) ..... 269
  - 8.2.3 Laufzeitanalyse mit dem ABAP Trace/  
ABAP Profiler (SAT) ..... 277
  - 8.2.4 Single Activity Trace (ST12) ..... 288
  - 8.2.5 SQL Monitor (SQLM) ..... 291
  - 8.2.6 Laufzeitprüfungs-Monitor (SRTCM) ..... 299
- 8.3 Kombinierte Auswertung statischer und  
dynamischer Prüfungen (SWLT) ..... 301

**9 Durchführung von Performanceanalysen ..... 305**

- 9.1 Systemweite Performanceprobleme ..... 305
- 9.2 Performanceanalyse eines ABAP-Programms ..... 307
  - 9.2.1 Einstieg und Voraussetzungen ..... 307
  - 9.2.2 Überblick und erste Grobanalyse ..... 308

- 9.2.3 Auswertung und Validierung der  
Daten und Traces ..... 310
- 9.2.4 Weitere Analysemöglichkeiten ..... 317

**10 Neue Möglichkeiten zur Performanceoptimierung  
mit SAP HANA und Open SQL 7.4 ..... 319**

- 10.1 SAP HANA im Überblick ..... 319
  - 10.1.1 Grundlagen der HANA-Architektur ..... 319
  - 10.1.2 Column und Row Store ..... 321
  - 10.1.3 Indexe ..... 323
- 10.2 Open-SQL-Programmierung für SAP HANA mit  
SAP NetWeaver 7.4 ..... 324
  - 10.2.1 Goldene Regeln auf SAP HANA ..... 324
  - 10.2.2 Neue performancerelevante  
Open-SQL-Features ..... 327
- 10.3 Code Pushdown ..... 331
  - 10.3.1 Techniken des Code Pushdowns ..... 331
  - 10.3.2 Empfehlungen zum Einsatz ..... 335

**TEIL IV Sicherheit**

**11 Stellenwert von Sicherheitsaspekten in  
ABAP-Programmen ..... 339**

- 11.1 Klassifizierung der verschiedenen  
Sicherheitsrisiken ..... 341
- 11.2 Typische Lösungsansätze ..... 342
  - 11.2.1 Escaping ..... 344
  - 11.2.2 Eingabeprüfung für ganze Zahlen  
(Integer) ..... 345
  - 11.2.3 Whitelist – Definition von Festwerten ..... 345
  - 11.2.4 Output Encoding – Escaping ..... 348
  - 11.2.5 Eingabeprüfung – Spaltennamen ..... 350
  - 11.2.6 Eingabeprüfung – Tabellennamen ..... 350

**12 Schwachstellen und Schutzmaßnahmen ..... 353**

- 12.1 SQL-Injection ..... 353
  - 12.1.1 Beispiel SQL-Injection ..... 353

12.1.2	Open SQL und natives SQL .....	358
12.1.3	Lösungsansätze .....	363
12.2	Unicode-Prüfung .....	366
12.2.1	Beispiel zu Unicode-Prüfung .....	367
12.2.2	Weitere Aspekte .....	369
12.2.3	Lösungsansätze .....	370
12.3	Directory Traversal .....	372
12.3.1	Beispiel für ein Directory Traversal .....	374
12.3.2	Berechtigungsprüfung .....	376
12.3.3	Lösungsansätze .....	377
12.4	Berechtigungsprüfung .....	381
12.4.1	Beispiel zu Berechtigungsprüfung .....	382
12.4.2	Weitere Aspekte .....	382
12.4.3	Lösungsansätze .....	385
12.5	Cross Site Scripting .....	392
12.5.1	Beispiel für Cross Site Scripting .....	393
12.5.2	Weitere mögliche Angriffspunkte .....	394
12.5.3	Lösungsansätze .....	396
12.6	Dynamischer ABAP-Quelltext .....	398
12.6.1	Beispiel zu dynamischem ABAP-Quelltext .....	398
12.6.2	Weitere Aspekte .....	401
12.6.3	Lösungsansätze .....	402
12.7	Remotefähige Funktionsbausteine .....	404
12.7.1	Beispiel für remotefähige Funktionsbausteine .....	404
12.7.2	Dynamische Funktionsbausteinaufrufe und Unified Connectivity .....	408
12.7.3	Lösungsansätze .....	410
12.8	Betriebssystemzugriffe .....	413
12.8.1	Beispiel für Betriebssystemzugriffe .....	413
12.8.2	Weitere Aspekte .....	415
12.8.3	Lösungsansätze .....	415
12.9	Weitere Sicherheitsaspekte bei der ABAP-Entwicklung .....	420
12.9.1	Beispiel zu weiteren Sicherheitsaspekten bei der ABAP-Entwicklung .....	420
12.9.2	Tabellenzugriffe .....	421
12.9.3	Lösungsansätze .....	423

## 13 Sicherheitsprüfungen für den ABAP-Quelltext durchführen ..... 427

13.1	Sicherheitsanalysen aus Sicht des Entwicklers .....	431
13.2	Konfiguration des Code Vulnerability Analyzers .....	436
13.2.1	Definition von eigenen vertrauens- würdigen Datenquellen .....	439
13.2.2	Hinzufügen von eigenen sicherheits- kritischen Funktionsbausteinen .....	440
13.2.3	Hinzufügen von eigenen Prüfungen auf Systemvariablen .....	441
13.3	Sicherheitsanalysen aus Sicht des Qualitäts- managements .....	441
13.4	Integration des Code Vulnerability Analyzers in die Entwicklungslandschaft .....	443

## TEIL V Praxistipps

## 14 Qualitätsaspekte in der Projektplanung berücksichtigen ..... 447

14.1	Qualitätsmanagement .....	448
14.1.1	Organisatorische Voraussetzungen .....	448
14.1.2	Technische Voraussetzungen .....	449
14.1.3	Einführung von Qualitätsprozessen .....	450
14.2	Qualität messen und steuern .....	451
14.2.1	Quantitative Messungen .....	452
14.2.2	Zusätzliche qualitative Messungen .....	453
14.2.3	Qualität steuern mithilfe von Messungen .....	454
14.3	Empfehlungen für unterschiedliche Projekttypen ....	456
14.3.1	Einführungsprojekt .....	457
14.3.2	Upgrade oder Migration .....	458
14.3.3	Optimierungsprojekt .....	459
14.3.4	Projektentwicklung .....	461
14.3.5	Softwareübergabe .....	462

**15 Empfehlungen für den Einsatz der Techniken  
und Werkzeuge ..... 465**

- 15.1 Praxistipps für die Entwicklung ..... 466
  - 15.1.1 Richtlinien und Konventionen ..... 466
  - 15.1.2 ABAP-Programmierung ..... 468
- 15.2 Praxistipps für den Einsatz von Werkzeugen ..... 469
  - 15.2.1 Entwicklungswerkzeuge ..... 470
  - 15.2.2 Nutzung des ABAP Test Cockpits ..... 470
  - 15.2.3 Testen mithilfe von ABAP Unit  
und eCATT ..... 473
  - 15.2.4 Einsatz von Analysewerkzeugen ..... 475
- 15.3 Praxistipps zum Entwicklungsprozess ..... 477
  - 15.3.1 Einsatz von agilen Methoden ..... 477
  - 15.3.2 Planung und Durchführung von Tests ..... 479
  - 15.3.3 Planung und Durchführung von  
Performanceoptimierungen ..... 480
- 15.4 Top-Ten-Empfehlungen für besseres ABAP ..... 481

Die Autoren ..... 485

Index ..... 487