Perspectives on Data Science for Software Engineering

Edited by

Tim Menzies

Laurie Williams

Thomas Zimmermann

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

ELSEVIER

Morgan Kaufmann is an imprint of Elsevier

Contents

Contributors xxiii
Acknowledgments xxix

INTRODUCTION

Perspectives on Data Science for Software Engineering 3
T. Menzies, L. Williams and T. Zimmermann
Why This Book? 3
About This Book 5
The Future 6
References 6

Software Analytics and its Application in Practice 7
Dongmei Zhang and Tao Xie
Six Perspectives of Software Analytics 7
Experiences in Putting Software Analytics into Practice 9
References 10

Seven Principles of Inductive Software Engineering: What we do is different 13
T. Menzies
Different and Important 13
Principle #1: Humans Before Algorithms 13
Principle #2: Plan for Scale 14
Principle #3: Get Early Feedback 15
Principle #4: Be Open Minded 15
Principle #5: Be Smart with Your Learning 15
Principle #6: Live with the Data You Have 16
Principle #7: Develop a Broad Skill Set That Uses a Big Toolkit 17
References 17

The Need for Data Analysis Patterns (in Software Engineering).... 19
B. Russo
The Remedy Metaphor 19
Software Engineering Data 20
Needs of Data Analysis Patterns 21
Building Remedies for Data Analysis in Software Engineering Research 21
References 23

From Software Data to Software Theory: The Path Less Traveled 25
J. Whitehead

Pathways of Software Repository Research 25
From Observation, to Theory, to Practice 26
References 28

Why Theory Matters 29
D.I.K. Sj0berg, G.R. Bergersen and T. Dyba

Introduction 29
How to Use Theory 30
How to Build Theory 30
Constructs 31
Propositions 31
Explanation 32
Scope 32
In Summary: Find a Theory or Build One Yourself 32
Further Reading 33

SUCCESS STORIES/APPLICATIONS

Mining Apps for Anomalies 37
A. Zeller

The Million-Dollar Question 37
App Mining 38
Detecting Abnormal Behavior 39
A Treasure Trove of Data 40
… But Also Obstacles 41
Executive Summary 42
Further Reading 42

Embrace Dynamic Artifacts 43
Venkatesh-Prasad Ranganath

Can We Minimize the USB Driver Test Suite? 43
Yes, Let's Observe Interactions 44
Why Did Our Solution Work? 44
Still Not Convinced? Here's More 45
Dynamic Artifacts are Here to Stay 45
Acknowledgments 45
References 46

Mobile App Store Analytics 47
M. Nagappan and E. Shihab
Introduction 47
Understanding End Users 48
Conclusion 49
References 49

The Naturalness of Software 51
E.T. Barr and P. Devanbu
Introduction 51
Transforming Software Practice 53
Porting and Translation 53
The "Natural Linguistics" of Code 53
Analysis and Tools 54
Assistive Technologies 54
Conclusion 55
References 55

Advances in Release Readiness 57
P. Rotella
Predictive Test Metrics 58
Universal Release Criteria Model 59
Best Estimation Technique 60
Resource/Schedule/Content Model 60
Using Models in Release Management 61
Research to Implementation: A Difficult (But Rewarding) Journey 62

How to Tame Your Online Services 63
Qingwei Lin, Jian-Guang Lou, Hongyu Zhang and Dongmei Zhang
Background 63
Service Analysis Studio 64
Success Story 65
References 65

Measuring Individual Productivity 67
T. Fritz
No Single and Simple Best Metric for Success/Productivity 68
Measure the Process, Not Just the Outcome 68
Allow for Measures to Evolve 69
Goodhart's Law and the Effect of Measuring 69

How to Measure Individual Productivity? 70
References 71

Stack Traces Reveal Attack Surfaces 73
C. Theisen and L. Williams
Another Use of Stack Traces? 73
Attack Surface Approximation 75
References 76

Visual Analytics for Software Engineering Data 77
Zhitao Hou, Hongyu Zhang, Haidong Zhang and Dongmei Zhang
References 80

Gameplay Data Plays Nicer When Divided into Cohorts 81
J. Huang
Cohort Analysis as a Tool for Gameplay Data 81
Play to Lose 82
Forming Cohorts 82
Case Studies of Gameplay Data 83
Challenges of Using Cohorts 83
Summary 84
References 84

A Success Story in Applying Data Science in Practice 85
A. Bener, B. Turhan, A. Tosun, B. Caglayan and E. Kocaguneli
Overview 86
Analytics Process 87
Data Collection 88
Exploratory Data Analysis 88
Model Selection 88
Performance Measures and Benefit Analysis 89
Communication Process—Best Practices 89
Problem Selection 89
Managerial Support 89
Project Management 89
Trusted Relationship 89
Summary 89
References 90

There's Never Enough Time to do all the Testing you Want 91
K. Herzig
The Impact of Short Release Cycles
(There's Not Enough Time) 91

Testing is More Than Functional Correctness (All the Testing You Want) 92
Learn From Your Test Execution History 92
Test Effectiveness 93
Test Reliability/Not Every Test Failure Points to a Defect 93
The Art of Testing Less 93
Without Sacrificing Code Quality 94
Tests Evolve Over Time 94
In Summary 94
References 95

The Perils of Energy Mining: Measure a Bunch, Compare Just Once 97
A. Hindle
A Tale of Two HTTPs 97
Let's ENERGISE Your Software Energy Experiments 98
Environment 99
N-Versions 99
Energy or Power 99
Repeat! 99
Granularity 100
Idle Measurement 100
Statistical Analysis 101
Exceptions 101
Summary 101
References 101

Identifying Fault-prone Files in Large Industrial Software Systems 103
E. Weyuker and T. Ostrand
Acknowledgment 106
References 106

A Tailored Suit: The Big Opportunity in Personalizing Issue Tracking 107
O. Baysal
Many Choices, Nothing Great 107
The Need for Personalization 108
Developer Dashboards or "A Tailored Suit" 109
Room for Improvement 109
References 110

What Counts is Decisions, Not Numbers—Toward an Analytics Design Sheet 111
G. Ruhe and M. Nayebi

Decisions Everywhere 111
The Decision-Making Process 112
The Analytics Design Sheet 112
Example: App Store Release Analysis 113
References 114

A Large Ecosystem Study to Understand the Effect of Programming languages on Code Quality 115
B. Ray and D. Posnett

Comparing Languages 115
Study Design and Analysis 116
Results 117
Summary 117
References 118

Code Reviews are not for Finding Defects—Even Established Tools Need Occasional Evaluation 119
J. Czerwonka

Results 120
Effects 121
Conclusions 122
References 122

TECHNIQUES

Interviews 125
C. Bird

Why Interview? 125
The Interview Guide 126
Selecting Interviewees 127
Recruitment 127
Collecting Background Data 128
Conducting the Interview 128
Post-Interview Discussion and Notes 129
Transcription 129
Analysis 130
Reporting 130
Now Go Interview! 131
References 131

Look for State Transitions in Temporal Data 133
R. Holmes
Bikeshedding in Software Engineering 133
Summarizing Temporal Data 133
Recommendations 135
Reference 135

Card-sorting: From Text to Themes 137
T. Zimmermann
Preparation Phase 138
Execution Phase 139
Analysis Phase 140
References 141

Tools! Tools! We Need Tools! 143
D. Spinellis
Tools in Science 143
The Tools We Need 144
Recommendations for Tool Building 146
References 147

Evidence-based Software Engineering 149
T. Dyba, G.R. Bergersen and D.I.K. Sj0berg
Introduction 149
The Aim and Methodology of EBSE 150
Contextualizing Evidence 150
Strength of Evidence 151
Evidence and Theory 152
References 152

Which Machine Learning Method do you Need? 155
L.L. Minku
Learning Styles 155
Do Additional Data Arrive Over Time? 156
Are Changes Likely to Happen Over Time? 156
If You Have a Prediction Problem, What Do You Really Need to Predict? 157
Do You Have a Prediction Problem Where Unlabeled Data are Abundant and Labeled Data are Expensive? 157
Are Your Data Imbalanced? 158
Do You Need to Use Data From Different Sources? 158
Do You Have Big Data? 158

Do You Have Little Data? 158
In Summary 158
References 159

Structure your Unstructured Data First!: The Case of Summarizing Unstructured Data with Tag Clouds 161
A. Bacchelli

Unstructured Data in Software Engineering 161
Summarizing Unstructured Software Data 162
As Simple as Possible... But not Simpler! 162
You Need Structure! 164
Conclusion 167
References 168

Parse that data! Practical Tips for Preparing your Raw Data for Analysis 169
P. Guo

Use Assertions Everywhere 170
Print Information About Broken Records 170
Use Sets or Counters to Store Occurrences of Categorical Variables 171
Restart Parsing in the Middle of the Data Set 171
Test on a Small Subset of Your Data 172
Redirect Stdout and Stderr to Log Files 172
Store Raw Data Alongside Cleaned Data 172
Finally, Write a Verifier Program to Check the Integrity of Your Cleaned Data 172

Natural Language Processing is No Free Lunch 175
S. Wagner

Natural Language Data in Software Projects 176
Natural Language Processing 176
How to Apply NLP to Software Projects 176
Do Stemming First 177
Check the Level of Abstraction 177
Don't Expect Magic 178
Don't Discard Manual Analysis of Textual Data 178
Summary 179
References 179

Aggregating Empirical Evidence for More Trustworthy Decisions....181
D. Budgen

What's Evidence? 181
What Does Data From Empirical Studies Look Like? 182
The Evidence-Based Paradigm and Systematic Reviews 183
How Far can We Use the Outcomes From Systematic Review to Make Decisions? 184
References 186

If it is Software Engineering, it is (Probably) a Bayesian Factor. 187
A. Bener and A. Tosun

Causing the Future with Bayesian Networks 187
The Need for a Hybrid Approach in Software Analytics 188
Use the Methodology, Not the Model 189
References 190

Becoming Goldilocks: Privacy and Data Sharing in "Just Right" Conditions 193
F. Peters

The "Data Drought" 193
Change is Good 194
Don't Share Everything 195
Share Your Leaders 196
Summary 197
Acknowledgments 197
References 197

The Wisdom of the Crowds in Predictive Modeling for Software Engineering 199
L.L. Minku

The Wisdom of the Crowds 199
So... How is That Related to Predictive Modeling for Software Engineering? 200
Examples of Ensembles and Factors Affecting Their Accuracy 200
Crowds for Transferring Knowledge and Dealing with Changes 201
Crowds for Multiple Goals 202

A Crowd of Insights 202
Ensembles as Versatile Tools 203
References 203

Combining Quantitative and Qualitative Methods (When Mining Software Data) 205
M. Di Penta

Prologue: We Have Solid Empirical Evidence! 205
Correlation is Not Causation and, Even if We Can Claim Causation 206
Collect Your Data: People and Artifacts 207
Source 1: Dig Into Software Artifacts and Data 207
Source 2: Getting Feedback From Developers 208
How Much to Analyze, and How? 209
Build a Theory Upon Your data 209
Conclusion: The Truth is Out There! 210
Suggested Readings 210
References 210

A Process for Surviving Survey Design and Sailing Through Survey Deployment 213
T. Barik and E. Murphy-Hill

The Lure of the Sirens: The Attraction of Surveys .213
Navigating the Open Seas: A Successful Survey Process in Software Engineering .214
Stage 1: Initial Recruitment .215
Stage 2: Interviews .215
Stage 3: Survey Design .216
Stage 4: Survey Piloting .216
Stage 5: Survey Deployment .217
Stage 6: Survey Analysis and Write-Up .218
In Summary .218
Acknowledgments .218
References .219

WISDOM

Log it All? 223
G.C. Murphy

A Parable: The Blind Woman and an Elephant 223
Misinterpreting Phenomenon in Software Engineering 223

Using Data to Expand Perspectives 224
Recommendations 225
References 225

Why Provenance Matters 227
M.W. Godfrey

What's Provenance? 228
What are the Key Entities? 228
What are the Key Tasks? 228
Another Example 230
Looking Ahead 231
References 231

Open from the Beginning 233
G. Gousios

Alitheia Core 233
GHTorrent 234
Why the Difference? 235
Be Open or Be Irrelevant 236
References 237

Reducing Time to Insight 239
T. Carnahan

What is Insight Anyway? 239
Time to Insight 240
The Insight Value Chain 241
What To Do 241
A Warning on Waste 243
References 243

Five Steps for Success: How to Deploy Data Science in Your Organizations 245
M. Kim

Step 1. Choose the Right Questions for the Right Team 246
Step 2. Work Closely with Your Consumers 246
Step 3. Validate and Calibrate Your Data 247
Step 4. Speak Plainly to Give Results Business Value 247
Step 5. Go the Last Mile—Operationalizing Predictive Models 247
References 248

How the Release Process Impacts your Software Analytics 249
Bram Adams
Linking Defect Reports and Code Changes to a Release 250
How the Version Control System can Help 251
References 253
Security Cannot be Measured 255
A. Meneely
Gotcha #1: Security is Negatively Defined 255
Gotcha #2: Having Vulnerabilities is Actually Normal 256
Gotcha #3: "More Vulnerabilities" Does not Always Mean "Less Secure" 256
Gotcha #4: Design Flaws are not Usually Tracked 257
Gotcha #5: Hackers are Innovative Too 258
An Unfair Question 259
Gotchas from Mining Bug Reports 261
S. Just and K. Herzig
Do Bug Reports Describe Code Defects? 262
It's the User That Defines the Work Item Type 262
An Example 263
Who Cares About the Report Categories? 263
How Big is the Problem of "False Bugs"? 263
Do Developers Apply Atomic Changes? 264
Version Control Systems are not Granular Enough 264
How Big is the Problem of "Tangled Changes"? 264
In Summary 265
References 265
Make Visualization Part of your Analysis Process 267
S. Diehl
Leveraging Visualizations: An Example with Software Repository Histories 267
How to Jump the Pitfalls 268
Don't Forget the Developers! (and be Careful with your Assumptions) 271
A. Orso
Disclaimer 271
Background 271
Are We Actually Helping Developers? 272
Some Observations and Recommendations 273

Acknowledgments 274
References 274

Limitations and Context of Research 277
B. Murphy

Small Research Projects 278
The Importance of Being the First Author on a Publication 278
Requirements for Novel Research Techniques 279
Data Quality of Open Source Repositories 279
Lack of Industrial Representatives at Conferences 280
Research From Industry 280
Summary 281

Actionable Metrics are Better Metrics 283
A. Meneely

What Would You Say... I Should DO? 284
The Offenders 284
Number of Bugs 285
Code Churn 285
Actionable Heroes 285
Number of Developers 286
Number of Callees (Coupling) and Number of Parameters 286
Cyclomatic Complexity: An Interesting Case 286
Are Unactionable Metrics Useless? 287
Reference 287

Replicated Results are More Trustworthy 289
M. Shepperd

The Replication Crisis 289
Reproducible Studies 291
Reliability and Validity in Studies 291
So What Should Researchers Do? 292
So What Should Practitioners Do? 292
References 293

Diversity in Software Engineering Research 295
H. Valdivia-Garcia and M. Nagappan

Introduction 295
What is Diversity and Representativeness? 296
What Can We Do About It? 297

Evaluation 297
Evaluating the Sample Selection Technique 297
Evaluating the Sample Coverage Score 298
Recommendations 298
Future Work 298
Reference 298

Once is not Enough: Why we Need Replication 299
N. Juristo

Motivating Example and Tips 299
Exploring the Unknown 300
Types of Empirical Results 301
Do's and Don't's 301
Further Reading 302

Mere numbers Aren't Enough: A Plea for Visualization 303
P. Runeson

Numbers Are Good, but 303
Case Studies on Visualization 303
Product Scoping 304
Regression Test Selection 305
What to Do 306
References 307

Don't Embarrass Yourself: Beware of Bias in Your Data 309
C. Bird

Dewey Defeats Truman 309
Impact of Bias in Software Engineering 310
Identifying Bias 312
Assessing Impact 313
Which Features Should I Look At? 315
References 315

Operational Data are Missing, Incorrect, and Decontextualized 317
A. Mockus

Background 317
Examples 318
Missing Data 318
How to Augment? 318
Incorrect Data 318
How to Correct? 319
Decontextualized Data 319
How to Identify Context 319

A Life of a Defect 320
What to Do? 321
References 322

Data Science Revolution in Process Improvement and Assessment? 323
M. Oivo

Correlation is not Causation (or, When not to Scream "Eureka!") 327
T. Menzies

What Not to Do 327
Example 327
Examples from Software Engineering 328
What to Do 329
In Summary: Wait and Reflect Before You Report 330
References 330

Software Analytics for Small Software Companies: More Questions than Answers 331
R. Robbes

The Reality for Small Software Companies .332
Small Software Companies Projects: Smaller and Shorter .332
Different Goals and Needs .333
What to Do About the Dearth of Data? .333
What to Do on a Tight Budget? .334
References .335

Software Analytics under the Lamp Post (Or What Star Trek Teaches us about the Importance of Asking the Right Questions) 337
N. Medvidovic and A. Orso

Prologue 337
Learning from Data 338
Which Bin Is Mine? 340
Epilogue 340

What can go Wrong in Software Engineering Experiments? 341
S. Vegas and N. Juristo

Operationalize Constructs 342
Evaluate Different Design Alternatives 342
Match Data Analysis and Experimental Design 343
Do Not Rely on Statistical Significance Alone 343
Do a Power Analysis 344

Find Explanations for Results 344
Follow Guidelines for Reporting Experiments 344
Improving the Reliability of Experimental Results 344
Further Reading 345

One Size Does Not Fit All 347
T. Zimmermann

References 348

While Models are Good, Simple Explanations are Better 349
Venkatesh-Prasad Ranganath

How Do We Compare a USB2 Driver to a USB3 Driver? 349
The Issue With Our Initial Approach 350
"Just Tell us What Is Different and Nothing More" 351
Looking Back 351
Users Prefer Simple Explanations 352
Acknowledgments 352
References 352

The White-Shirt Effect: Learning from Failed Expectations 353
L. Prechelt

A Story 353
Revelation 353
Work, Work, Work 354
Disappointment 354
The Right Reaction 355
Practical Advice 356
Always Think of a Causation Model 356
Think of a Causation Model, *Before* You Check the Expectation 356
Be Wary of Failing Expectations 356
Be Ready to Accept Inferior Types of Evidence 356
For Researchers: Know the FNR 357

Simpler Questions Can Lead to Better Insights 359
B. Turhan and K. Kuutti

Introduction 359
Context of the Software Analytics Project 359
Providing Predictions on Buggy Changes 360
How to Read the Graph? 361
(Anti-)Patterns in the Error-Handling Graph 362

How to *Act* on (Anti-)Patterns? 362
Summary 363
References 363

Continuously Experiment to Assess Values Early On 365
J. Munch

Most Ideas Fail to Show Value 365
Every Idea Can Be Tested With an Experiment 366
How Do We Find Good Hypotheses and Conduct the Right Experiments? 367
Key Takeaways 368
Further Reading 368

Lies, Damned Lies, and Analytics: Why Big Data Needs Thick Data 369
M.-A. Storey

How Great It Is, to Have Data Like You 369
Looking for Answers in All the Wrong Places 370
Beware the Reality Distortion Field 370
Build It and They Will Come, but Should We? 371
To Classify Is Human, but Analytics Relies on Algorithms 371
Lean in: How Ethnography Can Improve Software Analytics and Vice Versa 372
Finding the Ethnographer Within 373
References 373

The World is Your Test Suite 375
Andrew J. Ko

Watch the World and Learn 376
Crashes, Hangs, and Bluescreens 376
The Need for Speed 376
Protecting Data and Identity 377
Discovering Confusion and Missing Requirements 377
Monitoring Is Mandatory 378
References 378