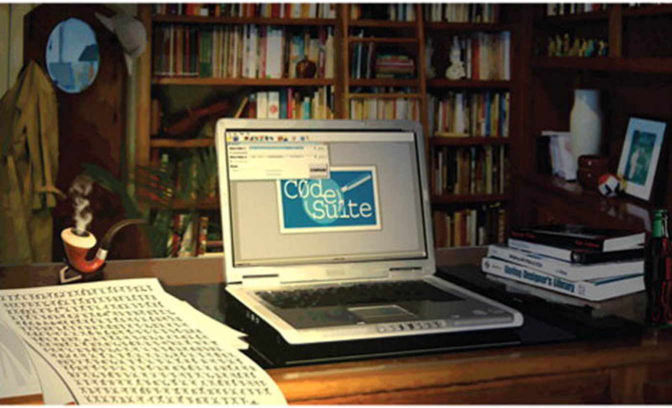
PRENTICE HALL

THE SOFTWARE IP DETECTIVE'S HANDBOOK

Measurement, Comparison, and Infringement Detection



BOB ZEIDMAN

Praise for *The Software IP Detective's Handbook*

"As a frequent expert witness myself, I found Bob's book to be important and well written. Intellectual property and software plagiarism are complicated subjects, as are patents and copyrights. This book explains the key elements better than anything else I have seen. The book is highly recommended to anyone who develops software and also to those who need to protect proprietary software algorithms. The book should also be useful to attorneys who are involved with intellectual property litigation."

—Capers Jones, president, Capers Jones & Associates, LLC

"Intellectual property [IP] is an engine of growth for our high-tech world and a valuable commodity traded in its own right. Bob Zeidman is a leading authority on software IP, and in this book he shares with us his expertise. The book is comprehensive. It contains clear explanations of many difficult subjects. Businesspeople who study it will learn how to protect their IP. Lawyers will use it to understand the specifics of how software embodies IP. Judges will cite it in their decisions on IP litigation."

—Abraham Sofaer, George P. Shultz Senior Fellow in Foreign Policy and National Security Affairs, Hoover Institution, Stanford University

"Bob has done a fantastic job in making computer science forensics understandable to mere mortals: attorneys, engineers, and managers. This is the ultimate handbook for expert witnesses, due diligence execution, and developing a baseline for software valuation. Buy it before your competitors do!"

—Don Shafer, CSDP, chief technology officer, Athens Group, LLC

"Bob has considerable experience in dealing with issues associated with unauthorized use of software code. His insights in this book are particularly helpful for those seeking to provide expert analysis with respect to software code."

—Neel I. Chatterjee, partner, Orrick, Herrington & Sutcliffe, LLC

"This readable book perfectly bridges the jargon divide between software engineers and IP attorneys. It helps each group finally understand exactly what the other is talking about. As a software developer and expert witness I will definitely keep a copy handy and recommend it to others on my team."

—Michael Barr, president, Netrino, LLC

"This book makes intellectual property law understandable and accessible to programmers by combining discussions of the law with discussions of computer science, and interweaving case studies to elucidate the intersection of these disciplines."

—Robert C. Seacord, Secure Coding Manager, Software Engineering Institute

This page intentionally left blank

The Software IP Detective's Handbook

This page intentionally left blank

The Software IP Detective's Handbook

Measurement, Comparison, and Infringement Detection

Robert Zeidman
Software Analysis and Forensic Engineering Corporation

PRENTICE
HALL

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/ph

Library of Congress Cataloging-in-Publication Data is on file with the Library of Congress

Copyright © 2011 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax: (617) 671-3447

ISBN-13: 978-0-13-703533-5
ISBN-10: 0-13-703533-0

Text printed in the United States on recycled paper at Courier in Westford, Massachusetts.
First printing, April 2011

Editor-in-Chief
Mark L. Taub

Acquisitions Editor
Bernard Goodwin

Managing Editor
John Fuller

Full-Service Production Manager
Julie B. Nahil

Copy Editor
Barbara Wood

Indexer
Lenity Mauhar

Proofreader
Linda Begley

Cover Designer
Anne Jones

Compositor
LaurelTech

This book is dedicated to all those who attempt to do things that others say are wrong or impossible, and to all those who encourage them.

This page intentionally left blank

CONTENTS

Preface xxi

Acknowledgments xxiii

About the Author xxv

PART I INTRODUCTION 1

Objectives 2

Intended Audience 2

Chapter 1 About this Book 5

Part I: Introduction 6

Part II: Software 6

Part III: Intellectual Property 6

Part IV: Source Code Differentiation 9

Part V: Source Code Correlation 9

Part VI: Object and Source/Object Code Correlation 10

Part VII: Source Code Cross-Correlation 10

Part VIII: Detecting Software IP Theft and Infringement 11

Part IX: Miscellaneous Topics 11

Part X: Past, Present, and Future 11

Chapter 2 Intellectual Property Crime 13

2.1 The Extent of IP Theft 14

2.1.1 Unintentional Loss 14

2.1.2 Poor Economy 15

2.1.3 Cheap Labor 16
2.1.4 Criminal Operations 17

PART II SOFTWARE 21
Objectives 22
Intended Audience 22
Chapter 3 Source Code 23
3.1 Programming Languages 24
3.2 Functions, Methods, Procedures, Routines, and Subroutines 26
3.3 Files 32
3.4 Programs 35
3.5 Executing Source Code 36
3.5.1 Compilers 36
3.5.2 Interpreters 36
3.5.3 Virtual Machines 36
3.5.4 Synthesizers 37
Chapter 4 Object Code and Assembly Code 39
4.1 Object Code 39
4.2 Assembly Code 40
4.3 Files 43
4.4 Programs 44
Chapter 5 Scripts, Intermediate Code, Macros, and Synthesis Primitives 45
5.1 Scripts 45
5.2 Intermediate Code 47
5.3 Macros 48
5.4 Synthesis Primitives 49

PART III INTELLECTUAL PROPERTY 53
Objectives 55
Intended Audience 55
Chapter 6 Copyrights 57
6.1 The History of Copyrights 57
6.2 Copyright Protections 60
6.3 Software Copyrights 63
6.3.1 Copyrighting Code 64
6.3.2 Copyrighting Screen Displays 67

6.3.3 Registering a Software Copyright 70
6.3.4 Term of Copyright Protection 71
6.4 Allowable and Nonallowable Uses of Copyrighted Code 72
6.4.1 Fair Use 72
6.4.2 Reimplementation 72
6.4.3 Versions, Modifications, and Derivatives 73
6.4.4 Compilations 73
6.4.5 Reverse Engineering 74
Chapter 7 Patents 79
7.1 The History of Patents 80
7.2 Types of Patents 81
7.3 Parts of a Patent 82
7.4 Patenting an Invention 85
7.5 Special Types of Patent Applications 86
7.5.1 Provisional 86
7.5.2 Divisional 87
7.5.3 Continuation 88
7.5.4 Continuation-in-Part (CIP) 88
7.6 Software Patents 90
7.7 Software Patent Controversy 91
7.7.1 Arguments for Software Patents 91
7.7.2 Arguments against Software Patents 92
7.7.3 The Supreme Court Rules? 94
7.8 Patent Infringement 95
7.8.1 Direct Infringement 95
7.8.2 Contributory Infringement 96
7.8.3 Induced Infringement 96
7.8.4 Divided Infringement 97
7.9 NPEs and Trolls 99
Chapter 8 Trade Secrets 103
8.1 The History of Trade Secrets 103
8.2 Uniform Trade Secrets Act (UTSA) 104
8.3 Economic Espionage Act 105
8.4 Aspects of a Trade Secret 106
8.4.1 Not Generally Known 108
8.4.2 Economic Benefit 108
8.4.3 Secrecy 110

8.5 Trade Secret Theft 111
8.6 Patent or Trade Secret? 112
Chapter 9 Software Forensics 113
9.1 Forensic Science 115
9.2 Forensic Engineering 116
9.3 Digital Forensics 119
9.4 Software Forensics 120
9.5 Thoughts on Requirements for Testifying 121
9.5.1 Certification 122
9.5.2 Neutral Experts 123
9.5.3 Testing of Tools and Techniques 124

PART IV SOURCE CODE DIFFERENTIATION 125
Objectives 126
Intended Audience 126
Chapter 10 Theory 127
10.1 *Diff* 128
10.1.1 *Diff* Theory 128
10.1.2 *Diff* Implementation 131
10.1.3 False Positives 131
10.1.3 False Negatives 132
10.2 Differentiation 133
10.2.1 Definitions 134
10.2.2 Axioms 134
10.3 Types of Similarity 135
10.3.1 Mutual Similarity 135
10.3.2 Directional Similarity 136
10.4 Measuring Similar Lines 136
10.4.1 Simple Matching 137
10.4.2 Fractional Matching 137
10.4.3 Weighted Fractional Matching 139
10.4.4 Case Insensitivity 140
10.4.5 Whitespace Reduction 140
10.5 Measuring File Similarity 140
10.6 Measuring Similar Programs 142
10.6.1 Maximizing Based on Files 143
10.6.2 Maximizing Based on Programs 144

Chapter 11 Implementation 147
11.1 Creating and Comparing Arrays 147
11.1.1 Sorting Lines 149
11.1.2 Ordering Lines 149
11.1.3 Hashing Lines 150
11.1.4 Saving Arrays to Disk 151
11.2 Number of Optimal Match Score Combinations 151
11.3 Choosing Optimal Match Scores for Calculating File Similarity 153
11.3.1 Simple Matching 153
11.3.2 Fractional Matching 156
11.4 Choosing File Similarity Scores for Reporting Program Similarity 161
11.4.1 Choosing File Pairs for Optimally Determining Program Similarity 161
11.4.2 Choosing File Pairs for Approximately Determining Program Similarity 162
Chapter 12 Applications 165
12.1 Finding Similar Code 165
12.1.1 Comparing Multiple Versions of Potentially Copied Code 165
12.1.2 Use of Third-Party Code 167
12.1.3 Use of Open Source Code 168
12.2 Measuring Source Code Evolution 168
12.2.1 Lines of Code 169
12.2.2 Halstead Measures 170
12.2.3 Cyclomatic Complexity 171
12.2.4 Function Point Analysis 172
12.2.5 How Source Code Evolves 173
12.2.6 Changing Lines of Code Measure (CLOC) 175
12.2.7 Non-Header Files 181
PART V SOURCE CODE CORRELATION 183
Objectives 185
Intended Audience 185
Chapter 13 Software Plagiarism Detection 187
13.1 The History of Plagiarism Detection 187
13.2 Problems with Previous Algorithms 189
13.3 Requirements for Good Algorithms 192

Chapter 14 Source Code Characterization 197
14.1 Statements 199
14.1.1 Special Statements 201
14.1.2 Instructions 202
14.1.3 Identifiers 204
14.2 Comments 206
14.3 Strings 207
Chapter 15 Theory 209
15.1 Practical Definition 210
15.1.1 Statement Correlation 211
15.1.2 Comment/String Correlation 211
15.1.3 Identifier Correlation 213
15.1.4 Instruction Sequence Correlation 213
15.1.5 Overall Source Code Correlation 213
15.2 Comparing Different Programming Languages 213
15.3 Mathematical Definitions 214
15.4 Source Code Correlation Mathematics 215
15.4.1 Commutativity Axiom 215
15.4.2 Identity Axiom 216
15.4.3 Location Axiom 216
15.4.4 Correlation Definition 216
15.4.5 Lemma 216
15.5 Source Code Examples 216
15.6 Unique Elements 218
15.7 Statement Correlation 219
15.7.1 Statement Correlation Equations 219
15.7.2 Calculating the Statement Correlation 220
15.8 Comment/String Correlation 223
15.8.1 Comment/String Correlation Equations 223
15.8.2 Calculating the Comment/String Correlation 224
15.9 Identifier Correlation 225
15.10 Instruction Sequence Correlation 227
15.11 Overall Correlation 228
15.11.1 S-Correlation 228
15.11.2 A-Correlation 229
15.11.3 M-Correlation 229
15.11.4 Recommended Correlation Method 230

Chapter 16 Implementation 233
16.1 Creating Arrays from Source Code 234
16.2 Statement Correlation 239
16.3 Comment/String Correlation 240
16.4 Identifier Correlation 241
16.5 Instruction Sequence Correlation 243
16.6 Overall Correlation 245
16.7 Comparing Programs in Different Programming Languages 246
16.8 Comparing Sections of Code Other than Files 246

Chapter 17 Applications 247
17.1 Functional Correlation 248
17.2 Identifying Authorship 249
17.3 Identifying Origin 251
17.4 Detecting Copyright Infringement (Plagiarism) 252
17.5 Detecting Trade Secret Theft 252
17.6 Locating Third-Party Code (Open Source) 253
17.7 Compiler Optimization 254
17.8 Refactoring 254
17.9 Detecting Patent Infringement 255

PART VI OBJECT AND SOURCE/OBJECT CODE CORRELATION 257
Objectives 258
Intended Audience 259

Chapter 18 Theory 261
18.1 Practical Definition 266
18.2 Extracting Elements 268
18.2.1 Comment/String Elements 269
18.2.2 Identifier Elements 269
18.3 Comparing Different Programming Languages 270
18.4 Mathematical Definitions 270
18.5 Object and Source/Object Code Correlation Mathematics 272
18.5.1 Commutativity Axiom 272
18.5.2 Identity Axiom 272
18.5.3 Location Axiom 272
18.5.4 Correlation Definition 272
18.5.5 Lemma 273

18.6 Comment/String Correlation 273

18.6.1 Comment/String Correlation Equations 273

18.7 Identifier Correlation 274

18.8 Overall Correlation 275

18.8.1 S-Correlation 276

18.8.2 A-Correlation 276

18.8.3 M-Correlation 276

18.9 False Negatives 276

Chapter 19 Implementation 279

19.1 Creating Text Substring Arrays from Object Code 279

19.2 Creating Arrays from Source Code 283

19.2.1 Extracting Identifiers 283

19.2.2 Extracting Comments and Strings 284

19.2.3 String Delimiters 284

19.2.4 Escape Characters 284

19.2.5 Substitution Characters 286

19.3 Identifier Correlation 287

19.4 Comment/String Correlation 287

19.5 Overall Correlation 287

Chapter 20 Applications 289

20.1 Pre-Litigation Detective Work 289

20.1.1 The Code in Dispute Is Distributed as Object Code 290

20.1.2 A Third Party Is Bringing the Suit 290

20.1.3 The Original Source Code Cannot Be Located 292

20.2 Tracking Malware 293

20.3 Locating Third-Party Code 293

20.4 Detecting Open Source Code License Violations 294

Part VII Source Code Cross-Correlation 295

Objectives 296

Intended Audience 296

Chapter 21 Theory, Implementation, and Applications 299

21.1 Comparing Different Programming Languages 300

21.2 Mathematical Definitions 300

21.3 Source Code Cross-Correlation Mathematics 301

21.3.1 Commutativity Axiom 301

21.3.2 Identity Axiom 302

21.3.3 Location Axiom 302
21.3.4 Correlation Definition 302
21.3.5 Lemmas 303
21.4 Source Code Examples 303
21.5 Statement-to-Comment/String Correlation 307
21.6 Comment/String-to-Statement Correlation 308
21.7 Overall Correlation 308
21.7.1 S-Correlation 309
21.7.2 A-Correlation 309
21.7.3 M-Correlation 309
21.8 Implementation and Applications 310

PART VIII DETECTING SOFTWARE IP THEFT AND INFRINGEMENT 313
Objectives 315
Intended Audience 315

Chapter 22 Detecting Copyright Infringement 317
22.1 Reasons for Correlation 318
22.1.1 Third-Party Source Code 318
22.1.2 Code Generation Tools 319
22.1.3 Commonly Used Elements 321
22.1.4 Commonly Used Algorithms 322
22.1.5 Common Author 325
22.1.6 Copying (Plagiarism, Copyright Infringement) 325
22.2 Steps to Find Correlation Due to Copying 326
22.2.1 Filtering Out Correlation Due to Third-Party Code 326
22.2.2 Filtering Out Correlation Due to Automatically Generated Code 327
22.2.3 Filtering Out Correlation Due to Common Elements 328
22.2.4 Filtering Out Correlation Due to Common Algorithms 329
22.2.5 Filtering Out Correlation Due to Common Author 329
22.2.6 Any Correlation Remaining Is Due to Copying 330
22.3 Abstraction Filtration Comparison Test 331
22.3.1 Background 331
22.3.2 How CodeSuite Implements the Test 333
22.3.3 Problems with the Test 334
22.4 Copyright Infringement Checklist 338

Chapter 23 Detecting Patent Infringement 341
23.1 Interpreting the Claims 341
23.1.1 Markman Hearing 345
23.1.2 The Role of Experts at a Markman Hearing 347
23.2 Examining the Software 348
23.2.1 Searching for Comments 348
23.2.2 Searching for Identifier Names 349
23.2.3 Reviewing from a High Level 349
23.2.4 Instrumenting Running Software 349
23.3 Tools 350
23.3.1 Understand 350
23.3.2 Klocwork Insight 351
23.3.3 DMS Software Reengineering Toolkit 351
23.3.4 Structure101 352
23.4 Determining Patent Validity 352
23.4.1 Invalidity Based on MPEP 35 U.S.C. § 102 353
23.4.2 Invalidity Based on MPEP 35 U.S.C. § 103 354
23.4.3 Invalidity Based on MPEP 35 U.S.C. § 112 356
Chapter 24 Detecting Trade Secret Theft 359
24.1 Identifying Trade Secrets 360
24.1.1 Top-Down versus Bottom-Up 360
24.1.2 Input from Owner 361
24.1.3 State with Specificity 361
24.1.4 Reasonable Efforts to Maintain Secrecy 364
24.1.5 Copied Code as Trade Secrets 365
24.1.6 Public Sources 366
24.2 Tools 367

PART IX MISCELLANEOUS TOPICS 369
Objectives 370
Intended Audience 370
Chapter 25 Implementing a Software Clean Room 371
25.1 Background 372
25.2 The Setup 374
25.2.1 The Dirty Room 374
25.2.2 The Clean Room 375
25.2.3 The Monitor 376

25.3 The Procedure 376
25.3.1 Write a Detailed Description 377
25.3.2 Sign Agreements 379
25.3.3 Examine the Original Code in the Dirty Room 380
25.3.4 Transfer Specification to the Monitor and Then to the Clean Room 380
25.3.5 Begin Development in the Clean Room 380
25.3.6 Transfer Questions to the Monitor and Then to the Dirty Room 380
25.3.7 Transfer Answers to the Monitor and Then to the Clean Room 381
25.3.8 Continue Process Until New Software Is Completed 381
25.3.9 Check Final Software for Inclusion of Protected IP 381
25.3.10 Violations of the Procedure 381
Chapter 26 Open Source Software 383
26.1 Definition 383
26.2 Free Software 386
26.2.1 Copyleft 387
26.2.2 Creative Commons 387
26.2.3 Patent Rights 387
26.3 Open Source Licenses 388
26.4 Open Source Lawsuits 390
26.4.1 *SCO v. Linux* 391
26.4.2 The BusyBox Lawsuits 393
26.4.3 *IP Innovation LLC v. Red Hat and Novell* 394
26.4.4 *Network Appliance, Inc. v. Sun Microsystems, Inc.* 395
26.5 The Pervasiveness of Open Source Software 396
Chapter 27 Digital Millennium Copyright Act 399
27.1 What Is the DMCA? 399
27.2 For and Against the DMCA 400
27.3 Noteworthy Lawsuits 403
27.3.1 *Adobe Systems Inc. v. Elcom Ltd. and Dmitry Sklyarov* 403
27.3.2 *MPAA v. RealNetworks Inc.* 404
27.3.3 *Viacom Inc. v. YouTube, Google Inc.* 404

PART X CONCLUSION: PAST, PRESENT, AND FUTURE 407

Glossary 409

References 423

Index 435

PREFACE

WHAT IS THIS BOOK ABOUT?

This book is generally about software intellectual property and specifically about the field of software forensics. While you may never testify in a courtroom, attempt to deduce the true owner of a valuable program, or measure the intellectual property (IP) value of software source code, if you are at all involved with software you need to know how to protect your rights to that software. This book will give you an understanding of those rights, their limitations, how to protect those rights, and how to take action against someone or some organization that you believe has infringed on those rights.

Unlike digital forensics, which studies the bits and bytes on a digital storage medium, such as a hard disk or DVD-ROM, without a deep understanding of what those ones and zeros represent, software forensics studies the software code that instructs a computer to perform operations. Software forensics discovers information about the history and usage of that software for presentation in a court of law. It combines information and techniques from computer science, mathematics, and law in a way that is unique and, I believe, particularly interesting.

HOW IS THIS BOOK ORGANIZED?

This book contains overviews as well as in-depth information about law, mathematics, and computer science. The book is organized into chapters that can be categorized as follows: those primarily about intellectual property law, those about computer science, those about mathematics, and those about business and business procedures. There is, of course, overlap. Chapter 1 describes the organization of the book in detail and allows you to choose those chapters that are most relevant to your career and your interests.

Who Should Read This Book?

This book is for anyone interested in software intellectual property. Specifically, I believe the book will appeal to computer scientists, computer programmers, business managers, lawyers, engineering consultants, expert witnesses, and software entrepreneurs. While the focus is on software IP measurement, comparison, and infringement detection, the book also has useful information about many issues related to software IP, and I believe that many people involved with software will find the book valuable and interesting.

Support and Comments

Thank you for purchasing my book. Please send me feedback on corrections and improvements. Of course, I also like to hear about the things I did right, the things you like about the book, and how it has helped you in some way.

Bob Zeidman
bob@SAFE-corp.biz
www.SAFE-corp.biz
Cupertino, California
March 2011

ACKNOWLEDGMENTS

When I finished my last book, back in 2002, I swore I'd never write another one. They say that pregnancy is the same way. Ask a woman right after delivering a child whether she'd do it again and, if she's coherent, she'll give you an emphatic, "Never!" Yet within months, if not days, she forgets the pain of birth and only remembers the pleasure of spending time with her baby, then her infant, then her toddler, then her kid. (By the time the child is a teenager, some of the pain of birth may have returned to her memory, though.)

So it took me seven years to forget the pain of giving birth to my last book. Actually, I remembered that pain (though the memory had softened over the years) but really felt I had something worthwhile to say once again. I'm proud of all of my books (a parent isn't supposed to favor one child), but this one is different. Those other books were intended to explain areas of engineering that many others had invented, developed, and explored before me. I explained some of my own discoveries here and there, but the bulk of the creativity is correctly credited to the pioneers who preceded me. For this book, most of the work is original. The techniques, the mathematics, the algorithms, and the procedures in this book are being adopted fairly well, and that's exciting. I hope that others can take what I've done and build upon it. I continue to do that myself and am finding lots of unexplored areas that are being revealed, offering plenty of opportunities, I believe, for mathematicians, lawyers, programmers, computer scientists, and entrepreneurs.

Many people helped me with this book, some explicitly and some implicitly. First, I'll mention the lawyers who, despite their unbelievably full schedules, still found time to

review my work and offer suggestions and corrections. This was particularly important because I'm not a lawyer, but I wanted this book to be accurate. Thanks to Joe Zito, Ed Kwok, Tait Graves, and Neel Chatterjee. Next are the mathematicians and computer scientists who gave me suggestions for representations for the formulas in the book. Thanks to John Wakerly, Kelvin Sung, and Ron Summers, and to the engineers who looked over my definitions and explanations and suggested numerous improvements. Thanks to Robert Seacord, Tom Quilty, Capers Jones, Jack Grimes, and Michael Barr. Special thanks go to Don Shafer and Chuck Pfleeger. Don did some scathing critiques of early versions of the manuscript that really pushed me to improve the book. Chuck did such detailed reviews and gave me such in-depth feedback that at times I was afraid that he understood the material better than I did. The employees at my companies have also provided a lot of great research and development, especially Nik Baer who, among other things, united the concepts of CLOC into one concise set of equations and nomenclature. Other employees who supplied a lot of great ideas and input include Jim Zamiska, Ilana Shay, Larry Melling, Tim Hoehn, and Michael Everest. I also want to thank Grace Seidman for setting up and taking the great photograph that became the cover art, and Jessica Yates for taking the flattering pictures of me.

I have a great, supportive editor at Prentice Hall: Bernard Goodwin. Sometimes I think that if I went to Bernard and said I want to do a book on the engineering principles of navel lint and dust bunnies, he'd give me the okay. As long as it was about engineering. I've heard horror stories about tough editors who put a lot of pressure on their writers, but that certainly wasn't the case with Bernard. And when I told him the schedule was slipping, but I felt I needed more time to make some important improvements, he responded, "Better trumps the schedule." I hope this is the "better" book we planned.

I want to thank Mike Flynn, Professor Emeritus of Electrical Engineering at Stanford. Mike has supported every one of my business and engineering endeavors, and this book is no exception. The concepts in this book took root in his office when he suggested I create a theory about source code correlation and spent a day working on it with me. Before that, I had a very useful program; after that, I had a new area of study including a set of axioms, theorems, and equations, and a vision about how to create something pretty big.

Finally, I want to thank my wife, Carrie. She has always been supportive of every crazy thing I decide to do, no matter how much time it takes me away from her. In this case, she gave up many weekends we wanted to spend together because I just had to get a few more pages written. I promise to spend all the weekends with her from now on, though she knows I'll find some other project that will keep me busy and that I'll keep on promising. I love her and thank her for her patience and also for the great artwork on the book cover and the figures inside the book, which she created.

About the Author

Bob Zeidman is the president and founder of Zeidman Consulting (www.ZeidmanConsulting.com), a premiere contract research and development firm in Silicon Valley that focuses on engineering consulting to law firms handling intellectual property dispute cases. Since 1983, Bob has designed computer chips and circuit boards for RISC-based parallel processor systems, laser printers, network switches and routers, and other complex systems. His clients have included Apple Computer, Cisco Systems, Cadence Design Systems, Facebook, Intel, Symantec, Texas Instruments, and Zynga. Bob has worked on and testified in cases involving billions of dollars in disputed intellectual property.

Bob is also the president and founder of Software Analysis and Forensic Engineering Corporation (www.SAFE-corp.biz), the leading provider of software intellectual property analysis tools. Bob is considered a pioneer in the field of analyzing software source code, having created the CodeSuite program for detecting software intellectual property theft, which is sold by SAFE Corp, founded in 2007.

Previously, Bob was the president and founder of Zeidman Technologies (www.zeidman.biz) where he invented the patented SynthOS program for automatically generating real-time software. Before that, Bob was the president and founder of The Chalkboard Network, an e-learning company that put high-end business and technology courses from well-known subject matter experts on the web. Prior to that, Bob invented the concept of remote backup and started Evault, the first remote backup company.

Bob is a prolific writer and instructor, giving seminars at conferences around the world. Among his publications are numerous articles on engineering and business as well as three textbooks—*Designing with FPGAs and CPLDs*, *Verilog Designer's Library*, and *Introduction to Verilog*. Bob holds numerous patents and earned two bachelor's degrees, in physics and electrical engineering, from Cornell University and a master's degree in electrical engineering from Stanford University.

Bob is also active in a number of nonprofits. He also enjoys writing novels and screenplays and has won a number of awards for this work.

1 About This Book

This book crosses a number of different fields of computer science, mathematics, and law. Not all readers will want to delve into every chapter. This is the place to start, but from this point onward each reader's experience will be different. In this chapter I describe each of the parts and chapters of the book to help you determine which chapters will be useful and appealing for your specific needs and interests.

I should make clear that I am not a lawyer, have never been one, and have never even played one on TV. All of the issues I discuss in this book are my understanding based on my technical consulting and expert witness work on nearly 100 intellectual property cases to date. My consulting company, Zeidman Consulting, has been growing over the years, and now the work is split between my employees and me. When I refer in the book to my experiences, in most cases that is firsthand information, but in other cases it may be information discovered and tested by an employee and related and explained to me.

In this book I also refer to forensic analysis tools that I have used to analyze software, in particular the CodeSuite tool that is produced and offered for sale by my software company, Software Analysis and Forensic Engineering Corporation (S.A.F.E. Corporation), and can be downloaded from the company website at www.SAFE-corp.biz. The CodeSuite set of tools currently consists of the following functions: BitMatch, CodeCLOC, CodeCross, CodeDiff, and SourceDetective. Functions are being continually added and updated. Each of these functions uses one or more of the algorithms described in later chapters.

Also, the CodeMeasure program uses the CLOC method to measure software evolution, which is explained in Chapter 12. It is also produced and sold by S.A.F.E. Corporation and can be downloaded from its own site at www.CodeMeasure.com.

Table 1.1 should help you determine which chapters will be the most helpful and relevant to you. Find your occupation at the top of the table and read downward to see the chapters that will be most relevant to your background and your job.

Part I: Introduction

The introduction to the book is just that—an introduction, intended to give you a broad overview of the book and help you determine why you want to read it and which chapters you will find most in line with your own interests and needs. This part includes a description of the other parts and chapters in the book. It also gives information and statistics about intellectual property crime, to give you an understanding of why this book is useful and important.

Part II: Software

In this part I describe source code, object code, interpreted code, macros, and synthesis code, which are the blueprints for software. This part describes these important concepts, which are well known to computer scientists and programmers but may not be understood, or may not be understood in sufficient depth, by attorneys involved in software IP litigation. This part will be valuable for lawyers to help them understand how different kinds of software code relate to each other, and how these different kinds of software code can affect a software copyright infringement, software trade secret, or software patent case.

Part III: Intellectual Property

In this part I describe intellectual property, in particular copyrights, patents, and trade secrets. I have found that many of these concepts are unclear or only partially understood by many computer scientists, programmers, and corporate managers. In this part I define these terms in ways that I believe will be comprehensible to those with little or no legal background.

Table 1.1 Finding Your Way through This Book

Chapter	Title	Computer scientist	Computer programmer	Manager	Lawyer	Consultant/ expert witness	Software entrepreneur
Part I	Introduction	X	X	X	X	X	X
Chapter 1	About This Book	X	X	X	X	X	X
Chapter 2	Intellectual Property Crime	X	X	X	X	X	X
Part II	Software	X	X	X	X	X	X
Chapter 3	Source Code			X	X		
Chapter 4	Object Code and Assembly Code			X	X		
Chapter 5	Scripts, Intermediate Code, Macros, and Synthesis Primitives			X	X		
Part III	Intellectual Property	X	X	X	X	X	X
Chapter 6	Copyrights	X	X	X		X	X
Chapter 7	Patents	X	X	X		X	X
Chapter 8	Trade Secrets	X	X	X		X	X
Chapter 9	Software Forensics	X	X	X	X	X	X
Part IV	Source Code Differentiation	X	X	X	X	X	X
Chapter 10	Theory	X				X	X
Chapter 11	Implementation	X	X			X	X
Chapter 12	Applications	X	X	X		X	X
Part V	Source Code Correlation	X	X	X	X	X	X
Chapter 13	Plagiarism Detection	X				X	X
Chapter 14	Source Code Characterization	X	X		X	X	X

Continues

Table 1.1 Finding Your Way through This Book *(Continued)*

Chapter	Title	Computer scientist	Computer programmer	Manager	Lawyer	Consultant/ expert witness	Software entrepreneur
Chapter 15	Theory	X				X	X
Chapter 16	Implementation	X	X			X	X
Chapter 17	Applications	X	X	X		X	X
Part VI	Object and Source/Object Code Correlation	X	X	X	X	X	X
Chapter 18	Theory	X				X	X
Chapter 19	Implementation	X	X			X	X
Chapter 20	Applications	X	X	X		X	X
Part VII	Source Code Cross-Correlation	X	X	X	X	X	X
Chapter 21	Theory, Implementation, Application	X	X	X		X	X
Part VIII	Detecting Software IP Theft and Infringement	X	X	X	X	X	X
Chapter 22	Detecting Copyright Infringement			X	X	X	X
Chapter 23	Detecting Patent Infringement			X	X	X	X
Chapter 24	Detecting Trade Secret Theft			X	X	X	X
Part IX	Miscellaneous Topics	X	X	X	X	X	X
Chapter 25	Implementing a Software Clean Room		X	X	X	X	X
Chapter 26	Open Source Software		X	X	X	X	X
Chapter 27	Digital Millennium Copyright Act		X	X	X	X	X
Part X	Past, Present, and Future	X	X	X		X	X

I also define the field of software forensics in this part. When I am asked to work on a case, there is sometimes confusion about the fields of software forensics and digital forensics. In some cases, engineers practicing digital forensics claim to practice software forensics and sometimes use the tools of digital forensics to attempt to draw conclusions about software IP, yielding incorrect or inconclusive results. Software forensics requires the specialized tools of the field and expertise in the field to extract relevant information from the tools, reach appropriate conclusions, and opine on those conclusions. In this part I offer definitions of the two fields. In fact, the definition of software forensics has, to this point, been somewhat vague. My explanation in this part will clarify the practice of software forensics, show how it fits into the field of forensic science, and differentiate it from digital forensics.

PART IV: SOURCE CODE DIFFERENTIATION

This part describes source code differentiation, a very basic method of comparing and measuring software source code. Source code differentiation is especially useful for finding code that has been directly copied from one program to another and for determining a percentage of direct copying. While there are many metrics for measuring qualities of software, source code differentiation has some unique abilities to measure development effort, software changes, and software intellectual property changes that are particularly useful for determining software intellectual property value for such applications as transfer pricing calculations.

In this part I introduce the mathematics of the theory of source code differentiation and explain implementations of source code differentiation for programmers who want to understand how to implement it. I also describe the "changing lines of code" or "CLOC" method of measuring software growth that is based on source code differentiation, and I compare it to traditional methods like "source lines of code" or "SLOC." I then discuss various applications of source code differentiation, though I believe that many more applications of this metric will be found in the future.

PART V: SOURCE CODE CORRELATION

This part starts by exploring the various methods and algorithms for "software plagiarism detection" that have been developed over the last few decades.

I describe the origins of these methods and algorithms, and I explain their limitations. In particular, there have been no standard definitions and no supporting theory for this work, so I introduce the theory of source code correlation and definitions for characterizing source code. This characterization of software source code is practical for determining correlation and, ultimately, for determining whether copying occurred. While the theory and definitions are broad enough to be useful in various areas of computer science, they are particularly valuable in litigation.

In this part I also describe practical implementations of the theory for those programmers who want to understand how to implement the algorithms. Additionally, I describe applications of the theory in the real world. This part is highly mathematical, though the chapter on source code characterization will be useful for lawyers in understanding how elements of software source code can be categorized, how these various elements relate, and how the elements can affect a software copyright infringement, software trade secret, or software patent case.

Part VI: Object and Source/Object Code Correlation

In this part I introduce the theory and mathematics of object code correlation, which is used to compare object code to object code to find signs of copying. I also introduce the theory of source/object code correlation, which is used to compare source code to object code to find signs of copying. Both of these correlation measures are helpful before litigation when there is no access to source code from at least one party's software. I also describe practical implementations of the theory for those programmers who want to understand how to implement these correlation measures, and I describe applications of the methods and algorithms in the real world.

Part VII: Source Code Cross-Correlation

In this part I introduce the theory and mathematics of source code cross-correlation, which is specifically used to compare functional source code statements to nonfunctional source code comments to find signs of copying. This correlation measure is effective, in certain cases, for finding copied code that has been disguised enough to avoid detection with one of the other correlation measures. I describe some ways of effectively implementing code to measure source code cross-correlation for those programmers who want to understand how to implement this measure, and I describe applications of source code cross-correlation in the real world.

Part VIII: Detecting Software IP Theft and Infringement

All of the correlation measures described in previous parts are useful for detecting software intellectual property theft; however, expert review is still required. Previously developed algorithms often produced a measure that claimed to show whether code was copied or not. In reality, a mathematical measure in and of itself is not enough to make this determination, and that is one of the problems with previous work in this area. In this part I describe detailed, precise steps to be taken once correlation has been calculated. These steps are as important to the standardization and objectivity required for determining intellectual property theft and infringement as are the various correlation measurements described in the previous parts.

Part IX: Miscellaneous Topics

This part covers areas that have come up in my involvement with intellectual property litigation. These subjects were also suggested by some of the experienced reviewers of this book who felt they deserved discussion. The issues described in this part often arise in software intellectual property cases and are also important for code developers and managers to understand. In particular, I discuss procedures for implementing a software clean room, I explain open source code, and I describe the Digital Millennium Copyright Act.

Part X: Past, Present, and Future

The topics discussed in this book are cutting-edge, and I find them to be very interesting and exciting. A lot of work remains to be done, including extending the theories, advancing the mathematics, standardizing the definitions, and promoting the methodologies. In this part I discuss what has been done to date, speculate on areas of future research that build on the concepts in this book, and look toward new applications in various aspects of law and computer science.

This page intentionally left blank

Index

A

Abelson, Hal, 387

Abstract ideas, 95

Abstraction, CodeSuite, 333–337

Abstraction filtration comparison test, 331–338

Accolade, 74–76

A-correlation, 229–230, 275–276, 309

Adobe Systems Inc. v. Dmitry Sklyarov, 403–404

Aiken, Alex, 191–192

Albrecht, Allan, 172

Alert, 285

Algorithms, 189–195, 322–325, 329

Altai, Inc., 331–336

Alto, 70

Apache HTTP Server, 179–180

Apple Computer, 66–70

Apple Computer Inc. v. Franklin Computer Corp., 66

Arney, Claude, 332

Arrays
- brute-force, 148
- creating from source code, 283–287
- hashing lines, 150–151
- ordering lines, 149–150
- saving to disk, 151
- sorting lines, 149
- from source code, 234–238, 245

ASCII, 279–283

Assembly code, 40–44, 64

Assignment operator, 24

Associative property, 137

Asterisk (*) symbol, 24

Atanasoff-Berry Computer (ABC), 89–90

Atari court ruling, 76–77

Atari Games Corp. v. Nintendo of America Inc., 74–77

AT&T, 391

Author (programmer) in common, 325, 329–330

Author rights, 57–58, 60

Authorship, identifying, 249–251
Automatic code generation (ACG), 49, 319–321, 327–328
AutoZone, 392

B

Backslash, 48, 285
Backspace, 279, 285
Baer, Nik, 182
BASIC (programming language), 25
Baxter, Ira, 254–255
Bell, Alexander Graham, 83–84
Binary code, 39
Bit, 23
BitMatch, 258, 279
Black Duck Software, 392
Blank lines, 30, 169, 236
Board of Patent Appeals and Interferences (BPAI), 95
Book publishers, 58–59
Boolean variable, 30
Bottom-up approach, 360–361
Brackets, 27–31, 149, 329
Built-in functions, 31
Business method patent, 94–95
BusyBox, 393–394
Byte, 23
Bytecode, 47

C

C (programming language), 23*n,* 25
C++ (programming language), 25
CA Scheduler, 331–338
Cadence Design Systems Inc. v. Avant! Corporation, 212
Carriage return, 279, 285
Case-insensitive comparison, 239–241, 287
Case-insensitive matching, 140
Case-sensitivity, 237, 241
Catcher in the Rye, The (Salinger), 65
Categories of IP, 1
Certification of experts, 122–123
Challenger space shuttle, 116–118
Changing lines of code measure (CLOC)
  CLOC growth, 177, 179, 180
  file continuity, 177–178
  LOC continuity, 178
  measured results, 178–180
  SLOC growth, 177
  terms, 175–177
Character-weighting function, 134, 139
Checklist, copyright infringement, 338–340
Churchett, Dale, 254–255
Circular 1, *Copyright Basics* (U.S. Copyright Office), 60
CL (continuing LOC), 176, 178, 179
Claim construction, 341–347
Clean room development
  dirty room, 374–375, 380
  guidelines for implementation, 376–380
  history, 372–373
  the monitor, 376, 380–381
  room separation, 378
  security, 378
  setup, 374–376
Clinton, William Jefferson, 105
CLOC. *See* Changing lines of code measure (CLOC)
Clone detection, 254–255
COBOL (programming language), 25
Code comments, 170–171
Code generation tools. *See* Automatic code generation (ACG)

Code loop, 31

CodeMeasure, 6, 175

CodeSuite

- abstraction, 333–337
- BitMatch, 258, 279
- CodeCLOC, 126
- CodeCross, 296
- CodeDiff, 126, 147, 182, 408
- CodeMatch, 115, 130, 193–195, 212, 233, 335
- filtration, 334
- SourceDetective, 327–328, 334
- tools, 5

Coincidence, 330*n*

Collected Cases of Injustice Rectified (Ci), 115

Colting, Fredrik, 65

Columbia space shuttle, 116–118

Comment comparison, 212

Comment delimiters, 207, 235–236

Commented-out statements, 310–311

Comments

- in code, 27–29, 31, 170–171, 198, 206–207
- extracting, 284
- and patent infringement, 348

Comment/string correlation, 211, 214, 224–225, 240–241, 273–274, 287

Comment/string element identity match score, 223, 273, 308

Comment/string elements, 268–269

Comment/string-matching algorithm, 240, 245

Comment/string-to-statement correlation, 299, 308

Commodities trading patent, 94–95

Commutativity axiom, 215–216, 272, 301–302

Commutativity property, 129–130, 134–136

Compilations, code, 73–74

Compilers, 36, 39–40, 254

Compiling process, 39

Computer Associates (CA), 331–338

Computer Associates International Inc. v. Altai Inc., 331–337

Computer patents and history, 88–89

Computer program definition, 60

Computer Software Copyright Act of 1980, 60

Computer visual display, 67–70

"Conditions for Patentability; Non-Obvious Subject Matter" (MPEP, 35 U.S.C. § 103) 354–355

"Conditions for Patentability; Novelty and Loss of Right to Patent" (MPEP, 35 U.S.C. § 102), 353–354

Confidentiality agreements, 107, 110, 364–365

Constant, Benjamin, 54

Constants, 198

Context-dependent macros, 51

Continuation character, 48

Continuation patents, 88

Continuation-in-part (CIP) patents, 88

Continuing file, 176–177, 179

Continuing LOC, 176, 178, 179

Contributory infringement, 96

Control words, 198, 202–203

Copyleft, 387

Copyright(s), 1

- Atari court ruling, 76–77
- code, 64–66
- compilations, 73–74
- fair use, 72, 74–76
- Great Britain, 58

Copyright(s) (*Cont.*)
  history and origins, 57–60
  inception, 61
  ineligible works and material, 61
  infringement lawsuit, 62
  redacting code, 64, 71
  registration, 61–63, 70–71
  reimplementation, 72–73
  reverse engineering, 74–76
  revisions and updates, 73
  rule of doubt, 71
  screen displays, 67–70
  term, 71
  U.S. Copyright Office circular, 60–61
  WIPO definition, 60
Copyright Acts, 58–60, 63, 71
Copyright infringement
  abstraction-filtration-comparison test, 331–338
  author in common, 325, 329–330
  checklist, 338–340
  code generation tools, 319–321, 327–328
  common algorithms, 322–325, 329
  open source code, 318–319, 326–327
  plagiarism, 325–326, 330–331
  standard elements, 321, 328–329
Correlation
  definition, 216, 272–273, 302–303
  due to copying, 326–331
  functional, 248–249
  six reasons for, 318–326
Country of origin, identifying, 251
Court of Appeals for the Federal Circuit (CAFC), 95
Creative Commons (CC) license, 387
Creative works, 331
Crime. *See also* Digital Millennium Copyright Act (DMCA); Software forensics
  Economic Espionage Act, 105–106
  intellectual property (IP), 13–19
Cross compiler, 40
Cross-language comparisons, 213–214, 246
Cybertheft, 17
Cyclomatic complexity (a.k.a. McCabe measure), 171–172

D

DaimlerChrysler, 392
Data recovery, 119–120
Date, priority, 87–88
Date of copyright registration, 62
Date of patent grant, 81
Date of publication, 58, 62–63
Debug option, 266, 269
Decompiled source code file, 263–265
Defender, 64–65
Defense Command, 64–65
Delimiter, 227, 235, 275, 284
Derivative works, 60, 70, 73
  Salinger v. Colting, 65
Design patent, 81
Diamond v. R. Diehr, 91
Diff (comparison utility)
  false negatives/false positives, 131–132
  implementation, 131
  theory, 128–131
Digital forensics, 119–120
Digital Millennium Copyright Act (DMCA), 58
  Adobe Systems Inc. v. Dmitry Sklyarov, 403–404
  industry support of, 400–403
  MPAA v. RealNetworks Inc., 404

opposition, 401–402
origins, 399
Titles I through V, 399–400
Viacom Inc. v. YouTube, Google Inc., 404–405
Direct infringement, 95–96
Directional similarity, 136, 156
Dirty room, 374–375, 380
Divided infringement, 97
Divisional patents, 84, 87–88
DMS Software Reengineering Toolkit (SRT), 351–352
Double equal (==) sign, 31
Double quotes, 200, 285
Double slash (//), 207
Duration of copyright, 71–72
Duration of patents, 87

E

Eckert-Mauchly Computer Corporation, 89
Economic benefit, trade secrets, 108–109
Economic Espionage Act (18 U.S.C. § 1831 and § 1832), 105–106
Eldred, Eric, 387
Element identity match score, 214, 271, 300
Element match score, 214, 271, 300
Elements, 198–199, 214, 218–219, 321, 328–329
else statement, 201–202
Employee policies, 14–15, 110–112, 364
Employee theft, 14–16
Enabling (patent criterion), 85
Energy Risk Management patent application, 94–95
ENIAC, 88–89
Equality test operator (==), 203–204
Equals (=) symbol, 24, 202
Escape character, 284–285
Escape sequences, 208, 285–286
eVault Remote Backup Service, 98–99
Executable code, 39, 44
Executing source code, 36–37
Expert witnesses, 121–124, 347
Extended ASCII, 281–283
Extrinsic evidence, 346–347

F

Faidhi, J. A. W., 187–188
Fair use, 72, 74–76
False negatives/false positives, 132–133, 230–231, 276–277
Feynman, Richard, 116–118
File continuity, 177–178, 179
File decay, 177–178
File element identity match score, 215, 272, 301
File element match score, 215, 271, 301
File similarity scores, 161–164
Filtration, 326–338
Firmware, 66
First Electronic Computer, The (Burks and Burks), 90
First-to-file patent law, 86
First-to-invent patent law, 86
Flynn, Mike, 194
for statement, 201–202
Foreign laws, and IP theft, 16–17
Forensics
digital forensics, 119–120
expert witnessing, 121–124
forensic engineering, 116–118
forensic examiner, 119–121
forensic science, 114–116
Form feed, 285

Fortran (programming language), 25

Four essential freedoms (FSF), 386

Fowler, Martin, 254

Fractional matching, 137–141, 156–160

Franklin Computer Corporation, 66

Free, Libre and Open Source Software (FLOSS), 294

Free software, 386–388

Free Software Foundation (FSF), 294, 386–388

French Revolution, 54

Function declarations, 35

Function point analysis, 172

Functional correlation, 248–249

Functions, 26–32, 198

G

Galler, Bernard, 376

Gates, Bill, 70, 173

GNU Operating System open source project, 386

GNU Public License (GPL), 294, 386, 393–394

Google, Inc., 404–405

Graves, Charles Tait, 362

Great Britain, 58

Guidelines, clean room development, 376–380

GUIs, 69–70

H

Halstead, Maurice H., 170, 189

Halstead measures, 170–171, 173, 189

Hamblen, James, 189

Hard disk data recovery, 119–120

Hashing algorithm, 131, 151

Hashing lines, 150–151

Head patents (Texas Instruments), 341–345

Header comments, 27–29

Header files, 33–34

Headway Software, 352

Hex number, 285

History

- clean rooms, 372–373
- copyrights, 57–60
- Digital Millennium Copyright Act (DMCA), 399
- forensic science, 115
- patents, 79–80
- plagiarism detection, 187–189
- trade secrets, 103–104

Hitz, Dave, 395–396

Honeywell, 89

I

"Identification of Trade Secret Claims in Litigation: Solutions for a Ubiquitous Dispute," (Graves et al.), 362

Identifier(s), 198, 204–205

- delimiter, 284
- elements, 269
- extracting, 284
- match score, 225–227
- names, 188, 213, 349

Identifier correlation, 211, 213, 214, 225–227, 241–243, 268, 274–275, 287

Identifier-matching algorithm, 241–243, 245

Identity axiom, 216, 272, 302

Identity property, 129, 134–135

if statement, 201–202

Include files, 33

Induced infringement, 96

Industrial property, 1

Ineligible material, 61

Infringement. *See also* Copyright infringement; Patent infringement

contributory, 96

direct, 95–96

divided, 97

induced, 96

lawsuit, 62

patents, 95–97

Instruction sequence correlation, 213, 214, 227–228, 243–245

Instruction sequence-matching algorithm, 243–244

Instructions (code), 198, 202–203

Instrumenting the software, 349–350

Intellectual property (IP). *See also* Copyrights; Patent(s); Software forensics; Trade secret(s)

categories, 1

crime, 13–19

definition, 1

early legal rulings, 53–55

opponent, 54

tax issues, 181–182

Intermediate code, 47

Internal Revenue Service, 181–182

International Copyright Act of 1891, 58

Internet Archive, 366

Internet code searches, 18, 327–328

Interpreters (source code), 36, 45–46

Intrinsic evidence, 346–347

Invalidity sections of MPEP, 353–357

IP Innovation v. Red Hat and Novell, 394–395

J

Jankowitz, Hugo, 189–190

Java (programming language), 25

JavaScript (programming language), 25, 46

Jefferson, Thomas, 80

Jobs, Steve, 70

JPlag program, 192

K

Kentucky Fried Chicken, 106–107

Khodorkovsky, Mikhail, 16

King Henry VI, 80

Klocwork Insight, 351

L

Labels, 198, 204–205

LCCS (longest common contiguous subsequence), 134, 137–138, 140, 210*n*, 242–243

LCS (longest common subsequence), 134, 137–138, 140, 210*n*

Lebedev, Platon, 16

Lemma, 137, 216, 273, 303

Lessig, Larry, 387

Licenses, open source, 388–390

Line continuity, 177–178

Line decay, 177–178

Lines of code (LOC), 169–170, 173–175, 177–180

Line-weighting function, 134, 139–140

Linux (operating system), 391–393

LISP (programming language), 25

Literary work, 67

Litigation preparation, 289–293

LLOC, 169–170

LOC, 169–170, 173–175, 177–180

LOC continuity, 178, 179

Local variables, 30

Location axiom, 216, 272, 302

Location property, 130–131, 135

Logical SLOC, 169–170

Longest common contiguous subsequence. *See* LCCS
Longest common subsequence. *See* LCS
Longest common substring. *See* LCCS
Loop, 31

M

Machine language, 39
Macros, 48–49
Malpohl, Guido, 192
Malware, 293
Manual of Patent Examining Procedure (MPEP) (U.S. Patent and Trademark Office), 83, 353–357
Markman hearing, 343–347
Markman v. Westview Instruments, Inc., 345–347
Match score, 134, 137–138, 140–145, 151–160, 214–216, 275, 307
Match types, 210
Matching lines, 133, 135–136
Mathematical definitions, 214–215, 270–272, 300–301
Mathematics, 215–216, 272–273, 301
MatrixCalc(), 156–160
MatrixCalcOptimized(), 159–160
McAfee, Inc. research report, 14–17
McCabe, Thomas, 171
McCabe measure (a.k.a. cyclomatic complexity), 171–172
MCF (modified continuing files), 174, 176, 179
M-correlation, 229–230, 275–276, 309–310
Measuring similar files, 140–142
Measuring similar lines, 136–140
Measuring similar programs, 142–145
Measuring source code evolution. *See* Source code changes and evolution
Metadata preservation, 119–120
Methods, 26, 204
Michels, Doug, 391
Michels, Larry, 391
Microsoft Visual Studio code, 319–321
Microsoft Windows, 67–70
Misappropriation, 111–112
Modified continuing files (MCF), 174, 176, 179
Moglen, Eben, 386
Molasses, 100–101
Monitor, 376, 380–381
Monsoon Multimedia, Inc., 393
Morton Thiokol, Inc., 117
MOSS program, 192
Mozilla Firefox, 178–180
MPAA v. RealNetworks Inc., 404
MPEP. *See Manual of Patent Examining Procedure*
Mutual similarity, 135–136, 156

N

National Conference of Commissioners on Uniform State Laws (NCCUSL), 104
Native code, 40
NDAs, 110–112, 364
Netscape Communications Corporation, 384
Neutral experts, 123–124
Newline, 208, 240, 285
NF (new files), 176, 178
Nintendo, 74–77
NLCF (new LOC in continuing files), 176, 179
Nondisclosure agreements (NDAs), 110–112, 364
Non-enabling (patent criterion), 85
Non-header files, 181
Noninfringement, 352–357

Nonobviousness, 85, 91, 354–355
Non-practicing entities (NPEs), 99–100
Nonprintable characters, 285
Novell Corporation, 391–392, 394
Novelty, 85
Null, 236, 285

O

Object code
  A-correlation, 275
  arrays, 283–287
  and assembly code, 40–43
  comment/string correlation, 273–274, 287
  copyrights, 64–66
  decompiled source code file, 263–265
  definition, 266–268
  elements, 268–270
  false negatives, 276–277
  files, 43–44
  identifier correlation, 274–275, 287
  libraries, 290
  litigation preparation, 289–293
  malware, 293
  mathematical definitions, 270–272
  mathematics, 272–273
  M-correlation, 275
  open source code, 294
  original source code file, 261–263
  overall correlation, 287
  S-correlation, 275
  text strings, 279–283
  third-party code, 293
Octal number, 285
On Demand Machine Corporation v. Ingram Industries, Inc. et al., 97
OOSITA (one of ordinary skill in the art), 86
Open source code, 108, 168, 253–254, 294, 318–319, 326–327
Open source database search engine, 327
Open Source Initiative (OSI), 384–385
Open source software
  free software, 386–388
  lawsuits, 390–396
  open source licenses, 388–390
  OSI certification, 385
Operators, 198, 203–204, 233
Opposition to DMCA, 401–402
Ordering lines, 149–150
Origin identity, 251
Outsourcing, 16–17
Overall correlation, 245–246, 287, 299, 308–309
Overall source code correlation, 213, 214, 228–230, 268

P

Parker, Alan, 189
Partial matching, 210, 242, 274, 287, 307
Patent(s)
  anticipated, 353–354
  arguments for/against, 91–94
  compared with trade secrets, 112
  continuation, 88
  continuation-in-part (CIP), 88
  criteria, 85
  divisional, 87–88
  duration, 87
  free software and, 388
  history, 80–81
  NPEs and trolls, 99–100
  process, 85–86
  provisional, 86–87
  public disclosure, 86

Patent(s) (*Cont.*)
  software, 90–95
  structure of, 82–84
  types, 81–82
  validity/invalidity, 352–357
Patent infringement
  claim construction, 341–347
  claims interpretation guidelines, 346–347
  DMS Software Reengineering Toolkit (SRT), 351–352
  invalidity sections of MPEP, 353–357
  Klocwork Insight, 351
  types, 95–97
Patent trolls, 99–100
Peabody v. Norfolk, 104
Perens, Bruce, 384
Perl, 46
Phillips v. AWH Corporation, 346
Phlippsen, Michael, 192
PHP (programming language), 25
Physical SLOC, 169
Plagiarism. *See also* Copyright infringement
  algorithms, 192–195
  code correlation, 252, 325–326
  defined, 191
  detection history, 187–192
  Jankowitz algorithm, 189–190
  JPlag program, 192
  levels of, 188
Plague program, 191–192
Plant patents, 81
Plus (+) symbol, 24
Ponemon Institute, 15
Prechelt, Lutz, 192
Primitives, 37, 49–51
Printable characters, 279–283
Priority date, 87–88
Procedures (functions), 26
Programming languages
  cross language comparison, 213–214
  description table, 25
Programming style, 325
Prototypes, 35
Provisional patent application, 86–87
PTO. *See* U.S. Patent and Trademark Office (PTO)
Public disclosure of invention, 86, 87*n*
Public knowledge of trade secrets, 366–367
Public performance, 61
Publication date, 61–63
Published works and copyright, 60–62
Python, 46

Q

Qualifications of expert witnesses, 121–124
Queen Anne, 58–59
Quotation marks, 200, 285

R

Rabbinic law, 57
Rabin-Karp algorithm, 138
Range, Brian D., 362
Raymond, Eric S., 384
Reagan, Ronald, 116
Recursive function, 156–160
Red Hat, 394–395
Redacting code, 64, 71
Refactoring, 254–255
Registration of copyright, 61–62
Reimplementing computer programs, 72–73
Remote backup, 97–98
Restatement of Torts (§ 757 and § 758), 104
Reverse engineering, 74–76, 350–352

Revisions to software, 73
Rewriting source code, 72–73
Robinson, S. K., 187–188
ROM, 66
Routines (functions), 26
Rule of doubt, 71

S

S.A.F.E. Corporation (Software Analysis and Forensic Engineering Corporation), 5, 279, 333, 408
Salinger v. Colting, 65, 81
Schleimer, Saul, 191
Scientific Toolworks, 350
SCO Group, 391
SCO v. IBM, 391
SCO v. Linux, 391–392
S-correlation, 228–230, 275–276, 309
Screen displays, 67–70
Scripting languages, 46
Scripts, 45–47
Secrecy, 364–365
Secrecy policy, 110–111
Security
  clean room, 378
  measures, 365
  policies, 14–15
  source code, 14
Sega Enterprises Ltd. v. Accolade Inc., 74–76
Semicolon, 200
Separators, 233–235
Shell Oil, 16
Similarity measurements
  case insensitivity, 140, 156
  directional similarity, 136, 156
  fractional matching, 137–140, 156–160
  mutual similarity, 135–136, 156
  programs, 142–145
  simple matching, 137, 140–142, 153–156
  weighted fractional matching, 139–140
  whitespace, 140
Simple matching, 137, 140–142, 153–156
Single quote, 285
Slashes, 48, 207, 235, 285
SLOC (source lines of code), 169–170, 173–175, 177–180
SLOC growth, 177, 179, 180
Software, instrumenting, 349–350
Software and Intellectual Property Protection (Galler), 376
Software architecture, 337–338
Software copyrights
  Apple Computer Inc v. Franklin Computer Corp., 66
  Apple GUIs, 69–70
  first submissions, 63
  length of protection, 71
  redaction, 64, 71
  registration, 70–71
  screen displays, 67–69
  trade secrets, 63–64
  Williams Electronics, Inc. v. Arctic International, Inc., 64–65
Software forensics
  digital forensics, 119–120
  expert witnessing, 121–124
  forensic engineering, 116–118
  forensic science, 114–116
  need for, 114–115
Software Freedom Law Center (SFLC), 294, 393
Software patents
  arguments for/against, 91–94
  infringement, 95–97
  worldwide variances, 91

Software plagiarism. *See* Plagiarism
Software synthesis, 37, 49–51
Software valuations, 181–182
Sorting lines, 149
Source code
  copyrights, 64
  detailed sample, 27–29
  example language, 23*n*
  executing, 36–37
  files, 32–35, 43–44
  functions, 26–32
  header files, 33–35
  programming languages, 24–25
  programs, 35–36
  rewriting, 72–73
  submission requirements, 70–71
  transfer security, 14
Source code changes and evolution, measuring
  CLOC metric, 175–180
  cyclomatic complexity, 171–172
  function point analysis, 172
  Halstead measures, 170–171
  lines of code, 169–170
  non-header files, 181
  project evolution, 173–175
  transfer pricing, 181–182
Source code characterization
  comments, 206–207
  source code elements, 197–199
  statements, 199–205
  strings, 207–208
Source code correlation
  arrays, 234–238
  authorship, 247, 249–251
  Cadence Design Systems Inc. v. Avant! Corporation, 212
  comment comparison, 212
  comment/string correlation, 211, 223–225, 240–241
  compiler optimization, 248, 254
  copyright infringement, 247, 252
  cross-language comparisons, 213–214, 246
  false negatives/false positives, 230–231
  functional correlation, 248–249
  identifier correlation, 213, 225–227, 241–243
  instruction sequence correlation, 213, 227–228, 243–245
  match types, 210
  mathematical definitions, 214–215
  mathematics, 215–216
  open source code, 248, 253–254
  origin identity, 247, 251
  overall correlation, 228–230, 245–246
  overall source code correlation, 213
  patent infringement, 248, 255–256
  refactoring, 248, 254–255
  source code examples, 217–218
  statement correlation, 211, 220–222
  statement-matching algorithm, 239–240
  trade secret theft, 247, 252–253
  transformational matches, 246
  unique elements, 218–219
Source code cross-correlation
  commented-out statements, 310–311
  comment/string-to-statement correlation, 308
  commutativity axiom, 301–302
  correlation definition, 302–303
  cross-language comparison, 300
  identity axiom, 302
  lemmas, 303
  location axiom, 302

mathematical definitions, 300–301
overall correlation, 308–309
sample code, 303–307
statement-to-comment/string correlation, 307
Source code differentiation
arrays, 147–151
axioms, 134–135
changing lines of code measure (CLOC), 175–180
cyclomatic complexity (aka McCabe measure), 171–172
definitions, 134
diff (comparison utility), 128–133
directional similarity, 135–136, 156
file similarity scores, 161–164
finding similar code, 165–168
function point analysis, 172
halstead measures, 170–171, 173
lines of code, 169–170
measuring file similarity, 140–142
measuring similar lines, 136–140
measuring similar programs, 142–145
mutual similarity, 135–136, 156
non-header files, 181
optimal match score, 151–160
similarity, 135–136
similarity score, 133
source code differentiation, 133–135
source code evolution, 173–175
tax issues, 181–182
Source lines of code (SLOC), 169–170, 173–175, 177–180
SourceDetective, 327–328, 334
Source/object code correlation applications
malware, 293
open source code, 294
pre-litigation work, 289–293
third-party code, 293
Source/object code correlation implementation
arrays, 283–287
comment/string correlation, 287
identifier correlation, 287
overall correlation, 287
text strings, 279–283
Source/object code correlation theory
A-correlation, 275
comment/string correlation, 273–274
decompiled source code file, 263–265
definition, 266–268
elements, 268–270
false negatives, 276–277
identifier correlation, 274–275
mathematical definitions, 270–272
mathematics, 272–273
M-correlation, 275
original source code file, 261–263
S-correlation, 275
Space shuttles, 116–118
"Specification" (MPEP, 35 U.S.C. § 112), 356–357
Specificity, 361–364
Specifiers, 198, 203
Spelling errors, 211, 328–330
Sperry Rand, 89–90
SQL (programming language), 25
Stallman, Richard, 386, 392
Standard elements, 321, 328–329
State Street Bank & Trust Co. v. Signature Financial Group, 91
Statement correlation, 211, 239–240, 248
calculation, 214, 220–223

Statement correlation (*Cont.*)
case-sensitivity, 239
equations, 219–220
Statement element identity match score, 307
Statement element match score, 223
Statement-matching algorithm, 239
Statements
control words, 202–203
identifiers, 204–205
if and else statements, 201
operators, 203–204
quotation marks, 200
semicolon, 200
specifiers, 203
for statement, 201
Statement-to-comment/string correlation, 299–300, 307
Static analysis tools, 350–352
Statue of Liberty, 81–82
Statute of Anne, 58–59
Statutory damages, 62
String correlation, 211, 214, 223–225
String delimiters, 235, 284
Strings, 198, 200, 207–208
Stroud, John, 171
Stroud number, 171
Structure101, 352
Subroutines (functions), 26, 48
Substantial changes, 73
Substantial code, 331
Substantial similarity, 332–333
Substitution characters, 286–287
Substring, 283
Symantec v. Commissioner of Internal Revenue, 181–182
Synthesis primitives, 49–51
Synthesizers, 37

T

Tab, 208, 285
Tarantella, 391
Tax issues, 181–182
TCF (total continuing files), 176–178
Telephone patent, 83–84
Term of copyright protection, 71, 92
Texas Instruments, 341–345
Text characters, 279
Text length threshold, 283
Text strings, 279–283
Text substring, 281, 283
TF (total files), 176, 179
Third-party code, 248, 253–254, 293–294. *See also* Open source code
Title 17, U.S. Code, 72
TL (total LOC), 176–180
TLCF (total LOC in continuing files), 176, 179
TNL (total new LOC), 176, 179
Top-down approach, 360
Torvalds, Linus, 392
Trade secrets
characteristics, 106–111, 365–366
compared with patents, 112
copied code, 365–366
Economic Espionage Act (18 U.S.C. § 1831 and § 1832), 105–106
history, 103–104
identifying, 360–367
Peabody v. Norfolk, 104
public knowledge of, 366–367
secrecy of, 364–366
software copyrights, 63–64, 71
specificity, 361–364
theft, 106, 111–112, 252–253

Vickery v. Welch, 104
Yield Dynamics, Inc. v. TEA Systems Corporation, 109
Transfer pricing, 181–182
Transformational matches, 210, 246
Translations, 73

U

UCF (unchanged continuing files), 174, 176, 177–178, 179
Unchanged file continuity, 177, 179
Unclean hands, 76–77
Underscore, 234, 270
Understand (reverse-engineering tool), 350–351
Unicode, 279, 285
Unicode Consortium, 281
Unified Modeling Language (UML), 21
Uniform Trade Secrets Act (UTSA), 104–105
Unique elements, 218–219
UNIX (operating system), 128, 391–392
Unpublished works and copyright, 60
U.S. Code (Title 17), 72
U.S. Constitution (Art. I, sec. 8), 54
U.S. Copyright Office circular, 60–61
U.S. Patent Act, 80
U.S. Patent and Trademark Office (PTO), 81, 90–91
U.S. Supreme Court, 95
Utility (patentability criteria), 85
Utility patent, 81

V

Variables, 24, 30, 198
Verilog (programming language), 25, 49, 322–325
Veritas, 181–182
Vertical tab, 285
VHDL, 49
Viacom Inc. v. YouTube, Google Inc., 404–405
Vickery v. Welch, 104
Video game lawsuits, 64–65, 74–77
Virtual machines, 36–37
VisiCalc, 66
Visual arts work, 67
Visual Basic (programming language), 25

W

Washington, George, 80
WayBack Machine, 366
Weighted fractional matching, 139–140
Whale, Geoff, 191
Whitespace, 283
Whitespace patterns, 317*n*
Whitespace reduction, 134, 140, 234–235, 269, 273
Wilkerson, Daniel, 191
Williams, James, 332
Williams Electronics, Inc. v. Arctic International, Inc., 64–65
WIPO (World Intellectual Property Organization) definitions
copyright, 60
intellectual property (IP), 1
patent, 79
trade secrets, 103
WIPO treaties, 399
Word (bytes), 23

X

Xenix, 391
Xerox, 67–70

Y

YAP programs, 191

Yield Dynamics, Inc. v. TEA Systems Corporation, 109

YouTube, 404–405

Z

Zeidman Consulting, 5, 338–340

Zeke, 332

Zettabyte File System (ZFS), 395