

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Einführung | 1 |
| 1.1 | Aufbruch in das ubiquitäre Computerzeitalter | 1 |
| 1.2 | Was ist Software-Qualität? | 6 |
| 1.3 | Warum ist Software eigentlich so schlecht? | 12 |
| 1.4 | Gibt es Licht am Ende des Tunnels? | 19 |
| 1.4.1 | Produktqualität | 19 |
| 1.4.2 | Prozessqualität | 25 |
| 2 | Software-Fehler | 27 |
| 2.1 | Lexikalische und syntaktische Fehlerquellen | 27 |
| 2.2 | Semantische Fehlerquellen | 36 |
| 2.3 | Parallelität als Fehlerquelle | 41 |
| 2.4 | Numerische Fehlerquellen | 44 |
| 2.5 | Portabilitätsfehler | 46 |
| 2.6 | Optimierungsfehler | 49 |
| 2.7 | Von tickenden Zeitbomben | 51 |
| 2.8 | Spezifikationsfehler | 55 |
| 2.9 | Nicht immer ist die Software schuld | 58 |
| 2.10 | Fehlerbewertung | 61 |
| 3 | Konstruktive Qualitätssicherung | 65 |
| 3.1 | Software-Richtlinien | 65 |
| 3.1.1 | Notationskonventionen | 66 |
| 3.1.2 | Sprachkonventionen | 73 |
| 3.2 | Typisierung | 76 |
| 3.2.1 | Typsysteme | 77 |
| 3.2.2 | Grenzen der Typisierung | 81 |
| 3.3 | Vertragsbasierte Programmierung | 93 |
| 3.3.1 | Vor- und Nachbedingungen | 93 |
| 3.3.2 | Invarianten | 95 |
| 3.3.3 | Zusicherungen | 96 |

| | | |
|----------|--|------------|
| 3.4 | Fehlertolerante Programmierung | 98 |
| 3.4.1 | Software-Redundanz | 98 |
| 3.4.2 | Selbstüberwachende Systeme | 101 |
| 3.4.3 | Ausnahmebehandlung | 105 |
| 3.5 | Portabilität | 107 |
| 3.5.1 | Portabilität auf Implementierungsebene | 110 |
| 3.5.2 | Portabilität auf Sprachebene | 131 |
| 3.5.3 | Portabilität auf Systemebene | 134 |
| 3.6 | Dokumentation | 141 |
| 3.6.1 | Spezifikationsdokumente | 142 |
| 3.6.2 | Implementierungsdokumente | 149 |
| 4 | Software-Test | 157 |
| 4.1 | Motivation | 157 |
| 4.2 | Testklassifikation | 158 |
| 4.2.1 | Prüfebenen | 159 |
| 4.2.2 | Prüfkriterien | 170 |
| 4.2.3 | Prüftechniken | 173 |
| 4.3 | Black-Box-Testtechniken | 175 |
| 4.3.1 | Äquivalenzklassentest | 175 |
| 4.3.2 | Grenzwertbetrachtung | 180 |
| 4.3.3 | Zustandsbasierter Software-Test | 183 |
| 4.3.4 | Use-Case-Test | 186 |
| 4.3.5 | Entscheidungstabellenbasierter Test | 190 |
| 4.3.6 | Paarweises Testen | 192 |
| 4.3.7 | Diversifizierende Verfahren | 198 |
| 4.4 | White-Box-Testtechniken | 200 |
| 4.4.1 | Kontrollflussmodellierung | 202 |
| 4.4.2 | Anweisungsüberdeckung | 206 |
| 4.4.3 | Zweigüberdeckung | 209 |
| 4.4.4 | Pfadüberdeckung | 210 |
| 4.4.5 | Bedingungsüberdeckung | 214 |
| 4.4.6 | McCabe-Überdeckung | 216 |
| 4.4.7 | Defs-Uses-Überdeckung | 220 |
| 4.4.8 | Required- <i>k</i> -Tupel-Überdeckung | 227 |
| 4.5 | Testmetriken | 231 |
| 4.5.1 | Überdeckungsmetriken | 231 |
| 4.5.2 | Mutationstest | 238 |
| 4.6 | Grenzen des Software-Tests | 243 |
| 5 | Statische Code-Analyse | 247 |
| 5.1 | Software-Metriken | 247 |
| 5.1.1 | LOC und NCSS | 249 |
| 5.1.2 | Halstead-Metriken | 251 |
| 5.1.3 | McCabe-Metrik | 259 |

| | | |
|-------|---|-----|
| 5.1.4 | Objektorientierte Metriken | 262 |
| 5.1.5 | Visualisierung von Messwerten | 267 |
| 5.2 | Konformitätsanalyse | 271 |
| 5.2.1 | Syntax-Analyse | 272 |
| 5.2.2 | Semantik-Analyse | 281 |
| 5.3 | Exploit-Analyse | 300 |
| 5.3.1 | Buffer Overflows | 301 |
| 5.3.2 | Gegenmaßnahmen | 306 |
| 5.4 | Anomalienanalyse | 313 |
| 5.4.1 | Kontrollflussanomalien | 313 |
| 5.4.2 | Datenflussanomalien | 315 |
| 5.5 | Manuelle Software-Prüfung | 321 |
| 5.5.1 | Walkthroughs | 323 |
| 5.5.2 | Reviews | 324 |
| 5.5.3 | Inspektionen | 327 |
| 6 | Software-Verifikation | 333 |
| 6.1 | Motivation | 333 |
| 6.2 | Deduktion | 338 |
| 6.2.1 | Vor- und Nachbedingungen | 338 |
| 6.2.2 | Das Hoare-Kalkül | 342 |
| 6.3 | Modellprüfung | 350 |
| 6.3.1 | Temporallogik | 352 |
| 6.3.2 | Verifikation temporaler Eigenschaften | 356 |
| 6.4 | Abstrakte Interpretation | 361 |
| 6.4.1 | Fixpunktiteration nach Floyd, Park und Clarke | 362 |
| 6.4.2 | Datenabstraktion | 367 |
| 7 | Software-Lebenszyklus | 371 |
| 7.1 | Wenn Software altern | 371 |
| 7.2 | Gründe der Software-Alterung | 373 |
| 7.2.1 | Bewegliche Ziele | 373 |
| 7.2.2 | Auch Software setzt an | 380 |
| 7.2.3 | Kaschieren statt Reparieren | 385 |
| 7.2.4 | Rückwärtskompatibilität | 389 |
| 7.2.5 | Wissen ist flüchtig | 391 |
| 7.3 | Ist die Software-Alterung unumgänglich? | 395 |
| 7.3.1 | Refactoring | 396 |
| 7.3.2 | Redesign | 406 |
| 8 | Software-Infrastruktur | 415 |
| 8.1 | Versionsverwaltung | 417 |
| 8.1.1 | Anforderungen und Konzeption | 419 |
| 8.1.2 | Revisionen | 423 |
| 8.1.3 | Entwicklung in großen Teams | 428 |

| | | |
|---------------------------------------|---|------------|
| 8.1.4 | Algorithmische Grundlagen der Versionskontrolle | 443 |
| 8.2 | Build-Automatisierung | 450 |
| 8.2.1 | Inkrementelle Compilierung | 450 |
| 8.2.2 | Verteilte Compilierung | 463 |
| 8.3 | Testautomatisierung | 468 |
| 8.3.1 | Regressionstests | 468 |
| 8.3.2 | Oberflächentests | 472 |
| 8.4 | Defektmanagement | 477 |
| 8.4.1 | Fehlerdatenbanken | 477 |
| 8.4.2 | Crash Reports | 485 |
| 9 | Managementprozesse | 491 |
| 9.1 | Vorgehensmodelle | 493 |
| 9.1.1 | Wasserfallmodell | 493 |
| 9.1.2 | V-Modell | 496 |
| 9.1.3 | Rational Unified Process | 502 |
| 9.1.4 | Extreme Programming | 506 |
| 9.2 | Reifegradmodelle | 514 |
| 9.2.1 | Historische Entwicklung | 515 |
| 9.2.2 | CMM | 519 |
| 9.2.3 | CMMI | 530 |
| 9.2.4 | ISO 15504 (SPICE) | 535 |
| 9.2.5 | Bewertung und Kritik | 540 |
| Literaturverzeichnis | 547 | |
| Sachverzeichnis | 557 | |