CONTENTS

PREFACE xi

Part One GENERAL PRINCIPLES OF SEARCHING FOR AND USING EVIDENCE

1 THE QUEST FOR CONVINCING EVIDENCE 3
by Tim Menzies and Forrest Shull

In the Beginning 4
The State of Evidence Today 4
Change We Can Believe In 8
The Effect of Context 10
Looking Toward the Future 11

2 CREDIBILITY, OR WHY SHOULD I INSIST ON BEING CONVINCED? 17
by Lutz Prechelt and Marian Petre

How Evidence Turns Up in Software Engineering 17
Credibility and Relevance 19
Aggregating Evidence 22
Types of Evidence and Their Strengths and Weaknesses 25
Society, Culture, Software Engineering, and You 32
Acknowledgments 33

3 WHAT WE CAN LEARN FROM SYSTEMATIC REVIEWS 35
by Barbara Kitchenham

An Overview of Systematic Reviews 36
The Strengths and Weaknesses of Systematic Reviews 39
Systematic Reviews in Software Engineering 44
Conclusion 49

4 UNDERSTANDING SOFTWARE ENGINEERING THROUGH QUALITATIVE METHODS 55
by Andrew Ko

What Are Qualitative Methods? 56
Reading Qualitative Research 58
Using Qualitative Methods in Practice 60
Generalizing from Qualitative Results 62
Qualitative Methods Are Systematic 62

5 LEARNING THROUGH APPLICATION: THE MATURING OF THE QIP IN THE SEL 65
by Victor R. Basili

What Makes Software Engineering Uniquely Hard to Research 65

Making software
2010

digitalisiert durch:
IDS Basel Bern

A Realistic Approach to Empirical Research 66
The NASA Software Engineering Laboratory: A Vibrant Testbed for Empirical Research 67
The Quality Improvement Paradigm 69
Conclusion 75

6 PERSONALITY, INTELLIGENCE, AND EXPERTISE: IMPACTS ON SOFTWARE DEVELOPMENT 79
by Jo E. Hannay

How to Recognize Good Programmers 81
Individual or Environment 95
Concluding Remarks 101

7 WHY IS IT SO HARD TO LEARN TO PROGRAM? 111
by Mark Guzdial

Do Students Have Difficulty Learning to Program? 112
What Do People Understand Naturally About Programming? 114
Making the Tools Better by Shifting to Visual Programming 117
Contextualizing for Motivation 118
Conclusion: A Fledgling Field 121

8 BEYOND LINES OF CODE: DO WE NEED MORE COMPLEXITY METRICS? 125
by Israel Herraiz and Ahmed E. Hassan

Surveying Software 126
Measuring the Source Code 127
A Sample Measurement 128
Statistical Analysis 133
Some Comments on the Statistical Methodology 139
So Do We Need More Complexity Metrics? 140

Part Two SPECIFIC TOPICS IN SOFTWARE ENGINEERING

9 AN AUTOMATED FAULT PREDICTION SYSTEM 145
by Elaine J. Weyuker and Thomas J. Ostrand

Fault Distribution 146
Characteristics of Faulty Files 149
Overview of the Prediction Model 150
Replication and Variations of the Prediction Model 151
Building a Tool 157
The Warning Label 157

10 ARCHITECTING: HOW MUCH AND WHEN? 161
by Barry Boehm

Does the Cost of Fixing Software Increase over the Project Life Cycle? 162
How Much Architecting Is Enough? 162
Using What We Can Learn from Cost-to-Fix Data About the Value of Architecting 165
So How Much Architecting Is Enough? 178
Does the Architecting Need to Be Done Up Front? 181
Conclusions 182

11 CONWAY'S COROLLARY 187
by Christian Bird

Conway's Law 187
Coordination, Congruence, and Productivity 189
Organizational Complexity Within Microsoft 194
Chapels in the Bazaar of Open Source Software 201
Conclusions 205

12 HOW EFFECTIVE IS TEST-DRIVEN DEVELOPMENT? 207
by Burak Turhan, Lucas Layman, Madeline Diep, Hakan Erdogmus, and Forrest Shull

The TDD Pill—What Is It? 208
Summary of Clinical TDD Trials 209
The Effectiveness of TDD 211
Enforcing Correct TDD Dosage in Trials 214
Cautions and Side Effects 215
Conclusions 216
Acknowledgments 217

13 WHY AREN'T MORE WOMEN IN COMPUTER SCIENCE? 221
by Michele A. Whitecraft and Wendy M. Williams

Why So Few Women? 222
Should We Care? 227
Conclusion 234

14 TWO COMPARISONS OF PROGRAMMING LANGUAGES 239
by Lutz Prechelt

A Language Shoot-Out over a Peculiar Search Algorithm 240
Plat_Forms: Web Development Technologies and Cultures 248
So What? 257

15 QUALITY WARS: OPEN SOURCE VERSUS PROPRIETARY SOFTWARE 259
by Diomidis Spinellis

Past Skirmishes 260
The Battlefield 261
Into the Battle 265
Outcome and Aftermath 286
Acknowledgments and Disclosure of Interest 289

16 CODE TALKERS 295
by Robert DeLine

A Day in the Life of a Programmer 295
What Is All This Talk About? 298
A Model for Thinking About Communication 307

17 PAIR PROGRAMMING 311
by Laurie Williams

A History of Pair Programming 312

Pair Programming in an Industrial Setting 314
Pair Programming in an Educational Setting 317
Distributed Pair Programming 319
Challenges 320
Lessons Learned 321
Acknowledgments 322

18 MODERN CODE REVIEW 329
by Jason Cohen

Common Sense 329
A Developer Does a Little Code Review 330
Group Dynamics 334
Conclusion 336

19 A COMMUNAL WORKSHOP OR DOORS THAT CLOSE? 339
by Jorge Aranda

Doors That Close 339
A Communal Workshop 342
Work Patterns 345
One More Thing... 347

20 IDENTIFYING AND MANAGING DEPENDENCIES IN GLOBAL SOFTWARE DEVELOPMENT 349
by Marcelo Cataldo

Why Is Coordination a Challenge in GSD? 350
Dependencies and Their Socio-Technical Duality 351
From Research to Practice 362
Future Directions 366

21 HOW EFFECTIVE IS MODULARIZATION? 373
by Neil Thomas and Gail Murphy

The Systems 374
What Is a Change? 376
What Is a Module? 381
The Results 383
Threats to Validity 389
Summary 390

22 THE EVIDENCE FOR DESIGN PATTERNS 393
by Walter Tichy

Design Pattern Examples 394
Why Might Design Patterns Work? 397
The First Experiment: Testing Pattern Documentation 398
The Second Experiment: Comparing Pattern Solutions to Simpler Ones 403
The Third Experiment: Patterns in Team Communication 407
Lessons Learned 410
Conclusions 412
Acknowledgments 413

23 EVIDENCE-BASED FAILURE PREDICTION 415
by Nachiappan Nagappan and Thomas Ball

Introduction 416
Code Coverage 417
Code Churn 418
Code Complexity 421
Code Dependencies 422
People and Organizational Measures 423
Integrated Approach for Prediction of Failures 426
Summary 430
Acknowledgments 432

24 THE ART OF COLLECTING BUG REPORTS 435
by Rahul Premraj and Thomas Zimmermann

Good and Bad Bug Reports 436
What Makes a Good Bug Report? 437
Survey Results 439
Evidence for an Information Mismatch 441
Problems with Bug Reports 444
The Value of Duplicate Bug Reports 445
Not All Bug Reports Get Fixed 448
Conclusions 449
Acknowledgments 450

25 WHERE DO MOST SOFTWARE FLAWS COME FROM? 453
by Dewayne Perry

Studying Software Flaws 454
Context of the Study 455
Phase 1: Overall Survey 456
Phase 2: Design/Code Fault Survey 462
What Should You Believe About These Results? 486
What Have We Learned? 490
Acknowledgments 492

26 NOVICE PROFESSIONALS: RECENT GRADUATES IN A FIRST SOFTWARE ENGINEERING JOB 495
by Andrew Begel and Beth Simon

Study Methodology 497
Software Development Task 501
Strengths and Weaknesses of Novice Software Developers 505
Reflections 507
Misconceptions That Hinder Learning 509
Reflecting on Pedagogy 510
Implications for Change 512

27 MINING YOUR OWN EVIDENCE 517
by Kim Sebastian Herzig and Andreas Zeller

What Is There to Mine? 518

Designing a Study 518
A Mining Primer 519
Where to Go from Here 526
Acknowledgments 528

28 COPY-PASTE AS A PRINCIPLED ENGINEERING TOOL 531
by Michael Godfrey and Cory Kapser

An Example of Code Cloning 532
Detecting Clones in Software 533
Investigating the Practice of Code Cloning 535
Our Study 540
Conclusions 543

29 HOW USABLE ARE YOUR APIS? 545
by Steven Clarke

Why Is It Important to Study API Usability? 546
First Attempts at Studying API Usability 548
If At First You Don't Succeed... 552
Adapting to Different Work Styles 559
Conclusion 563

30 WHAT DOES 10X MEAN? MEASURING VARIATIONS IN PROGRAMMER PRODUCTIVITY 567
by Steve McConnell

Individual Productivity Variation in Software Development 567
Issues in Measuring Productivity of Individual Programmers 569
Team Productivity Variation in Software Development 571

A CONTRIBUTORS 575

INDEX 587