

Inhaltsverzeichnis

1. Einführung	1
1.1. Aufbau dieses Kurses	1
1.2. Schreibweisen und Konventionen	2
1.3. Einsatz der Werkzeuge unter UNIX und LINUX	3
1.4. Einsatz der Werkzeuge unter WINDOWS	3
1.5. Literatur	3
I. Versionskontrolle	5
2. Änderungsverwaltung mit DIFF und PATCH	7
2.1. Einführung	7
2.2. Unterschiede erkennen mit DIFF	8
2.3. Änderungen anwenden mit PATCH	11
2.4. Änderungen manuell integrieren	13
2.5. Änderungen automatisch integrieren	16
2.6. Syntaktische und semantische Integration	19
2.7. Wie DIFF arbeitet	20
3. Revisionsverwaltung mit RCS	27
3.1. Einführung	27
3.2. Revisionen speichern und wiederherstellen	28
3.3. Äste und Varianten	30
3.4. Sperren	32
3.5. Änderungen vollziehen	33
3.6. Änderungsprotokolle und Schlüsselwörter	34
3.7. Wie RCS Revisionen ablegt	36
3.8. Neuerungen und Weiterentwicklungen	40

4. Parallele Programmentwicklung mit CVS	43
4.1. Von Komponenten zu Softwaresystemen	43
4.2. Dateibäume versionieren mit CVS	46
4.3. Wie CVS arbeitet	49
4.4. Parallele Programmentwicklung	51
4.5. Konflikte und Absprachen	54
4.6. Erweiterungen und Alternativen	55
II. Eingabeverarbeitung	61
5. Lexikalische Analyse mit LEX	63
5.1. Eingabeverarbeitung	63
5.2. Endliche Automaten	63
5.3. Reguläre Ausdrücke	65
5.4. LEX – ein Scanner-Generator	68
5.5. LEX-Spezifikationen	69
5.6. Feinheiten der Regelgestaltung	74
5.7. Scanner erzeugen	77
6. Syntaktische Analyse mit YACC	81
6.1. Einführung	81
6.2. Eingaben beschreiben mit Grammatiken	82
6.3. YACC-Spezifikationen	84
6.4. Anbindung von LEX an YACC	91
6.5. Wie ein YACC-Parser arbeitet	92
6.6. Konflikte und wie man sie auflöst	95
6.7. Bindungen explizit deklarieren	98
6.8. GLR-Parser	98
7. Lexikalische und syntaktische Analyse mit ANTLR	103
7.1. Lexikalische Analyse	103
7.2. Syntaktische Analyse	109
7.3. So funktioniert rekursiver Abstieg	110

III. Programmbau	115
8. Programme bauen mit MAKE	117
8.1. Programme aus Komponenten bauen	117
8.2. Inkrementelles Bauen	118
8.3. Wie MAKE arbeitet	119
8.4. Makefiles in der Praxis	122
8.5. Abhängigkeiten bestimmen	128
8.6. Ergänzungen und Erweiterungen	130
8.7. ANT: Ein Framework zum Programmieren	132
9. Software automatisch konfigurieren mit AUTOCONF	137
9.1. Einführung	137
9.2. Explizite Variantenverwaltung	138
9.3. Varianten aus Schablonen erzeugen	138
9.4. Bestimmen von Eigenschaften	142
9.5. Konfigurationsskripte anwenden	145
9.6. Konfigurationsskripte erzeugen	147
9.7. Weitere Automatisierungsmöglichkeiten	149
9.8. Variantenverwaltung mit RCS oder CVS?	150
10. Programme dokumentieren mit JAVADOC	153
10.1. Programmdokumentation	153
10.2. Dokumentieren mit JAVADOC	156
10.3. Dokumentieren mit DOXYGEN	161
10.4. Literate Programming	163
IV. Prototypen	167
11. Prototypen erstellen mit Tcl/Tk	169
11.1. Prototypen und Skriptsprachen	169
11.2. Tcl: Kommandos und Skripte	171
11.3. Tcl selbst erweitern	175
11.4. Tk: Schaltflächen und Befehle	177
11.5. Tcl/Tk als Prototyp-Sprache	182
11.6. Andere Skriptsprachen	184

V. Testen und Debuggen	191
12. Systemtests mit DEJAGNU	193
12.1. Gestern lief mein Programm – heute auch?	193
12.2. Programmierte Dialoge mit EXPECT	194
12.3. EXPECT im Regressionstesten	197
12.4. DEJAGNU – ein Regressionstestrahmen	200
12.5. Testen von grafisch-interaktiven Programmen	204
13. Komponententests mit JUNIT	209
13.1. Von Systemtests zu Komponententests	209
13.2. Programmzustände validieren	209
13.3. Komponententests ausführen	211
13.4. Testfälle für JUNIT	212
13.5. Zubehör einrichten	215
13.6. Testfälle organisieren	216
13.7. Ein testgetriebener Entwicklungsprozess	218
14. Problemverwaltung mit BUGZILLA	223
14.1. Probleme verfolgen und verwalten	223
14.2. Probleme melden	224
14.3. Die Problemverwaltung BUGZILLA	226
14.4. Probleme klassifizieren	228
14.5. Probleme bearbeiten	229
14.6. Problemverwaltung organisieren	231
14.7. Problembereiche und Testergebnisse	232
15. Fehlersuche mit GDB und DDD	235
15.1. Systematische Fehlersuche	235
15.2. Hypothesen aufstellen und prüfen	238
15.3. Debuggen mit DDD: Ein Beispiel	242
15.4. Weitere Debugger-Funktionalitäten	252
15.5. Wie GDB und DDD arbeiten	254
15.6. Speicher validieren	259
15.7. Fehler konstruktiv vermeiden	262

VI. Programmanalyse	265
16. Laufzeitanalyse mit GPROF und GCOV	267
16.1. Leistungssteigerung durch Laufzeitanalyse	267
16.2. Allgemeine Laufzeitbestimmung mit TIME	267
16.3. Programmprofile mit GPROF	268
16.4. Abdeckungsanalyse mit GCOV	272
16.5. Wie GPROF und GCOV arbeiten	275
16.6. Testen mit GPROF und GCOV	275
17. Stilprüfungen mit CHECKSTYLE	281
17.1. Programmierrichtlinien	281
17.2. Einsatz von CHECKSTYLE	282
17.3. Eine eigene Prüfung	284
18. Statische Programmanalyse mit LINT	287
18.1. Einführung	287
18.2. Typüberprüfung mit LINT	287
18.3. Datenflussanalysen	290
18.4. Spezifikationen	295
18.5. Anmerkungen in der Praxis	302
19. Program-Slicing mit UNRAVEL	305
19.1. Slices und Datenfluss	305
19.2. Program-Slicing mit UNRAVEL	307
19.3. Programmabhängigkeitsgraphen	307
19.4. Mengenoperationen mit Slices	312
VII. Werkzeug-Integration	319
20. Integrierte Entwicklung mit ECLIPSE	321
20.1. Projekte verwalten	322
20.2. Die Benutzeroberfläche	323
20.3. Arbeiten im Team	326
20.4. Der JAVA-Editor von ECLIPSE	328
20.5. Programme verstehen mit JDT	330
20.6. Refactoring	334
20.7. Fehlersuche: Die Debug-Perspektive	335
20.8. Ressourcenänderungen und Programmbau	336
20.9. Aufbau und Arbeitsweise von ECLIPSE	338
20.10 Ein Beispiel-Plug-in: Die ASTView	342
20.11 Automatisierung und Intuition	345
Literaturverzeichnis	351
