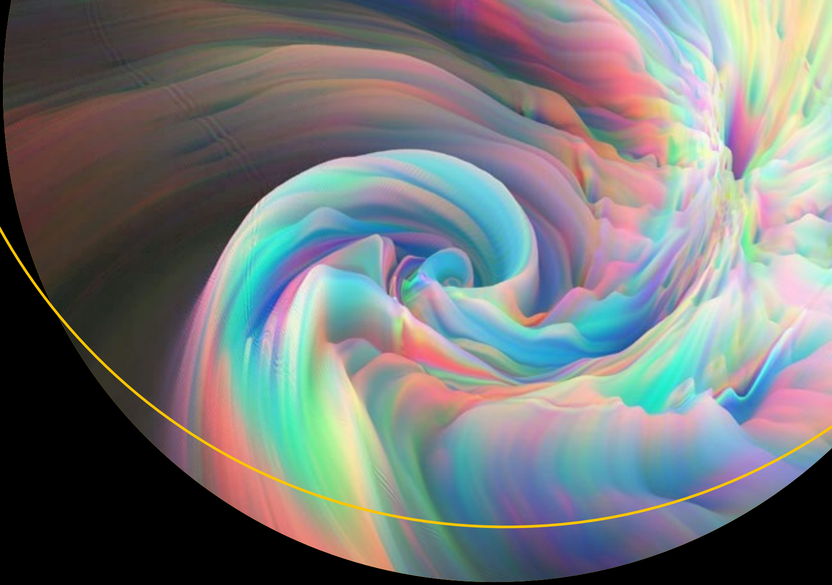

Reinventing ITIL® and DevOps with Digital Transformation

Essential Guidance to Accelerate the Process

Second Edition

Abhinav Krishna Kaiser

Apress®

Reinventing ITIL® and DevOps with Digital Transformation

Essential Guidance to Accelerate the Process

Second Edition

Abhinav Krishna Kaiser

Apress®

Reinventing ITIL® and DevOps with Digital Transformation: Essential Guidance to Accelerate the Process

Abhinav Krishna Kaiser
Bengaluru, India

ISBN-13 (pbk): 978-1-4842-9071-2 ISBN-13 (electronic): 978-1-4842-9072-9
https://doi.org/10.1007/978-1-4842-9072-9

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: James Markham
Coordinating Editor: Mark Powers
Copy Editor: Kezia Endsley

Cover designed by eStudioCalamar

Cover image by Glitch Lab on Unsplash (www.unsplash.com)

To my readers,
whose constant feedback and encouragement
keeps me churning out new books…

Table of Contents

About the Author xxi

About the Technical Reviewer xxiii

Introduction xxv

Part I: ITIL to DevOps 1

Chapter 1: Introduction to DevOps 3

What Exactly Is DevOps? 4

DevOps with an Example 4

Why DevOps? 6

Let's Look at the Scope 8

The Benefits of Transforming into DevOps 9

Insight from the State of DevOps Report 10

DevOps Principles 12

Culture 12

Automation 13

Lean 14

Measurement 14

Sharing 15

Elements of DevOps 15

People 18

DevOps Team 20

The Basis for a DevOps Team 21

An Example of a DevOps Team 21

Processes 22

Continuous Integration 23

Continuous Delivery 26
Continuous Deployment 29
Technology 32
Choosing the Right Tool 32
Categories of Tools 33
Source Code Repositories 34
Hosting Services 34
Orchestrators 35
Deployment and Environment Provisioning 35
Testing 36
Is DevOps the End of Ops? 36
Summary 38
Chapter 2: ITIL Basics 39
IT Service Management and ITIL 39
The Conception of ITIL 40
Competition to ITIL 42
Understanding Services 43
Service Types (Components) 44
Core Services 45
Enabling Services 46
Enhancement Services 46
Understanding Processes 47
Understanding Functions 48
Functions in ITIL 49
Processes vs. Functions 49
ITIL Service Lifecycle 50
Service Strategy 51
Service Strategy Processes 53
Service Design 53
Service Design Processes 54

Service Transition 54
Service Transition Processes 55
Service Operations 55
Service Operations Processes 56
Continual Service Improvement 56
Continual Service Improvement Process 57
ITIL Roles 58
Service Owner 58
Process Owner 59
Process Manager 59
Process Practitioner 60
RACI Matrix 60
An Example of RACI 61
Tips on RACI Creation 62
ITIL V3 and ITIL 4 63
The Service Lifecycle Is Dead 63
Introducing Practices 63
Service Has a New Definition 64
Governance Is a New Kid on the Block 64
Automation Is In 65
Summary 65

Chapter 3: ITIL and DevOps: An Analysis 67

Product vs. Services 68
Big Ticket Conflicts 72
Which Is It: Sequential vs. Concurrent? 72
Let's Discuss Batch Sizes 72
It's All About the Feedback 73
The Silo Culture 73
What Is Configuration Management? 74

Continuous Deployment Makes Release Management Irrelevant 75
Union of Mindsets 76
The Case for ITIL Adaptation with DevOps 77
To Conclude 78
Summary 80

Chapter 4: Integration: Alignment of Processes 81

Analysis of ITIL Phases 81
Analysis: Service Strategy Phase 83
- Strategy Management for IT Services 83
- Service Portfolio Management 86
- Financial Management for IT Services 86
- Demand Management 86
- Business Relationship Management 88

Analysis: Service Design Phase 89
- Design Coordination 89
- Service Catalog Management 93
- Service Level Management 94
- Availability Management 95
- Capacity Management 96
- IT Service Continuity Management 99
- Information Security Management 100
- Supplier Management 103

Analysis: Service Transition Phase 104
- Transition Planning and Support 104
- Change Management 105
- Service Asset and Configuration Management 105
- Release and Deployment Management 105
- Service Validation and Testing 105
- Change Evaluation 106
- Knowledge Management 107

Analysis: Service Operation Phase 108
Event Management 108
Incident Management 110
Request Fulfillment 110
Problem Management 111
Access Management 111
Continual Service Improvement 111
The Seven-Step Improvement Process 112
Summary 113
Chapter 5: Teams and Structures 115
A Plunge Into ITIL Functions 116
Service Desk 116
Technical Management 118
Application Management 119
IT Operations Management 120
DevOps Team Structure Revisited 122
Traditional Model 123
The Agile Model 125
Flat Hierarchy 125
No Project Manager 126
Single Team 126
Product Owner 127
Predictability 128
The DevOps Model 129
Composition of a DevOps Team 129
ITIL Role Mapping in a DevOps World 132
Strategy and Compliance 133
Umbrella Teams 134
Shared Teams 135
DevOps Teams 136
Summary 137

Chapter 6: Managing Configurations in a DevOps Project.................................. 139
ITIL Service Asset and Configuration Management Process.. 139
Objectives and Principles 140
Service Assets and Configuration Items 140
Scope of Service Asset and Configuration Management.. 142
Introducing the CMDB, CMS, DML, and DS 143
Configuration Management Database 143
Configuration Management System 143
Definitive Media Library and Definitive Spares 144
Service Asset and Configuration Management Processes... 145
Step 1: Management and Planning... 146
Step 2: Configuration Identification 147
Step 3: Configuration Control... 148
Step 4: Status Accounting and Reporting 149
Step 5: Verification and Audit 150
Why Configuration Management Is Relevant to DevOps 151
Configuration Management in a DevOps Sense 152
Decoding IaaS 153
Decoding PaaS.. 154
Application Deployment and Configuration... 154
Underlying Configuration Management 155
Automation in Configuration Management 156
Who Manages DevOps Configurations? 157
Comprehensive Configuration Management... 158
Configuration Management Database.. 159
CMDB for Change Management 160
CMDB for Provisioning Environments.. 161
CMDB for Incident Management... 161
Source Code Repository... 161
Basics of a Source Code Repository.. 162
What Can Be Stored in a Source Code Repository?... 162

Good Practices for Achieving DevOps Objectives........ 162
Choosing a Source Code Repository Tool 163
Artifact Repository 165
Managing Binaries 166
Summary........ 167
Chapter 7: Incident Management Adaptation 169
What Is ITIL Incident Management?........ 170
Incident Management Is Vital 170
Incident Management Is the First Line of Defense 171
Digging Deeper Into Incident Management 171
What Constitutes an Incident?........ 172
Who Can Register Incidents?........ 173
A Typical Incident Management Process 174
Step 1: Incident Identification........ 176
Step 2: Incident Logging........ 176
Step 3: Incident Categorization........ 177
Step 4: Incident Prioritization 178
Step 5: Diagnosis and Investigation 178
Step 6: Resolution and Recovery........ 179
Step 7: Incident Closure........ 180
Major Incidents 180
Incident Management in DevOps 181
Agile Project Management 182
DOR and DOD........ 188
Sprint Planning for a DevOps Team........ 188
Plan for What Is Currently on Your Plate........ 190
Keep Some Contingency Aside During the Planning Session........ 190
The Scope of the DevOps Team in Incident Management........ 191
Levels of Support........ 191
Incident Flow 193

Knowledge Management at the Core 194
ITIL's Knowledge Management 194
What Knowledge to Maintain 195
Knowledge Storing and Retrieval 197
The DevOps Incident Management Process 198
Step 1: Incident Identification 200
Step 2: Incident Analysis, Escalation, and Resolution 200
Step 3: Incident with DevOps Team 201
Step 4: Incident Manager Analyzes and Accepts Incidents 201
Steps 5 and 6: The Incident Is Prioritized and Added to the Sprint 202
Steps 7 and 8: The Scrum Team Makes Code Changes and Checks In 203
Step 9: Continuous Integration and Continuous Testing 204
Step 10: Auto Deployment 205
Step 11: Post-Mortem 206
Summary 206

Chapter 8: Problem Management Adaptation 207

Introduction to ITIL Problem Management 207
Objectives and Principles 208
Incidents vs. Problems 209
Key Terminologies in Problem Management 210
Root Cause 210
Root-Cause Analysis 210
Known Error 210
Known Error Database 211
Workarounds 211
Permanent Solutions 211
Problem Analysis Techniques 212
Brainstorming 212
The Five-Why Technique 214
Applying the Five-Why Technique 214
Limitations of the Five-Why Technique 216

The Ishikawa Diagram 217
The Kepner-Tregoe Method........ 220
A Typical Problem Management Process........ 221
Step 1: Problem Detection........ 222
Step 2: Problem Logging 224
Step 3: Problem Categorization 225
Step 4: Problem Prioritization........ 226
Step 5: Problem Investigation and Diagnosis 226
Step 6: Problem Resolution 227
Step 7: Problem Closure 227
Problem Management in DevOps........ 228
What Are the Possible Problems in a DevOps Project? 228
Making the Case for a Problem Manager 229
The DevOps Problem Management Process........ 231
Summary........ 235
Chapter 9: Managing Changes in a DevOps Project 237
What Constitutes a Change?........ 238
Overview of Resources and Capabilities 238
Change in Scope........ 239
Why Is Change Management Critical? 241
Objectives and Scope of ITIL Change Management 242
Types of Changes 243
ITIL Change Management Process........ 246
Step 1: Create a Request for Change........ 248
Step 2: Assess and Evaluate the Change........ 249
Step 3: Authorize the Build and Test It........ 249
Step 4: Build and Test........ 253
Step 5: Authorize the Implementation 253
Step 6: Implement and Verify 253
Step 7: Review and Close the Change........ 254
How Are DevOps Changes Different from ITIL Changes?........ 254

The Perceived Problem with ITIL Change Management.......... 255

DevOps to the Rescue 256

Project Change Management 256

Risk Mitigation Strategies.......... 261

Auto-Deployment and Auto-Checks.......... 261

DevOps Change Management Process 263

Change Management Adaption for Continuous Delivery.......... 263

Steps 1, 2, and 3: Change Initiation 264

Step 4: Build and Test.......... 265

Step 5: Deployment Authorization 265

Steps 6 and 7: Deployment and Verification.......... 265

Continuous Delivery for Maximum Change Governance 265

Change Management Adaption for Continuous Deployment.......... 266

Steps 1, 2, 3, and 5: Change Initiation and Authorization to Deploy 267

Step 4: Build and Test.......... 268

Step 6: Deployment to Production 268

Step 7: Change Verification and Closure.......... 268

Maximum Agility with Standard Changes.......... 269

Championing Standard Changes.......... 269

Process for Identifying and Managing Standard Changes.......... 271

Summary.......... 277

Chapter 10: Release Management in DevOps 279

Change Management vs. Release Management.......... 279

Release Management vs. Release and Deployment Management 281

The Basics of a Release.......... 282

Release Units.......... 282

Release Packages 283

Types of Releases 284

Major Releases.......... 284

Minor Releases.......... 284

Emergency Releases 285

- Early Life Support 285
Deployment Options 286
- The Big Bang Option 286
- The Phased Approach 287
The Four Phases of Release Management 288
- Release and Deployment Planning 288
- Release Build and Test 288
- Deployment 289
- Review and Close 290
Releases in DevOps 290
- Sequential and Iterative Nature of the Process 290
Release Management Process Adaption with Iterations 292
- Using Agile Release Trains 292
- Applying Release Management to Continuous Deployment 292
- Applying Release Management to Continuous Delivery 293
Expectations from Release Management 293
- Blue-Green Deployment 294
- The Scope of Release Management 295
- Automation of Release Management 296
The DevOps Release Management Team 297
- Release Management Team Structure 298
Welcome Release Manager, the Role for All Seasons 299
- Product Owners Are the New Release Managers 302
Summary 303

Part II: DevOps to Digital Transformation 305

Chapter 11: Digital Transformation: The Driver of Business Success 307

DevOps and Beyond 307
The World of Digital Transformation 309
- The Curious Case of Magic Link 310
- What About Google Glass? 311

The Right Questions to Ask 312
Digital Transformation and Business Disruption.. 312
Business Disruption 101.. 314
Does the Disruption Have to Be Big Bang?.. 316
Is Virtual the Assumed Goal? 317
Finding Synergy with Partner Organizations 318
Key Focus Areas.. 319
Customers 320
Value.. 320
Innovation 321
Data 323
Balancing All Things Digital.. 324
Roadmap vs. Agility 324
Planning vs. Experimentation.. 325
In-Housing vs. Collaboration.. 326
Summary.. 326
Chapter 12: The Digital Transformation Framework.. 327
The Battle Tank Framework 327
The Digital Transformation Strategy.. 329
Step 1: Identify Opportunities/Pain Points.. 331
Step 2: Prepare a Game Plan.. 333
Step 3: Seek Out Partners 335
Step 4: Execute a Small Project.. 336
Step 5: Observe, Refine, and Transform.. 338
Culture and Digital Culture.. 340
Innovative 342
Openness.. 343
Collaborative.. 344
Entrepreneurial.. 345
Customer Centric.. 346

People 348
The Coca-Cola Case Study 348
The Psychological Effect of Change 349
Fear of Automation 349
Technology 350
The L'Oréal Case Study 351
L'Oréal vs. Estée Lauder: A Digital Transformation Comparison 352
Techniques and Architectures 352
Golden Practices for Technology Implementation 353
Data 358
DIKW Cycle 359
Summary 361
Chapter 13: People and Leadership 363
Digital Transformation Is People Centric 363
The End of Work as We Know it 365
The Pitfalls of Legacy Working 366
The Talent Code 367
The Productivity Equation 367
The Flexible Model of Working 368
No Fixed Hours or Location Constraints 369
Asynchronous Work 369
Productivity as a KPI 370
Employee Engagement 371
The Framework for the Flexi-Work Model 372
Digital Envisioning 373
Enablement for the Flexible Work Model 374
Work Culture Fitment 375
The MURAL Story 376
Performance Management 377

Leadership in the Digital Age 379
Organization Structure 380
Leadership Style 381
Motivation 384
Responsibility 386
Leadership Levers to Stay Relevant 387
The Customer Is Still King 388
Agile and Nimble 390
Experimentation and Innovation 392
Build the Right Team and Foster People in the New Culture 395
Be Authentic 403
Summary 406
Chapter 14: Techniques and Tools for Managing Digital Teams 407
How Do You Manage Remote Work? 407
Trust the Teams to Deliver 408
Google's Team Effectiveness Study 409
Respect the Team 411
How Do You Hire the Right People? 413
The Fundamental Challenges with Hiring 413
The Automattic Case Study 416
Self-Supervised and Self-Motivated 418
Collaborators and Communicators 421
Managing Virtual Meetings 425
How to Run Virtual Meetings Effectively 427
Summary 431
Chapter 15: Adopting a Product-Led Approach 433
What Exactly Is a Product-Led Approach? 435
Why Should Companies Swivel Around Products? 435
Why Should Products Exist? 438
The DNA of a Product-Led Company 439
Benefits of the Product-Led Approach 442

Understanding Value Streams and Value Stream Mapping 443
    An Introduction to Value Streams 444
    What Is Value Stream Mapping? 446
    Carrying Out Value Stream Mapping 448
Looking at Data and Metrics 454
    The Problem of Perspective in the Digital Age 455
    Operational Metrics 456
    Usage Metrics 457
    Business Metrics 460
Summary 462

Index 463

About the Author

Abhinav Krishna Kaiser is a management consultant and works as a partner in a leading consulting firm. He consults with organizations that are looking to improve, become efficient, and transform. His areas of expertise include Digital Transformation, Product Models, Agile, DevOps, ITIL, and other connected IT areas.

Abhinav is a digital transformation enthusiast who has been instrumental in driving several transformation initiatives across sectors. He has consulted with companies to change their approach from traditional to a product-led model, and these initiatives have driven companies to achieve new heights in reaching higher customer satisfaction levels.

He is one of the leading names synonymous with ITIL, and his previous publication, *Become ITIL 4 Foundation Certified in 7 Days*, is one of the top guides recommended to IT professionals looking to get into the service management field and become ITIL Foundation certified.

Abhinav started consulting with clients several years ago on IT service management, creating value by developing robust service management solutions. He is one of the foremost authorities in the area of configuration management, and his solutions have stood the test of time, rigor, and technological advancements. A natural evolution in consulting, led him from service management to Agile and DevOps and now into digital transformation and product-led models.

Abhinav has trained thousands of IT professionals on DevOps processes, Agile methodologies, and ITIL expert-level certifications. He blogs and writes guides and articles on digital transformation, DevOps, Agile, and ITIL at `http://abhinavpmp.com`. His first book came out in 2015: *Workshop in a Box: Communication Skills for IT Professionals*. He runs a video channel on YouTube that has garnered several thousands of views and acclaim: `https://www.youtube.com/user/abhinavonthetube`.

As consultants go where the clients want them to, Abhinav has traveled across the globe and has lived in the United States, Australia, South Africa, and the United Kingdom, before settling down in Bangalore. He is happily married to Radhika, and they have two children—Anagha (daughter) and Aadwik (son).

About the Technical Reviewer

Rajeev Kesana brings two decades of IT industry experience, partnering with customers at Tech Mahindra, IBM Software Labs, Infosys, Capgemini, & TCS. As an accomplished leader in strategy and transformation, Rajeev has envisioned, designed, and built multiple technology-transformation-focused business units from the ground up over the years. Currently based in Hyderabad, where he resides with his wife, Chaitanya, he enjoys meditation, nature, cooking, and travelling.

Introduction

I am lucky to have worked on various transformation programs in the past 15 years. Although we didn’t call it "transformation" in the 2000s, it disrupted the notions of what was considered normal, the general principles that we applied and the outcomes that we considered are time immemorial. Just as a sculptor takes a piece of monolith and turns it into something beautiful, transformation projects involve fast-paced evolutions that change the current flow into something different.

Transformation is the same in every field. The principles are common across the board. I went through a physical transformation fairly recently. I weighed around 95 kilograms (210 pounds) and I am 177 centimeters (5'8") tall. My body mass was 28 percent fat, according to a machine that measures all kinds of bodily stats. I had accepted that this was my normal; this was who I was and honestly, it never bothered me.

I was introduced to a fitness coach by my wife, who had transformed some of her friends. Although I wasn’t serious at first, I decided to hire him. The regimen involved a combination of diet devoid of voluminous food and activity at the gym. The first couple of weeks were perhaps the most difficult part of transformation, with hunger eating at me. I was asked to cut down my carb intake to a fifth of what I used to consume. No chips. No pizzas. And no beer. As I saw my weight shed on a weekly basis, the cravings disappeared and so did the hunger. I longed to follow my prescribed diet, and I looked forward to my time at the gym. A few months down the line, I had lost around 25 kilograms (55 pounds) and I was down to 17 percent body fat.

The man in the mirror was transformed, but it did not happen overnight. It took a lot of discipline and will power to stay on course. My coach changed my diet every week based on the progress that I made—the importance of measurement and feedback struck me more than ever before. My weight loss was massive to begin with and slowed as the weeks passed, which is the expected to curve for any transformation, physical and digital. I have far from a perfect body, but I am in a much better place than when I started off. Digital transformations are typically like this; they don't end. At any point, you can see how much you have progressed, and new technologies, direction and pivots point towards the path where much more can be achieved.

Reinventing ITIL® and DevOps with Digital Transformation is the second edition of *Reinventing ITIL® in the Age of DevOps*. The first edition received lots of feedback from on-the-ground implementations. The ideas were the first of their kind, and that book provided solutions to thousands of ITIL projects that were moving the DevOps way. As the pandemic hit, digital transformation accelerated, and our notions of work changed with it. I added five chapters in the second edition to address this new level of evolution, whereby DevOps projects started to move into the bigger realm of digital transformation. The original chapters (with some modifications) are presented as Section I, while the new chapters covering digital transformation are in Section II. You can read this book like a story, from cover to cover, or you can use the table of contents to choose topics of interest.

In Chapter 12, I present a framework for strategizing and implementing digital transformations, called the *battle tank framework*. This framework has nothing to do with wars or the army—it illustrates various elements of digital transformation in conjunction with the parts of a battle tank. The final chapter of this book presents a product-led approach, which is quite distinct from the usual ways of working.

PART I

ITIL to DevOps

CHAPTER 1

Introduction to DevOps

New ways of working or new methodologies often come about because of a problem—yes, it all starts with a problem. DevOps too resulted from problems faced by businesses. Businesses craved quick turnarounds to their solutions. Businesses often found, in the midst of development, that they didn't have all the information they needed to make the right decisions. They wanted to make a few more changes to the requirements and still expected the delivery to happen on time. DevOps was born to solve this problem.

DevOps just didn't show up as the DevOps we have today. It evolved over time. It was clear to those who started solving the agility problem that DevOps had a lot of potential to not just solve that problem but also increase productivity by leaps and bounds. Further, the quality of the software developed had the potential to be the best. Thus, to this day, DevOps keeps evolving for the better.

DevOps is not just a methodology for developers. Operations reaps its share of benefits from DevOps as well. With increased automation, operations went from being a mundane job to an innovative one. Operations folks got a new lease on life through various tools that made their working lives a whole lot of fun, and they could look forward to integrating and configuring tools to do advanced stuff, rather than the repetitive workload that's generally associated with operations. Productivity shot up and human errors became much rarer.

Software development was carried out on the back of the software delivery lifecycle (SDLC) and was managed through waterfall project management. On the operations front, ITIL ruled the roost. Through DevOps, development and operations essentially came together to form a union. In the mix, the waterfall methodology gave way to Agile methodologies, and still people who designed DevOps processes did not have a good understanding of how ITIL would come into DevOps. A lot of noise started to circulate that the dawn of DevOps was the end for ITIL. This was plainly noise without any substance; you will learn in this book about the value that ITIL brings to the table and why DevOps cannot exist in its entirety without a framework such as ITIL.

© Abhinav Krishna Kaiser 2023
A. Krishna Kaiser, *Reinventing ITIL® and DevOps with Digital Transformation*,
https://doi.org/10.1007/978-1-4842-9072-9_1

The first part of the book is structured around the ITIL service management framework and explores what changes need to be made to ITIL to ease into DevOps projects. Chapter 4 covers common ITIL processes and ITIL functions with respect to DevOps and Chapters 5 through 10 provide in-depth analysis of major processes in ITIL around DevOps designs and implementations. You can use the book to readily implement ITIL in the most effective manner for it to create value in DevOps projects. The second part of the book shifts gears to transform DevOps a notch higher – into digital transformation.

This chapter briefly explains DevOps, including its principles, elements, and processes. Chapter 2 provides a snapshot of ITIL V3, including its lifecycle, phases, processes, and functions. Chapter 3 analyzes DevOps and ITIL, identifying the commonalities and conflicts that support the journey toward adapting ITIL for DevOps implementations.

What Exactly Is DevOps?

There are multiple perceptions about DevOps in the core. In fact, if you search the web, you will be surprised to find multiple definitions for DevOps. No two definitions have common aspects and elements.

I have trained thousands in the area of DevOps, and the best answer I have is that it combines the development and operations teams, and that's about it. Why does bringing two teams together create such a strong buzz across the globe? In fact, if it actually was just the culmination of two teams, DevOps probably would have been discussed in the human resources ecosphere, and it would have remained a semi-complex HR management process.

During the beginning of the DevOps era, to amuse my curiosity, I spoke to a number of people to understand what DevOps is. Most bent toward automation, some spoke of *that* thing they do in startups, and there were a very few who spoke of it as a cultural change. Interesting! Who talks of culture these days, when the edge of our seats burn a hole if we don't act on our commitments? A particular example made me sit up and start connecting the DevOps dots, and it all made sense eventually.

DevOps with an Example

Let's say that you are a project manager of an Internet banking product. The past weekend you deployed a change to update a critical component of the system after weeks of development and testing. The change was deployed successfully; however, during the post-implementation review, it threw an error that forced you to roll back the change.

The rollback was successful, and all the artifacts pertaining to the release were brought to the table to examine and identify the root cause the following Monday. Now what? The root cause was identified, a developer was pressed into action to fix the bug, and the code went through the scrutiny of various tests, including the tests that were not originally included that could have caught the bug in the functional testing stage rather than in production. All the tests ran okay and a new change was planned. It was approved by the change advisory board, and the change was implemented, tested, and green-lit.

These are the typical process activities that are undertaken when a deployment fails and has to be replanned. However, the moment things go south, what is the first thing that comes to your mind as the project manager? Is it what objective action you should take next, or do you start thinking about the developer who worked in this area, the person responsible for the bug in the first place? Or do you think about the tester who identified the scenarios, wrote the scripts, and performed the exploratory testing? It is true that most people start to think about the people responsible for the mess. Why? It is because of our culture. We live in a culture that blames people and tries to pass the buck.

I mentioned earlier about some respondents telling me that DevOps is about culture. So, what culture am I talking about in the context of this example? The example depicts a culture of blame, where the project manager is trying to pin the blame on the people on their team directly responsible for the failure. They could be factually right in pinning the blame on the people directly responsible, but I am focusing on the practice involving blaming individuals.

How is this practice different from the DevOps culture? In DevOps, the responsibility of completing a task is not considered an individual responsibility but rather a shared one. Although an individual works on a task, if the person fails or succeeds, the entire team gets the carrot or the stick. Individuals are not held responsible when we look at the overall DevOps scheme of things, and we don't blame individuals. We follow a blameless culture. This culture of blamelessness culminates from the fact that we all make mistakes because we are humans after all and far from perfect. We make mistakes. So, what's the point of blaming people? In fact, we expect that people do make mistakes, not based on negligence but from the experimentation mindset. This acceptance (of developers making mistakes) has led us to develop a system where the mistakes are identified and rectified in the developmental stages, way before they reach production.

How is this system (to catch mistakes) built? To make it happen, we brought the development and operations teams together (to avoid disconnect), we developed processes that are far more effective and efficient than what is out there (discussed in the rest of the book), and finally we took umbrage under automation to efficiently provide feedback on how we are doing (as speed is one of the main objectives we intend to achieve).

DevOps is a common phrase, and with its spread reaching far and wide, there are multiple definitions coming from various quarters. No two definitions are alike, but they do have a common theme: culture. So, for me, DevOps is a cultural transformation that brings people together from across disciplines. They work under a single umbrella to collaborate as one unit with an open mind and to remove inefficiencies.

Note A blameless culture does not mean that the individuals who make repeated mistakes do so without repercussions. Individuals are appraised justly and appropriately and in a constructive manner.

Why DevOps?

What gave rise to a new culture called DevOps, you might ask? The answer is evolution. If you take a timeline view of software, from the 1960s up to the advent of the internet, developing software was equivalent to building a project or launching a space shuttle. It required meticulous planning and activities that were planned to be executed sequentially. The waterfall project management methodology was thus born with five sequential steps, as indicated in Figure 1-1.

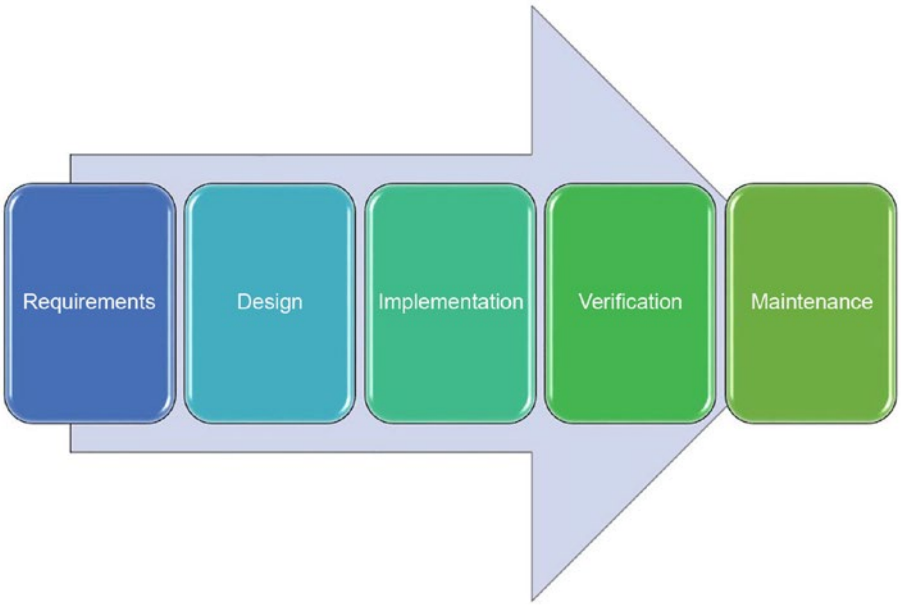

Figure 1-1. *Waterfall project management methodology*

When the Internet boomed, software was far more accessible, and this generated great demand. When the software industry started to expand, the waterfall model's limitations were exposed. The need to complete a detailed planning exercise and the sequential practice of flow seemed like an impediment to the advancement of the software industry.

Then in 2001, at a ski resort in Utah, the Agile Manifesto was born. A number of prevalent Agile methodologies came together to form a common goal that would remove the cast-in-stone waterfall sequential activities.

Agile was more fluid because all its requirements were not conceived at the beginning. It was an approach that was based on iterations, where all the project management activities just cycled over and over again. In between, if a requirement changed, that was okay because there were provisions to make changes that were not bureaucratic nor tedious in nature. In fact, the Agile methodology places emphasis on the response to changes in requirements rather than any map to be followed.

The flexibility and dynamism that came about through Agile spread its wings across the software industry. A number of software projects migrated to the Agile way of working, and to this day, there are projects that are undergoing serious coaching during this transformational phase.