

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Einleitung | 1 |
| 2 | Die Rolle der Software-Architekten | 9 |
| 2.1 | Wer wird als Software-Architekt angesehen? | 9 |
| 2.1.1 | Der Plattformspezialist als Software-Architekt | 10 |
| 2.1.2 | Der Entwurfsspezialist als Software-Architekt | 11 |
| 2.1.3 | Der Stratege als Software-Architekt | 11 |
| 2.2 | Probleme der technologiebezogenen Sichtweisen | 12 |
| 2.2.1 | Der Software-Architekt als Wunderheiler | 13 |
| 2.2.2 | Der Software-Architekt als kleiner König | 13 |
| 2.2.3 | Resultierende Konflikte | 14 |
| 2.2.4 | Die generische Flexibilisierungsfalle | 15 |
| 2.3 | Was sind die Aufgaben von Architekten? | 16 |
| 2.3.1 | Entwurf | 17 |
| 2.3.2 | Planung und Organisation | 18 |
| 2.3.3 | Bauüberwachung | 19 |
| 2.4 | Wer sollte Architekten beauftragen? | 19 |
| 2.4.1 | Die Bauherrenrolle | 20 |
| 2.4.2 | Entwurfsentscheide muss der Bauherr fällen | 21 |
| 2.4.3 | Qualitätssicherung braucht der Bauherr | 21 |
| 2.5 | Architektur studieren | 22 |
| 2.5.1 | Ausbildungsziele für ein Studium | 23 |
| 2.5.2 | Architektur eines Architekturstudiengangs | 24 |
| 2.6 | Fazit | 28 |
| | Teil I: Konstruktion von Architekturen | 31 |
| 3 | Architekturbeschreibung | 33 |
| 3.1 | Standpunkte und Sichten | 33 |
| 3.1.1 | Der ISO-Standard zu Software-Architekturbeschreibungen | 33 |
| 3.1.2 | Vorgehensweise bei der Wahl der Standpunkte | 35 |
| 3.1.3 | Vergleich verschiedener Standpunktmengen | 35 |
| 3.2 | Architekturbeschreibung mit der UML | 37 |

| | | |
|----------|--|-----------|
| 3.2.1 | Architekturbeschreibungs-Standpunkte und die UML | 38 |
| 3.2.2 | Relevante Modellierungselemente | 39 |
| 3.2.3 | Komponentendiagramm | 44 |
| 3.2.4 | Kompositionsstrukturdiagramm | 47 |
| 3.2.5 | Paketdiagramm | 49 |
| 3.2.6 | Kommunikationsdiagramm | 52 |
| 3.2.7 | Deployment-Diagramm | 53 |
| 3.2.8 | Spezifikation nicht funktionaler Eigenschaften | 56 |
| 3.2.9 | Das Metamodell der UML 2 für Architekturelemente | 57 |
| 3.3 | Architekturbeschreibungssprachen | 61 |
| 3.3.1 | Darwin | 62 |
| 3.3.2 | MetaH/ControlH | 63 |
| 3.3.3 | Rapide | 64 |
| 3.3.4 | Wright | 65 |
| 3.3.5 | Acme | 66 |
| 3.3.6 | xADL | 67 |
| 4 | Architektur- und Komponentenentwicklung | 69 |
| 4.1 | Charakter des Prozesses | 69 |
| 4.2 | Aktivitäten innerhalb einer Iteration | 70 |
| 4.3 | Entwurfsprinzipien | 76 |
| 4.3.1 | Abstraktion | 76 |
| 4.3.2 | Modularisierung | 77 |
| 4.3.3 | Kapselung | 77 |
| 4.3.4 | Hierarchische Dekomposition | 78 |
| 4.3.5 | Separation of Concerns | 79 |
| 4.3.6 | Einheitlichkeit | 80 |
| 4.4 | Entwicklungstätigkeiten | 81 |
| 4.5 | Entwurfsentscheidungen | 85 |
| 4.6 | Arbeitsmittelvorrat des Entwicklers | 90 |
| 5 | Modellgetriebene Software-Entwicklung | 93 |
| 5.1 | Begriffserklärungen | 94 |
| 5.2 | Transformation von Modellen | 97 |
| 5.2.1 | Klassifikation von Modelltransformationen | 99 |
| 5.2.2 | Praxisbeispiel: Das Graphical Modeling Framework | 104 |
| 5.2.3 | Transformationen in der Praxis | 106 |
| 5.3 | MDSO im Entwicklungsprozess | 107 |
| 5.3.1 | Modellgetriebene Entwicklung in Teams | 109 |
| 5.3.2 | Qualitätssicherung bei Modelltransformationen | 109 |
| 5.3.3 | Generierung zusätzlicher Artefakte | 110 |
| 5.3.4 | Mix von Generat und manuellem Code | 110 |
| 5.4 | Vor- und Nachteile | 111 |

| | | |
|----------|---|------------|
| 5.4.1 | Nutzen und Potenziale | 111 |
| 5.4.2 | Trade-offs und Herausforderungen | 113 |
| 5.4.3 | Alternativen und ergänzende Ansätze | 115 |
| 5.5 | Industrielle Reife und Einsetzbarkeit | 118 |
| 5.5.1 | Standards | 118 |
| 5.5.2 | Werkzeuge | 119 |
| 5.5.3 | Cartridges | 120 |
| 5.5.4 | Werkzeugauswahl | 121 |
| 5.6 | Fazit | 122 |
| 6 | Entwurf serviceorientierter Architekturen | 123 |
| 6.1 | Motivation und Einführung | 124 |
| 6.1.1 | Motivation | 124 |
| 6.1.2 | Geschäftsarchitektur und Anwendungslandschaft | 125 |
| 6.2 | SOA-Konzepte | 126 |
| 6.2.1 | Serviceorientiertes Business-IT-Alignment | 128 |
| 6.2.2 | Servicekonzept | 128 |
| 6.2.3 | Referenzarchitektur | 133 |
| 6.2.4 | Geschäftsprozessmanagement | 135 |
| 6.2.5 | Infrastruktur | 136 |
| 6.3 | SOA als Mittel zur Beherrschung von Komplexität | 137 |
| 6.3.1 | Einordnung von SOA | 137 |
| 6.3.2 | Treiber, Einsatzziele und Nutzeffekte | 139 |
| 6.4 | Einführung und Umsetzung einer SOA | 140 |
| 6.4.1 | Ziele von Methoden für die SOA | 140 |
| 6.4.2 | Grundlegende Anforderungen an Methoden für SOA | 142 |
| 6.4.3 | Generelles Projektvorgehen | 143 |
| 6.4.4 | Eine SOA-Methode | 144 |
| 6.5 | Fazit | 149 |
| 7 | Anwendungslandschaften serviceorientiert gestalten | 151 |
| 7.1 | Evolution von Anwendungslandschaften | 152 |
| 7.2 | Serviceorientierte Anwendungslandschaften | 154 |
| 7.2.1 | Serviceorientierung im Geschäft | 154 |
| 7.2.2 | Serviceorientierung in der IT | 156 |
| 7.3 | Fallbeispiel: Christoph Kolumbus Reisen AG | 157 |
| 7.4 | Geschäftsarchitektur und Anwendungslandschaften | 158 |
| 7.4.1 | Methoden im Überblick | 158 |
| 7.4.2 | Identifikation und Verfeinerung von Geschäftsservices | 158 |
| 7.4.3 | Entwurf von Domänen | 160 |
| 7.4.4 | Entwurf von Komponenten | 163 |
| 7.4.5 | Entwurf von Schnittstellen und Operationen | 166 |
| 7.4.6 | Gestaltung der Kopplungsarchitektur | 167 |

| | | |
|-------|--|-----|
| 7.5 | Integration und Plattformen | 169 |
| 7.5.1 | Integrationsarchitekturen | 170 |
| 7.5.2 | Integrationsplattformen und ihre Referenzarchitektur | 171 |
| 7.6 | Systematische Evolutionsplanung | 173 |
| 7.6.1 | Erhebung der Ist-Anwendungslandschaft | 175 |
| 7.6.2 | Bewertung der Ist-Anwendungslandschaft | 175 |
| 7.6.3 | Bestimmung von Hauptszenarien | 176 |
| 7.6.4 | Bestimmung der Soll-Anwendungslandschaft | 176 |
| 7.6.5 | Bestimmung der Roadmap | 177 |

Teil II: Evolution von Architekturen 179

| | | |
|-----------|--|------------|
| 8 | Grundlagen der Evolution von Software-Architekturen | 181 |
| 8.1 | Grundlegende Motivation zur Evolution | 182 |
| 8.1.1 | Methode | 183 |
| 8.1.2 | Anwendbarkeit der Evolution | 185 |
| 8.1.3 | Domänenspezifische Sprachen | 186 |
| 8.2 | Ansätze und Konzepte zur Software-Evolution | 187 |
| 8.3 | Refactoring: Evolution im Kleinen | 188 |
| 8.4 | Software-Evolution am Beispiel | 191 |
| 8.4.1 | Validierung von Refactorings durch Testen | 195 |
| 8.5 | Fazit | 197 |
| 9 | Reverse Engineering | 199 |
| 9.1 | Reverse Engineering ist zielgetrieben | 200 |
| 9.2 | Wissensbasiertes Reverse Engineering | 201 |
| 9.3 | Reverse Engineering von Entwurfsmustern | 202 |
| 9.4 | Abhängigkeitsbasiertes Reverse Engineering | 203 |
| 9.4.1 | Die Rolle von Metriken im Reverse-Engineering-Prozess | 204 |
| 9.4.2 | Programmabhängigkeitsgraphen | 205 |
| 9.4.3 | Programmschneiden | 205 |
| 9.5 | Reverse-Engineering-Prozess | 206 |
| 9.5.1 | Automatisches Reverse Engineering | 206 |
| 9.5.2 | Halbautomatisches Reverse Engineering | 208 |
| 9.6 | Agile Entwicklungsprozesse | 209 |
| 9.7 | Fazit | 210 |
| 10 | Migration von Altsystemen | 213 |
| 10.1 | Aspekte der sanften Migration | 214 |
| 10.1.1 | Migrationsstrategien | 214 |
| 10.1.2 | Gründe für die sanfte Migration | 215 |
| 10.2 | Das Dublo-Muster: DUal Business LOGic | 216 |

| | | |
|--------|---|-----|
| 10.2.1 | Problem und Kontext | 216 |
| 10.2.2 | Lösung | 216 |
| 10.2.3 | Anmerkungen und Einschränkungen | 219 |
| 10.3 | Erreichen einer serviceorientierten Zielarchitektur | 220 |
| 10.4 | Modellgetriebene Migration | 221 |
| 10.5 | Fazit | 222 |

Teil III: Management von Architekturen **223**

| | | |
|-----------|--|------------|
| 11 | Integratives IT-Architekturmanagement | 225 |
| 11.1 | Situation im Unternehmen | 226 |
| 11.2 | Modellbasiertes Architekturmanagement | 227 |
| 11.3 | Analyse der Anwendungsdomäne | 229 |
| 11.3.1 | Typische Fragen an die konzeptuelle IT-Architektur | 229 |
| 11.3.2 | Anwenderrollen einer konzeptuellen IT-Architektur | 230 |
| 11.3.3 | Defizitanalyse konzeptueller IT-Architekturen | 231 |
| 11.4 | Integratives IT-Architekturmanagement | 233 |
| 11.4.1 | Anforderungen an die konzeptuelle IT-Architektur | 233 |
| 11.4.2 | Lösungsansätze zur Modellstrategie | 234 |
| 11.4.3 | Metadaten und Modellmanagement | 236 |
| 11.5 | Vorgehensmodell | 238 |
| 11.5.1 | Lebenszyklus einer konzeptuellen IT-Architektur | 238 |
| 11.5.2 | Integration in vorhandene Strukturen und Modelle | 240 |
| 11.6 | Kriterien zur Werkzeugunterstützung | 242 |
| 11.6.1 | Verzahnung des Repositoriums mit der Umwelt | 242 |
| 11.6.2 | Unterstützung des Lebenszyklus | 244 |
| 11.6.3 | Weitere technische Aspekte | 246 |
| 12 | Management der Unternehmensarchitektur | 249 |
| 12.1 | Unternehmensarchitektur | 250 |
| 12.1.1 | Ziele der Unternehmensarchitektur | 250 |
| 12.1.2 | Ebenen der Unternehmensarchitektur | 251 |
| 12.1.3 | Ebenen und Zielkonflikte: Beitrag der Architektur zur Agilität | 254 |
| 12.2 | Management der Unternehmensarchitektur | 258 |
| 12.2.1 | Prozesse | 258 |
| 12.2.2 | Institutionalisierung, Organisationsstrukturen und Governance | 264 |

Teil IV: Bewertung von Architekturen **267**

| | | |
|-----------|--|------------|
| 13 | Bewertungstechniken – eine allgemeine Übersicht | 269 |
| 13.1 | Begriffsdefinitionen | 270 |
| 13.1.1 | Qualität | 270 |

| | | |
|---|---|------------|
| 13.1.2 | Ausgewählte Qualitätsmerkmale | 271 |
| 13.2 | Grundlagen der Architekturbewertung | 273 |
| 13.2.1 | Ziel der Architekturbewertung | 274 |
| 13.2.2 | Vorteile der Architekturbewertung | 274 |
| 13.2.3 | Probleme der Architekturbewertung | 275 |
| 13.2.4 | Eine Kategorisierung von Architekturbewertungstechniken | 276 |
| 13.3 | Geeignete Architekturbewertungstechniken | 277 |
| 13.3.1 | Evaluationstechniken vs. Qualitätseigenschaft | 278 |
| 13.3.2 | SAAM (Scenario-based Architecture Analysis Method) | 279 |
| 13.3.3 | ATAM (Architecture Trade-off Analysis Method) | 281 |
| 13.3.4 | CBAM (Cost Benefit Analysis Method) | 282 |
| 13.3.5 | ALMA (Architecture-Level Modifiability Analysis) | 285 |
| 13.4 | Fazit | 286 |
| 14 | Bewertungstechniken für die Systemsicherheit | 287 |
| 14.1 | Vorgehensmodell der Gefährdungsanalyse | 289 |
| 14.2 | Eine Klassifizierung von Gefährdungsanalysetechniken | 290 |
| 14.3 | Gefährdungsanalysetechniken | 291 |
| 14.3.1 | Fehlerbaumanalyse und Komponentenfehlerbäume | 291 |
| 14.3.2 | Zuverlässigkeits-Blockdiagramme | 294 |
| 14.3.3 | Markov-Analysen | 294 |
| 14.3.4 | HAZOP und HAZOP-basierte Techniken | 296 |
| 14.3.5 | FMEA und IF-FMEA | 298 |
| 14.3.6 | HiP-HOPS | 300 |
| 14.4 | Fazit | 302 |
| 15 | Bewertungstechniken für die Performance | 303 |
| 15.1 | Allgemeines Vorgehensmodell | 304 |
| 15.2 | Klassifikation von Performance-Bewertungstechniken | 306 |
| 15.2.1 | Architekturbeschreibung | 306 |
| 15.2.2 | Performance-Modelle | 308 |
| 15.2.3 | Auswertungsmethoden | 310 |
| 15.3 | Ausgewählte Verfahren | 311 |
| 15.3.1 | SPE | 311 |
| 15.3.2 | CB-SPE | 312 |
| 15.3.3 | umPSI | 314 |
| 15.3.4 | LQN | 315 |
| 15.3.5 | CP | 316 |
| Teil V: Wiederverwendung von Architekturen | | 319 |
| 16 | Software-Muster | 321 |
| 16.1 | Was ist ein Muster | 323 |

| | | |
|-----------|---|------------|
| 16.2 | Vorteile von Mustern | 330 |
| 16.3 | Musterkategorien | 331 |
| 16.3.1 | Architekturmuster | 331 |
| 16.3.2 | Entwurfsmuster | 332 |
| 16.3.3 | Idiome | 332 |
| 16.4 | Klassifikation von Mustern | 333 |
| 16.5 | Musterkataloge, Mustersysteme und Mustersprachen | 334 |
| 16.6 | Best-Practice-Mustersysteme | 335 |
| 16.7 | Muster-Enzyklopädien | 337 |
| 16.8 | Beziehungen zwischen Mustern | 337 |
| 16.9 | Instanziierung von Mustern | 338 |
| 16.10 | Pattern-Mining | 339 |
| 16.11 | Muster und andere Methoden | 340 |
| 16.12 | Weiterführende Hinweise | 341 |
| 16.13 | Fazit | 344 |
| 17 | Referenzarchitekturen | 345 |
| 17.1 | Typen von Referenzarchitekturen | 346 |
| 17.1.1 | Funktionale Referenzarchitekturen | 347 |
| 17.1.2 | Logische Referenzarchitekturen | 348 |
| 17.1.3 | Technische Referenzarchitekturen | 349 |
| 17.2 | Beschreibung von Referenzarchitekturen | 350 |
| 17.2.1 | Architekturüberblick | 351 |
| 17.2.2 | Komponentenstrukturen | 351 |
| 17.2.3 | Querschnittskonzepte und Prinzipien | 352 |
| 17.2.4 | Referenzschnittstellen | 353 |
| 17.2.5 | Infrastrukturen | 353 |
| 17.3 | Nutzung von Referenzarchitekturen | 355 |
| 17.3.1 | Ausgangspunkt der Software-Entwicklung | 355 |
| 17.3.2 | Grundlage für modellgetriebene Software-Entwicklung | 355 |
| 17.3.3 | Einheitliche Sprache im Unternehmen | 355 |
| 17.3.4 | Betrieb und Integration | 356 |
| 17.4 | Entwicklung, Pflege und Weiterentwicklung | 357 |
| 17.4.1 | Entwicklung als Reifungsprozess | 357 |
| 17.4.2 | Pflege und Weiterentwicklung als Produkt | 357 |
| 18 | Software-Produktlinien | 359 |
| 18.1 | Entwicklung von Software-Produktlinien | 360 |
| 18.2 | Variabilität | 363 |
| 18.2.1 | Variabilität des Produktraums | 364 |
| 18.2.2 | Variabilität der Software-Produktlinien-Artefakte | 365 |
| 18.2.3 | Variabilitätsarten | 366 |

| | | |
|---|--|------------|
| 18.2.4 | Auswirkungen der Variabilitätsarten auf die Software-Produktlinien-Architektur | 367 |
| 18.2.5 | Orthogonale Dokumentation der Variabilität | 370 |
| 18.3 | Evolution von Software-Produktlinien | 371 |
| 18.3.1 | Reifegrade von Software-Produktlinien | 371 |
| 18.3.2 | Evolution der Software-Produktlinien-Artefakte | 374 |
| 18.4 | Vorgehensmodelle | 378 |
| 18.4.1 | Proaktive Entwicklungsstrategien | 378 |
| 18.4.2 | Reaktive Entwicklungsstrategien | 379 |
| 18.5 | Abgrenzung zu anderen Konzepten | 380 |
| 18.6 | Fazit | 381 |
| 19 | Framework-Entwurf | 383 |
| 19.1 | Eigenschaften von Frameworks | 384 |
| 19.1.1 | Umkehrung des Kontrollflusses | 384 |
| 19.1.2 | Vorgabe einer konkreten Anwendungsarchitektur | 385 |
| 19.1.3 | Anpassbarkeit durch Variationspunkte | 385 |
| 19.2 | Arten von Frameworks | 386 |
| 19.2.1 | Objektorientierte Frameworks | 386 |
| 19.2.2 | Komponentenbasierte Frameworks | 389 |
| 19.2.3 | Mischformen von Frameworks | 390 |
| 19.2.4 | Abgrenzung von Frameworks zu anderen Konzepten | 391 |
| 19.3 | Der Entwicklungsprozess von Frameworks | 392 |
| 19.3.1 | Entwicklung von Frameworks | 392 |
| 19.3.2 | Nutzung von Frameworks | 393 |
| 19.3.3 | Komposition von Frameworks | 394 |
| 19.3.4 | Evolution und Wartung von Frameworks | 394 |
| 19.4 | Entwurf objektorientierter Frameworks | 394 |
| 19.4.1 | Der Hotspot-getriebene Entwurf von objektorientierten Frameworks | 395 |
| 19.4.2 | Definition eines speziellen Objektmodells | 395 |
| 19.4.3 | Identifikation der Hotspots und Erstellung der Hotspot-Karten | 397 |
| 19.4.4 | Entwurf und Überarbeitung des objektorientierten Frameworks mit Metamustern | 399 |
| 19.4.5 | Einsatz des objektorientierten Frameworks | 403 |
| 19.4.6 | Andere Entwurfsmethoden für objektorientierte Frameworks | 403 |
| 19.5 | Entwurf von Komponenten-Frameworks | 404 |
| 19.6 | Fazit | 405 |
| Teil VI: Beispiele von Architekturen | | 407 |
| 20 | Multimedia-Architekturen | 409 |
| 20.1 | Aspekte von Multimedia | 410 |

| | | |
|-----------|--|------------|
| 20.2 | Beispiele für Multimedia-Architekturen | 411 |
| 20.2.1 | Beispiele für (Multi-)Media-Datenbanken | 411 |
| 20.2.2 | Beispiele für Streaming-Architekturen | 413 |
| 20.2.3 | Beispiele für Präsentationsarchitekturen | 415 |
| 20.3 | Hintergrund zur Entwicklung von MM4U | 417 |
| 20.4 | Vorgehensweise bei der Entwicklung von MM4U | 419 |
| 20.5 | Entwicklung des MM4U-Frameworks | 422 |
| 20.5.1 | Analyse verwandter Arbeiten und spezielle Objektmodelle | 422 |
| 20.5.2 | Identifizierung und Erstellung der Hotspot-Karten | 422 |
| 20.5.3 | Identifikation der Komponenten und Entwurf der Architektur | 423 |
| 20.5.4 | Entwurf der Multimedia-Kompositionskomponente | 424 |
| 20.5.5 | Entwurf der Präsentationsformat-Generatorenkomponente | 428 |
| 20.5.6 | Implementierung, Überarbeitung und Nutzung des MM4U-Frameworks | 429 |
| 20.6 | Fazit | 431 |
| 21 | Peer-to-Peer-Architekturen | 433 |
| 21.1 | Definitionen | 433 |
| 21.2 | Klassifikation von P2P-Architekturen | 435 |
| 21.2.1 | Reine P2P-Architekturen | 435 |
| 21.2.2 | Hybride P2P-Architekturen | 437 |
| 21.3 | Schichten einer P2P-Applikation | 439 |
| 21.3.1 | Benutzungsschnittstellen | 439 |
| 21.3.2 | P2P-Dienste | 440 |
| 21.3.3 | P2P-Netzwerk | 440 |
| 21.4 | Beispiele von P2P-Architekturen | 441 |
| 21.4.1 | Napster | 441 |
| 21.4.2 | Freenet | 442 |
| 21.4.3 | Chord | 445 |
| 21.4.4 | FastTrack | 447 |
| 21.4.5 | JXTA | 448 |
| 22 | Grid-Architekturen | 451 |
| 22.1 | Definitionen | 452 |
| 22.1.1 | Grid | 452 |
| 22.1.2 | Virtuelle Organisationen | 453 |
| 22.2 | Standardisierung | 454 |
| 22.3 | Klassifikation von Grid-Architekturen | 455 |
| 22.3.1 | Compute Grid | 458 |
| 22.3.2 | Data Grid | 459 |
| 22.4 | Sicherheit | 460 |
| 22.4.1 | Authentifizierung | 461 |
| 22.4.2 | Single-Sign-on und Delegation von Rechten | 462 |

| | | |
|-----------|--|------------|
| 22.4.3 | Autorisierung | 463 |
| 22.5 | Beispiele großer Grid-Projekte | 465 |
| 22.5.1 | D-Grid | 465 |
| 22.5.2 | LHC Computing Grid | 466 |
| 23 | Serviceorientierte Architekturen | 467 |
| 23.1 | DEMS | 468 |
| 23.1.1 | Zielarchitektur DEMS-SOA | 469 |
| 23.1.2 | Information Model CIM | 470 |
| 23.1.3 | Serviceidentifikation und -entwicklung | 471 |
| 23.1.4 | ESB als Laufzeitumgebung | 474 |
| 23.1.5 | Service-Repository und SOA-Management | 476 |
| 23.2 | SOA-Anwendungsszenario und Servicemanagement | 477 |
| 23.2.1 | Dienstgüte als Grundlage für das Servicemanagement | 477 |
| 23.2.2 | SOA-Management-Referenzarchitektur | 478 |
| 23.2.3 | WSQoSX: Prototypische Implementierung eines SOA-Management-Systems | 482 |
| 23.3 | Fazit | 485 |
| 24 | Java-Enterprise-Architektur | 487 |
| 24.1 | Projektkontext | 488 |
| 24.2 | Architektur und Migrationsprozess | 488 |
| 24.2.1 | Vorhandene 4GL-Architektur | 488 |
| 24.2.2 | Technologieauswahl | 489 |
| 24.2.3 | Prozess der Architekturauswahl | 489 |
| 24.3 | Nicht-technische Aspekte sanfter Migration | 492 |
| 24.3.1 | Betriebswirtschaftliche Aspekte sanfter Migration | 492 |
| 24.3.2 | Organisatorische Aspekte sanfter Migration | 496 |
| 24.3.3 | Psychologische Aspekte sanfter Migration | 496 |
| 24.4 | Fazit | 497 |
| | Autorenverzeichnis | 499 |
| | Literatur | 513 |
| | Index | 551 |