Lean DevOps

A Practical Guide to **On Demand Service Delivery**

Robert Benefield

Lean DevOps

This page intentionally left blank

Lean DevOps

A Practical Guide to On Demand Service Delivery

Robert Benefield

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi
Mexico City • São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Library of Congress Control Number: 2022937306

Cover image: Vinap/Shutterstock

ISBN-13: 978-0-13-384750-5
ISBN-10: 0-13-384750-0

ScoutAutomatedPrintCode

Author: To my beautiful love, Gabrielle, my "little people" Aidan and Talia, and all of those who have enriched my life and, in the process, made the world a better place.

This page intentionally left blank

Contents

Foreword

Just about everyone can remember being inconvenienced by a computer outage at sometime during their lives. I particularly remember walking into the Melbourne airport on a sunny Saturday morning and discovering that the computer which ran all the check-in terminals was down. It stayed down all morning, exposing just how time-consuming and awkward the manual check-in process was. Some of us made it to our destinations many hours late; others were not so lucky. And I remember the hours-long check-in lines at the Baltimore airport many years later; the computer running Southwest's check-in kiosks was out and this time there was no manual backup process. After multiple long lines, we raced to our gate just in time to board before the door closed on a half-empty plane.

As I stood in those lines, I imagined the scene in the data center where "emergency responders" were no doubt scrambling to bring the system back up; after all, I had been there. Some years ago, I managed a factory data center, and we knew that any outage longer than a half hour would seriously curtail pack-out. It only happened once, and believe me, that was one time too many.

At last, we have a book written for the people on the emergency response side of a computer outage. It's not a book focused on being agile or lean; it's a book that focuses on being ready: ready to prevent serious service delivery problems, ready to limit the damage when they happen, and ready to uncover the cause of any incident and keep it from happening again. You would think, given our increasing reliance on technology, that there would be a lot of information about how to do these things well. But I haven't seen a lot of guidance for those who hold this challenging job, at least until now.

We have decades, if not centuries, of experience handling emergency situations in fire departments, hospitals, and military organizations. But rarely have we thought to apply that experience to the technical world, probably because we don't consider a technical failure to be a matter of life or death. But as Robert demonstrates throughout this book, lessons from other domains can be surprisingly relevant for those seeking to prevent computer

outages and, when they do (inevitably) occur, limit their damage and ensure a rapid, safe recovery.

Service delivery teams do a lot more than minimize outages and smoothly handle any that may arise; they provide a critical interface between their organization and its consumers. This book is based on a simple premise—the purpose of a technical system is to bring about expected outcomes for its consumers, and the purpose of service delivery teams is to ensure those expectations are met. They do this by understanding consumers' intent when using the service, discovering what prevents consumers from achieving the outcomes they expect, and learning how to improve the system and close the gaps.

This book is not about meeting service delivery targets; it's about paying attention: paying attention to consumers, being aware of their expectations, and being attentive to their frustrations. It's about paying attention to the ways in which complex technical systems interact, how information flows through organizations, and how decisions are made and executed. It's about paying attention to and improving the way work gets done and the outcomes that are delivered.

Be warned that focusing on delivering outcomes is a novel concept for IT teams, because historically, there was a large time and distance gap between technical teams and their consumers. Most "standard" IT practices, including Agile practices, presume that an intermediary translates consumer intent into proxy features and goals, which then guide the decisions of the technical team. This book eliminates such proxies, and in doing so, it may contradict some of your views on appropriate roles, responsibilities, and processes. Therefore, you might want to check your confirmation bias at the door as you enter, because, as the saying goes, "It ain't what you don't know that gets you into trouble. It's what you know for sure that just ain't so."

This is an important book. It is well written and engaging, with great stories and easy-to-understand analogies. It simultaneously challenges readers and offers very practical advice. I highly recommend it.

Mary Poppendieck
January, 2022

Register your copy of *Lean DevOps* on the InformIT site for convenient access to updates and/or corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account. Enter the product ISBN (9780133847505) and click Submit. Look on the Registered Products tab for an Access Bonus Content link next to this product, and follow that link to access any available bonus materials. If you would like to be notified of exclusive offers on new editions and updates, please check the box to receive email from us.

Figure Credits

Cover image:	Vinap/Shutterstock
Figures 2.1, 2.2, 2.4–2.11, 3.1–3.9, 4.1–4.7, 4.9–4.17, 5.1–5.11, 6.1–6.10, 7.1–7.4, 8.1–8.8, 9.2–9.8, 10.1–10.6, 11.1–11.8, 11.10, 12.1–12.9, 13.1–13.8, 14.1–14.7, 15.1–15.4:	Courtesy of Gabrielle Benefield
Figure 2.3:	John R. Boyd

Acknowledgments

I am far from a natural at writing a book. Its linear format feels overly constraining to the way that I think. Books also feel like a permanent record, while life to me feels more like a constant quest to test ideas, gain knowledge, and correct mistakes to improve yourself and, if you are lucky, maybe make the world a little bit better along the way.

This made creating this book rather arduous. In my desire to help you, the reader, get the most coherent message, I spent an inordinate amount of time rewriting and shuffling content around. I probably would still be at it today if it weren't for the immense help and patience of several key people along the way.

The person most central to helping turn this book into reality, and the one who drew the great illustrations in this book, is my wife and partner-in-crime Gabrielle Benefield. With her product innovation background and my delivery experience, we constantly challenge each other to understand what so often goes wrong with teams and organizations in order to find the best way to help others sustainably deliver the outcomes that really matter. This book, along with the Mobius Loop innovation framework, have come out of years of deep discussion and collaboration to bring out the best in people in order to overcome the problems and dysfunction that so often thwart organizational success.

I also cannot thank Mary and Tom Poppendieck enough for their help and inspiration throughout this process. Both so generously spent the time challenging me to dig deep into my experiences to try to best convey to the world *why* the key pillars of this book are so critical to DevOps success.

I want to extend a special thanks to Colonel David A Hopley OBE RM (Retd). I learned and adopted many of the principles of Mission Command by working alongside many talented military veterans early on in my career. Despite my hazy awareness of their origins and understanding of the nuances that make them so effective, the approach not only resonated with

me but always seemed to work better than traditionally run teams. Being the formal commanding officer of the UK Special Forces (SBS) and as a member of the teaching staff at the Joint Services' Defence College, Colonel Hopley helped improve upon my lay description of Mission Command.

I also want to thank those who hung in through thick and thin reviewing the many iterations of the manuscript, some who also spent several hours helping me clarify my often-jumbled thoughts. This includes: Steve Freeman, James Duncan Davidson, David McNab, Jeff Sutherland, and Tim Beattie. I am also grateful for the patience of the editorial and production team at Addison-Wesley, including: Haze Humbert, Chris Guzikowski, and Menka Mehta. Your help turned this book into something that I am proud of.

Finally, there are a number of people who were instrumental as sounding boards to help clarify many of the ideas in this book, including (in no particular order): Peter Webb, Priit Kaasik, Heikko Ellermaa, Martin Sulg, Ainar Sepp, Sarah-Jane Mason, Chris Matts, Ademar Aguiar, and Henrik Kniberg. You helped me express *why* the approaches in this book not only work, but can help others on their own journey to learn, improve, and maybe make the world a little bit better for all of us.

About the Author

Robert Benefield is an experienced technical leader who has decades of experience delivering robust on-demand services to solve hard problems in demanding ecosystems including banking and securities trading, medical and pharmaceutical, energy, telecom, government, and Internet services. His continual eagerness to learn and work with others to make a difference has taken him from building computers and writing code in the early days of the Internet at Silicon Valley startups to the executive suite in large multinational companies. He shares his unique experience in the hopes that others can continue to build on it without having to collect quite as many scars along the way.

Introduction

Delivering on-demand services well is never easy. Your success hinges on having both the capability and capacity to deliver what your customer needs while doing so at high speed with the consistency, reliability, security, privacy, and cost effectiveness that they expect. This is just as true whether you are providing an IT service or a more traditional courier or electric utility service.

However, unlike more traditional services, IT service providers are far less restricted by organizational size or physical location. With so many quickly deployable tools and cloud capabilities available, even the smallest IT service providers can now instantly scale to address nearly any identified market need globally.

Where IT service delivery providers do struggle is predictably and reliably delivering services that match customer expectations. This, of course, matters. No one wants the frustration and disappointment of a service that falls short of what is needed. What makes this particularly frustrating is that such shortfalls are not caused by misunderstanding the market need or the functionality customers are looking for. If anything, IT is flooded by tools and techniques that allow businesses to analyze and validate ideas quickly. Instead, the problems arise from awareness gaps caused by the way organizations deliver and manage the services themselves.

As IT service stacks grow in complexity, it becomes far more difficult to determine, let alone ensure, that the dynamics between service components and the delivery ecosystem match what the customer expected. Rather than put measures in place to improve their awareness and understanding of these dynamics, delivery teams have focused on other factors like delivering more faster, using the latest cloud technologies and architectural approaches, or adopting the process or methodology most in fashion. Unfortunately, in the process the delivery teams unknowingly create further disconnects that fragment the information flow and context necessary to understand those dynamics.

As the resulting gap between what delivery teams *believe* they are providing and what is *actually* delivered grows, the team's ability to maintain sufficient context to make effective decisions steadily degrades. Even when disconnects are found, organizations often double down on more processes and misunderstood tooling that do little to effectively bridge the gaps. This creates a vicious loop that creates more frustration as the team drifts further away from being able to deliver to meet customer expectations.

Learning How to See

It is not inevitable that delivery teams have to fall into such a dysfunctional spiral. To break the cycle, you first need to understand the many ways you can lose your situational awareness, from deeply entrenched bad habits that fragment information flow to biases and perceptions that distort your understanding of a situation and what is important. Only then can you begin to put measures in place to counteract these tendencies and improve everyone's situational awareness.

Sharpening your situational awareness is like gaining a new sense or superpower you never knew you had. I like to think of it as learning how to see.

The primary objective of this book is to help you on that journey so that you and your organization can close the awareness gap and deliver services that your customers can use to reach their target outcomes. This book is geared primarily for two audiences. The first comprises the individual contributors, like software developers and IT Operations staff, who are in the trenches delivering the services. The other key audience comprises the managers and leaders who are responsible for building and directing those delivery teams.

For individual contributors the journey begins by looking at the delivery process itself. The first step is how you determine the objective of the work you are performing. Can it be used to check how well what is delivered aligns with the target outcomes of the customer, or are the measures more output-focused, such as the number of features or service uptime? Then there are the ways you acquire, understand, learn, and improve your ability to deliver. There are a number of misperceptions that inject flaws into our decision making, and ultimately the effectiveness of our actions.

To break this cycle, in this book you will find various techniques to help you measure and improve your situational awareness and the quality of information flow across your organization so that you are able to make better delivery decisions that move your services closer toward meeting your customer's target outcomes. Along the way many of the excuses people make for not changing their behavior and way of working, from managing work to governance procedures, will be debunked so that you and the team can continue to make progress.

Here, managers and delivery leadership will find strategies to help delivery teams spot and eliminate awareness gaps and misalignments that hinder effective delivery. This begins by identifying the various problems that arise from many of the management styles, requirements management techniques, processes, communication styles, and incentive structures that have traditionally been relied upon to direct and control people. These lead to poor decision making, conflict, reduced learning and improvement, and ultimately failure to deliver in a way that meets customer expectations. You will also learn about the power of Mission Command, as well as ways to communicate, inspire, and support the members of your team to effectively deliver to the organization's vision and the outcomes customers are trying to achieve.

Those who do not fit neatly in one of the two audiences described likely will also find value within these pages. For example, you might uncover and correct your own misperceptions about service delivery. This can help you better understand and more effectively interact with service delivery teams.

The thinking and techniques in this book are part of the larger Mobius outcome delivery approach, which you can find at https://mobiusloop.com. Mobius has been developed by a community dedicated to harnessing the power of innovation and delivery excellence to more effectively achieve the outcomes that matter.

How to Use This Book

This book can be divided into three parts. The first part introduces the key dynamics that underlie the service delivery challenge. It sets the scene for how those of us in IT service delivery are constantly in danger of focusing

far too much on removing delivery friction or reducing perceived delivery risk, often at the cost of maintaining situational awareness and ensuring teams have the ability to learn and improve.

Understanding these dynamics is important for any IT service delivery organization, and especially for those that wish to pursue the promise of DevOps. Overlooking them and missing their effects are what causes so many who pursue DevOps and Agile delivery approaches to fail to meet their promise from the start. This lack of awareness and appreciation of the way they can distort how we perceive the delivery ecosystem is also where many automation tooling and artificial intelligence/machine learning (AI/ML) approaches so often fail.

The second part of the book dives into each of the key elements and the role they play in service delivery. It explores their importance, how they are so often misapplied, and the repercussions to service delivery and the team. I personally feel that this is the most important part of the book, and the one that is so often missing from most guides out there.

The third and last part of the book is a practical guide to help you improve your own service delivery effectiveness. It includes ways to determine the maturity of your team to ensure you have the key elements in place to deliver consistently and effectively. It also has a number of suggestions for how to organize and manage the flow of work, build and deploy instrumentation and automation solutions, and deal with governance associated with internal controls and those required to meet legal and regulatory requirements.

My Own Journey

This book draws from my own journey working in the trenches as an individual contributor and, later, a technical leader to build great IT services and improve the effectiveness of the teams delivering them. I know from firsthand experience that every ecosystem has different challenges, and what I have learned from my own missteps along the way has kept me level-headed and practical. More than anything, I want this book to be a useful addition to your bookshelf for a long time. This is why the focus of this

book is to help you better understand your circumstances so that you and your organization can deliver more effectively, not to talk about some specific process or set of technology that will quickly be supplanted by the next big fad.

I have been blessed throughout my career to have met and worked alongside a number of people far smarter than I am who early on in my career exposed me to revolutionary concepts and ways of working. Some had worked alongside John Boyd's "Fighter Mafia." Others were Training Within Industry (TWI) veterans, or had to come up with ways to deliver highly reliable services long before the existence of concepts like cloud computing or continuous delivery. Only later did I realize that what I learned along the way is what has allowed me to quickly cut through delivery ecosystem noise to help teams overcome seemingly intractable problems. At times it has felt like a superpower, one that I hope I can share to help you reduce your own delivery pain and frustration to secure success.

This page intentionally left blank

Chapter 2

How We Make Decisions

Prepare for the unknown by studying how others in the past have coped with the unforeseeable and the unpredictable.

George S. Patton

Decisions act as our steering wheel that guides us through the pathways of life. This is true whether a decision is simple, like deciding whether to have a cup of coffee, or complex, such as choosing the best architectural design for a new critical service. Decisions are also how we guide our own actions when delivering IT services, whether it is in how we approach coding a new feature or troubleshooting a production problem.

However, being effective at decision making isn't an innate skill. In fact, it is one that is surprisingly difficult to learn. For starters, effectiveness is more than how fast you decide or how adeptly you execute that decision. It must also achieve its objective while accounting for any conditions that might change the dynamics and thereby the outcome trajectory of your actions in the executing ecosystem.

In this chapter, we take a deeper look at the decision-making process itself. We will also explore the ingredients necessary for effective decision making, how they impact the decision process, and how they can become impaired.

Examining the Decision-Making Process

Decision making is the process of pulling together any *information* and *context* about our situation and evaluating it against the *capabilities* available to progress toward the *desired outcome*. This is an iterative, rapid process. To work well we have to determine with each cycle whether the decision progressed us toward the outcome. If so, how effective was it, and if not, why not? We also have to look to see if anything unexpected occurred that can tell us more about the situation and the efficacy of our current capabilities that we can use to adjust and adapt.

Figure 2.1
Figuring out the right mix of context and capabilities for decision making can be challenging.

Even though we make decisions all the time, the process for making them can be surprisingly complex and fraught with mistakes. Consider the example of taking a friend to a coffee shop. While the task is inherently simple, there are all sorts of elements involved that, without the right level of scrutiny, can cause problems. You may find that you have the wrong address (*wrong or incomplete information*), that you took a wrong turn because you thought you were on a different street (*flawed situational context*), or that your car is having engine trouble and the coffee shop is too far to walk to

(*mismatched capabilities*). It is also possible that your capabilities, context, and information to get to the coffee shop are all fine, but your friend is angry because the shop has no Internet service and she only agreed to go there with you because she assumed she would be able to get online (*misunderstood target outcome*).

Spotting and rectifying mistakes under such simple conditions is easy. However, as the setting becomes far more complex, particularly as decisions take the form of large chains like those needed in IT service delivery, mistakes can far more easily hide under several layers of interactions, where they can remain undiscovered all while causing seemingly intractable problems. As these mistakes mount, they steadily degrade our understanding of our delivery ecosystem in ways that, unless found, undermine the overall effectiveness of future decisions.

The military strategist John Boyd became captivated by the importance of decision making while trying to understand what factors determined the likelihood of success in combat. He studied how simple mistakes could cascade and destroy any advantage a unit might have, and sought ways to improve decision-making processes in order to create a strategic advantage over the enemy. His work soon came to revolutionize how elite units, and many Western militaries, began to approach warfare.

Boyd and the Decision Process

Like many of us, John Boyd began his search for what factors increased the likelihood for success by looking at the tools (in this case weapons) of the victor. Boyd was an American fighter pilot who had served in the Korean War, where he flew the highly regarded F-86 fighter jet that dominated the skies against Soviet-designed MiG-15s. He knew firsthand there were differences in each aircraft model's capabilities. He theorized that there must be a way to quantitatively calculate an aircraft's performance so that it could be used to compare the relative performance of different types. If this were possible, one could then determine both the optimal design and the combat maneuvers that would be the most advantageous against the enemy.

His work led to the discovery of Energy-Maneuverability (EM) theory. It modeled an aircraft's specific energy to determine its relative performance against an enemy. The formula was revolutionary and is still used today in the design of military aircraft.

From there, Boyd looked to combine his theory with the optimal processes to fully exploit an aircraft's capabilities. He used his time at the Fighter Weapons School in Nevada to develop *Aerial Attack Study*, a book of aerial combat maneuvers first released inside the US military in 1961. It is considered so comprehensive that it is still used by combat pilots as the definitive source today.

Having both a means to create the best tools and processes to use them, most of us would figure that Boyd now possessed the formula for success in warfare. Despite all of this, Boyd was still troubled.

When he ran his own formula against some of the most successful weapons of World War II and the Korean War, he found many instances where the "successful" ones were far less capable than those of the enemy. Particularly disturbing to him, this included the highly regarded F-86.

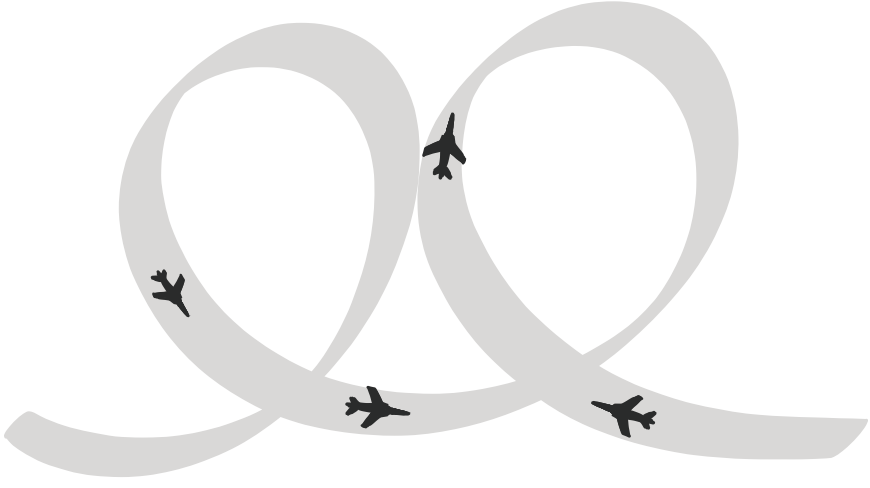

Figure 2.2
Searching for the ingredients to air superiority was difficult.

As Boyd went back to earlier wars, he found that this was hardly unique. In fact, throughout history, having superior weaponry seemed to rarely be a relevant factor in determining the victor. As Boyd continued to research, he repeatedly found instances where numerically and technically superior

forces lost spectacularly to their poorly equipped opponents. That meant that despite his revolutionary work on EM theory, combat success couldn't be determined by any one formula or set of maneuvers.

Boyd studied great military tacticians from Sun Tzu and Alexander the Great to the Mongols, Clausewitz, and Helmut von Moltke. He also interviewed surviving officers of the most successful German Army units during World War II to understand what made them different. He soon realized that battlefield success hinged on which side could make the right decisions more quickly and accurately to reach a given objective. This was true even in cases where the victor possessed inferior weapons, fewer soldiers, poorer training, and battlefield terrain disadvantages. Not only that, but he noticed that this decision-making advantage could be gained just as well by either optimizing your own decision-making abilities or by thwarting those of your opponent.

This realization led Boyd to examine more deeply how the decision-making process works and what can make it more or less effective. In the process, he invented what is now known as the *OODA loop*.

Operation Millennium Challenge

A good recent demonstration of outmaneuvering superior enemy forces happened in the *Operation Millennium Challenge 2002* war game event held by the US Armed Forces in the Persian Gulf. On one side was the "Blue" side with the latest US weapons, while on the other was an unknown "Red" adversary armed with little more than light weaponry.

The Blue side had built elaborate plans to make the most of its superior firepower. What they had not counted on was the savvy leadership of the Red team under Lt. General Paul Van Riper. Van Riper knew there was no way he could win head-to-head against his more capable opponent. Instead, he used his ability to adapt rapidly to outmaneuver and overwhelm the Blue side. He used motorcycle messengers and World War II–style light signal communications to avoid Blue's sophisticated electronic surveillance. He then launched a surprise raid on the Blue fleet by using lightly armed speedboats to locate the fleet followed by a large salvo of missiles. This approach not only eliminated any advantages Blue had, but also used Blue's size and rigidity of command as friction against itself.

Despite Red's seemingly long odds, the Blue side's ability to react was subsequently overwhelmed, resulting in 16 ships being "sunk," including an

aircraft carrier, ten cruisers, and five of the six amphibious ships involved. In all, 20,000 Blue service personnel were "lost" by the surprise maneuver, greatly embarrassing the US military.

The OODA Loop

Boyd hypothesized that all intelligent creatures and organizations undergo decision loops continuously as they interact within their environment. Boyd described this as four interrelated and overlapping processes that are cycled through continuously and which he called the OODA loop, depicted in Figure 2.3. These processes include the following:

- **Observe:** The collection of all practically accessible current information. This can be anything from observed activity, unfolding conditions, known capabilities and weaknesses present, available intelligence data, and whatever else is at hand. While having lots of information can be useful, information quality is often more important. Even understanding what information you do not have can improve the efficacy of decisions.
- **Orient:** The analysis and synthesis of information to form a mental perspective. The best way to think of this is as the context needed for making your decision. Becoming oriented to a competitive situation means bringing to bear not only previous training and experiences, but also the cultural traditions and heritage of whoever is doing the orienting—a complex interaction that each person and organization handles differently. Together with observe, orient forms the foundation for situational awareness.
- **Decide:** Determining a course of action based upon one's mental assessment of how likely the action is to move toward the desired outcome.
- **Act:** Following through on a decision. The results of the action, as well as how well these results adhere to the mental model of what was expected, can be used to adjust key aspects of our orientation toward both the problem and our understanding of the greater world. This is the core of the *learning* process.

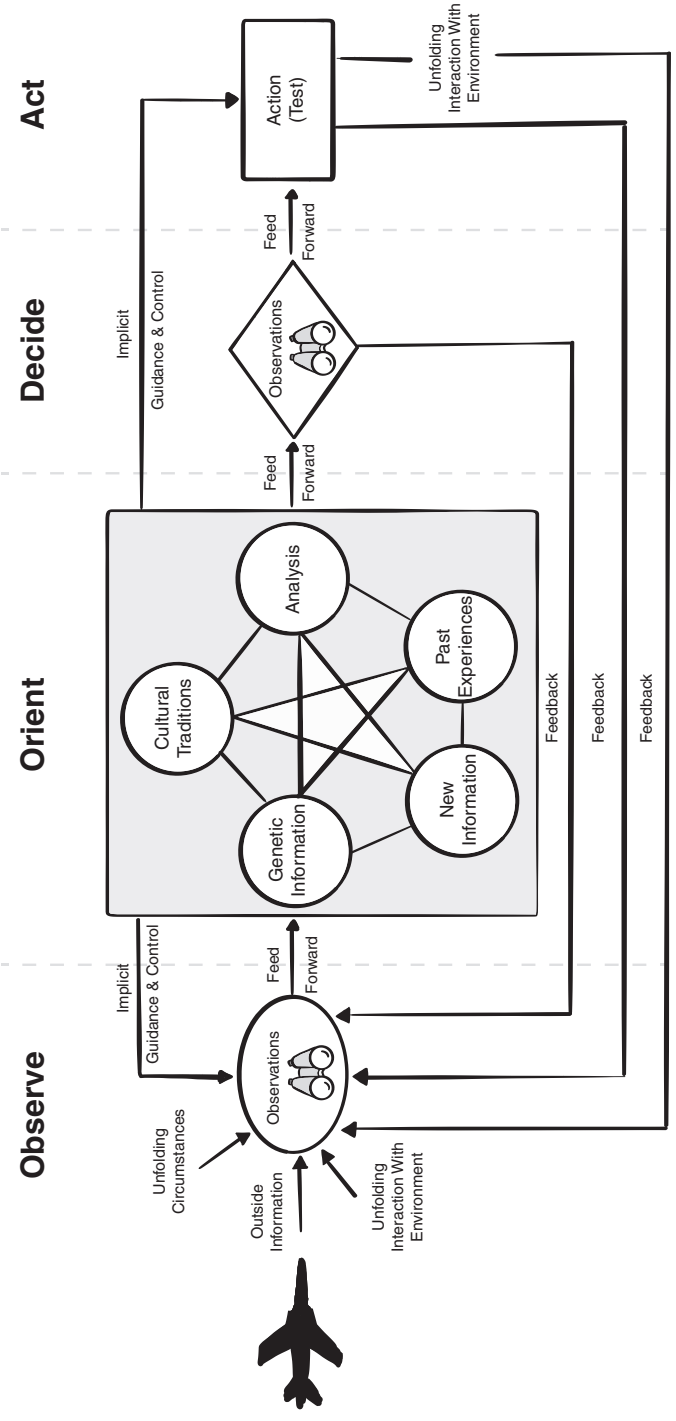

Figure 2.3
OODA loop.

Many who compare OODA to the popular PDCA cycle[1] by W. Edwards Demming miss the fact that OODA makes clear that the decision process is rarely a simple one-dimensional cycle that starts with observation and ends with action. The complex lines and arrows in Boyd's original diagram visualize the hundreds of possible loops through these simple steps in order to get to the desired outcome. As such, the most suitable path is not always the next in the list. Instead, it is the one that ensures there is sufficient alignment to the situation. That means that there will be many cases where steps are skipped, repeated, or even reversed before moving on. In some situations all steps occur simultaneously.

To help illustrate this nonlinear looping, let's take a very simplistic example of a typical process failure.

You get an alert that appears to be a failed production service (Observe->Orient).

You decide to investigate (Decide->Observe).

Before you can act, someone points out (Observe) that the alert came from a node that was taken out of production.

As you change your investigation (Orient) you then may go to see what might have been missed to cause the spurious alert (Observe).

Then you fix it (Decide->Act).

The action will likely involve changing the way people approach pulling nodes from production (Orient).

This solution may need to be checked and tuned over time to ensure it is effective without being too cumbersome (Observe, with further possible Decide->Act->Orient->Observe cycles).

What is important to recognize is that rapidly changing conditions and information discovered along the way can change the decision maker's alignment, leading to a new orientation. This can necessitate canceling or revising a plan of action, seeking out new information, or even throwing out decisions you may have *thought* you needed to make and replacing them with different and more appropriate ones. It may even require knowing when to throw away a once well-used practice in order to incorporate new learning.

1. Known as "Plan-Do-Check-Act," it is a control and continuous improvement cycle pioneered by Demming and Walter Shewart and now used heavily in Lean Manufacturing.

With the loop in hand and understanding how changing conditions can affect the way it is traversed, Boyd became interested in exploring the ways that one could not only out-decide their opponent, but also disrupt their decision process. Was there a way to overwhelm the enemy by changing the dynamics on the battlefield beyond the enemy's ability to decide effectively?

Boyd ultimately called this "getting inside" your opponent's decision cycle. Increasing the rate of change beyond the enemy's ability to adjust effectively can overwhelm their decision making enough to render them vulnerable to a nimbler opponent. He realized that the path to do this started with traversing the OODA decision loop to the target outcome faster than your opponent can.

With this knowledge, Boyd tried to identify the ingredients necessary to drive effective decision making.

The Ingredients of Decision Making

Figure 2.4
Success is having sufficient amounts of the right ingredients.

Just like any recipe, decisions are only as good as the quality of the ingredients on hand and the skill used putting them together. While the importance of any one ingredient can often differ from one decision to the next, they all play an important role in determining the efficacy of the decision-making process.

In order to understand more, let's take a look at each of these ingredients.

Ingredient 1: The Target Outcome

An effective decision is one that helps you in your pursuit of your *target outcome*. Target outcomes are the intended purpose of the decision. The better any target outcomes are understood by those making decisions, the better the person making the decision can choose the option that is most likely to lead to progress toward achieving those outcomes.

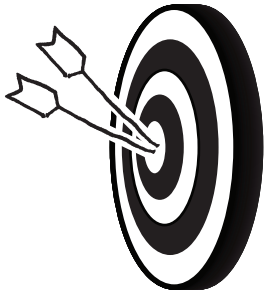

Figure 2.5
The target outcome.

Besides helping to highlight the better option in a decision, awareness of the target outcome also helps to determine decision efficacy by providing a means to measure any progress made toward it. This aids in learning and improvement, making it far easier to investigate cases where progress did not match expectations. This can help answer questions like:

- Was the decision executed in a timely and accurate way? If not, why not?
- Did the decision maker and others involved in executing it have sufficient situational awareness of current conditions to make an optimal match? If not, why not?

- Was the decision the best one to execute, or were there other, more suitable decisions that could have been made that might have had a more positive impact on outcome progress? If better alternatives existed, what can be learned to make it more likely to choose the more optimal decisions in the future?

The third point is the most important of these to note. While there are other ways to find awareness and execution improvement areas, it is far more difficult to determine whether the decision chosen to pursue, as well as the process for choosing it, could itself be improved without an adequately defined target outcome to measure it against. Remaining unaware that the target premise used to decide which decisions you need to make is flawed is incredibly corrosive to the efficacy of your decision-making abilities.

Despite their importance, target outcomes rarely get the attention they deserve. The problem begins with the fact that most decision frameworks spend far more time focusing on how quickly you execute a decision and not whether the decision is the one you need to make. This is especially true in IT, where far more value is often placed on team responsiveness and delivery speed. What information IT teams do get about outcome intent is usually a description of an output, such as a new feature, or performance target, like mean time to recover (MTTR).

There are also challenges around the communication of the outcomes themselves. Some people simply do not think delivery teams need to know anything about the desired outcome. More frequently, however, target outcomes are not clearly communicated because the requestor may not have a clear idea themselves of what they are. This "not knowing" your target outcome may sound strange but is surprisingly common.

To illustrate this, let's go back to our coffee shop example.

When we say we want a cup of coffee, what exactly is our underlying intent? It probably isn't to simply have one in our possession. In fact, most of us would be pretty upset if all we had were the disappointingly tepid remains of one.

Figure 2.6
Tepid coffee rarely sparks joy.

Typically, what we are looking for are the conditions that we associate with obtaining or consuming a cup of coffee. We might like the taste, the warmth, or the fact that it helps us feel more alert. It is possible that we don't actually care about the coffee but are more interested in the opportunity to relax or chat with friends or colleagues. Getting a coffee may be an excuse to find a convenient place to finalize a business opportunity. It might even be as simple as creating an opportunity to get Internet access, as it was with our friend earlier.

What sometimes makes achieving target outcomes tricky is that changing conditions can alter its intent or invalidate it altogether. For instance, stepping out of the office into a hot day can cause someone to change their order to an iced coffee. Suddenly finding yourself late for a meeting can throw out the idea of relaxing at the coffee shop, possibly eliminating the trip entirely. If we are with our Internet-obsessed friend, we might go to a different coffee shop where the coffee is less enjoyable and choose a smaller size or a different blend than we would normally have.

As a result, not only is all the hard work put in to enact the wrong decision a waste, but any measures used to gauge progress can at best be meaningless and at worst perpetuate the problem. Even if by chance you reach your desired outcome, the initial flaw means that your ability to learn from

your actions to improve and replicate the success has been undermined, opening you up for surprise and disappointment when it eventually fails.

The military faces very similar challenges. It is easy for a commander to order his troops to march up a hill or send a squadron to bomb a target. But if the commander's intent is not known, conditions might change and destroy any advantage the commander thought the action might have attained at best, or at worst lead to the destruction of the unit.

For this reason, Boyd and others found that it was better to turn the problem on its head, focusing instead on communicating to troops the target outcome rather than the actions they should use to try to achieve it. He became a strong proponent of Mission Command as a means for commanders and subordinates to communicate and build a true understanding of the target outcomes desired from a given mission. Mission Command, as discussed later in Chapter 3, "Mission Command," is an approach to communicate the intent and target outcomes desired that still gives those executing to achieve them the ability to adjust their actions as circumstances require them to do so.

This Mission Command method is useful in the service delivery world of DevOps. Not only does it enable teams to adjust to changing delivery conditions more quickly than more top-down approaches, it also enables teams to better discover and correct any flaws found in their understanding of the target outcomes themselves. Anyone who has worked in a delivery team knows that such flaws happen all the time, mostly due to the fact that few ever get to meet, let alone discuss, target outcomes with the actual customer. Most delivery teams instead have to rely upon proxies from the business or make intelligent guesses based upon their own experience and the data available around them.

It is in these proxies and guesses where things can go very wrong. The worst of these is when the focus is not on the actual outcome itself but on the solution itself, the method used to deliver the solution, or measures that have little connection to the target outcome.

To understand better, let's go through some of the common patterns of dysfunction to see why they occur and how they lead us astray.

Solution Bias

How many times have you been absolutely certain you knew exactly what was needed in order to solve a problem only to find out afterward that the solution was wrong? You are far from alone. It is human nature to approach a problem with a predetermined solution before truly understanding the problem itself. Sometimes it is the allure of a solution that misguides us, while at other times we fail to take the necessary time and effort to fully understand the expected outcome.

We commonly encounter solution bias when vendors push their products and services. Rather than spending any effort figuring out the problems you have and the outcomes you desire, vendors try to create artificial needs that are satisfied by their solution. Some do it by flashing fancy feature "bling." Others list off recognized established companies or competitors who are their customers in the hopes that this will entice you to join the bandwagon. Some look to become your answer for your "strategy" in a particular area, be it "cloud," "mobile," "Agile," "offshoring," "DevOps," or some other industry flavor of the month.

Solution providers are notorious for assuming their offering is the answer for everything. They are hardly the only one who suffers from solution bias, however. Even without the influence of slick marketing literature and sales techniques, customers are just as likely to fall into this trap. Whatever the cause, having a solution bias puts those responsible for delivery in an unenviable spot.

The School Bell

Figure 2.7
School bell.

Architect Alastair Parvin provided an illustrative example of this very problem in a story he gave at a TED talk in 2013.[2] A school had approached his architecture firm to redesign the school building. It was Victorian era with narrow hallways so cramped that it was difficult for students to get through between classes. The school administration had accepted that the construction was going to be expensive, but needed the problem to be solved.

At first glance the problem looked straightforward. The school wanted a new school building with spacious halls to accommodate the students moving between classes. Most of us would take that as a signal to start roughing out a design and a project plan for the school administration to look over.

The architects in Parvin's firm took a different approach. They started by looking more closely at the problem (crowded hallways) and the desired but only indirectly requested outcome (allowing students to move easily between classes). Anyone who has worked with flow problems knows that short, intense bursts can be a real mess. It is better to find ways to have lower and more even flows. This gave the architects an idea.

They came back to the school and recommended that instead of a new building they should redesign the bell system so that smaller school bells went off in different places at different times. This would allow for student traffic to be distributed more evenly, effectively eliminating congestion at a much lower cost. The solution worked brilliantly. By moving away from a predetermined solution the school managed to save millions.

Execution Bias

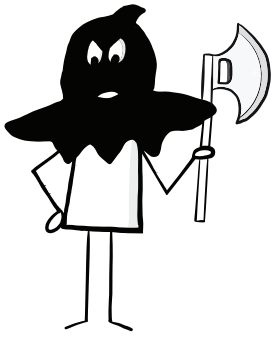

Figure 2.8
Bill let it be known he was completely focused on the execution.

2. "Architecture for the People by the People," Alistair Parvin: Ted Talk, https://www.ted.com/talks/alastair_parvin_architecture_for_the_people_by_the_people/transcript?language=en

Requesting and delivering inappropriate solutions is not the only way we stray from the target outcome. We all carry any number of personal biases about *how* we, or others on our behalf, execute. There is nothing in itself wrong with having a favorite technology, programming language, process, or management approach. Our previous personal experience, the advice of people we trust, and the realities of the ecosystem we are working in are inevitably going to create preferences.

The problem occurs when the method of execution eclipses the actual target outcome. As Boyd discovered, knowing all the possible maneuvers is of little benefit if there is no target to aim for.

In technology, execution bias can take the form of a favored process framework, technology, vendor, or delivery approach regardless of their appropriateness. When execution bias occurs, there is so much focus on how a particular process, technology, or approach is executed that any target outcomes get overlooked. This is why there can be so many excellent by-the-book Agile, Prince2, and ITIL implementations out there that, despite their process excellence, still fail to deliver what the customer really needs.

Delivering Measures over Outcomes

Good metrics measuring the effectiveness in moving toward achieving a target outcome can be very useful. They can help spot problem areas that require addressing, guide course correcting, and ultimately help teams learn and improve. However, it is far too easy to place more focus on achieving a metric target than the outcomes it is supposed to be aiding. There are two rather common causes for this.

The first is when the target outcomes are poorly understood or, as in our coffee example, more qualitative in nature. It may take a number of attempts with the customer to provide enough clues to really understand what they are looking for. This can seem arduous, especially when multiple teams who have little to no direct interaction with the customer are delivering components that together provide the solution. In many cases teams might find it too hard or time consuming to even try. Instead, they opt to track more easy-to-measure metrics that either follow some standard set of industry best practices or at least might *seem* like a reasonable proxy to measure customer value.

The other cause is the more general and long-held practice by managers to create measures to evaluate the individual performance of teams and team members. In Frederick Wilson Taylor's book *The Principles of Scientific Management*, management's role was to tell their workers what to do and how to do it, using the measures of output and method compliance as the means to gauge, and even incentivize, productivity.

Both cases encourage the use of measures like number of outputs per period of time, found defects, mean time to recover, delivery within budget, and the like. All are easy to measure and have the added benefit of being localized to a given person or team. However, unless a customer is paying only for outputs, they rarely have anything but the most tenuous link to the actual desired outcome.

Encouraging more outputs may not sound like such a bad thing on its face. Who wouldn't want more work done in a given time, or higher uptime? The former nicely measures delivery friction reductions, while the latter provides a sense that risk is being well managed. However, the problem is two-fold. The first and most obvious is that, as we saw in our coffee story, few outcomes are met simply by possessing an output. This is especially true if an output is only a localized part of the delivery. Likewise, few customers would be happy to have a great database on the backend or a high uptime on a service that doesn't help them achieve what they need.

Another challenge comes from human behavior when they know they are being assessed on how well they meet a target. Most will often ruthlessly chase after the target, even if doing so means stripping out everything but the absolute minimum required to achieve it. This is extremely problematic, especially if there is only at best a loose connection between the measure and the desired outcome. I have personally observed teams that have sacrificed service quality and maintainability by reclassifying defects as feature requests to meet defect targets. The defects were still there, and often even worse than before, but as the count of what was being measured was dropping, the team was rewarded for their deceit.

The Objectives and Key Results (OKR) framework tries to overcome this by using a common set of clearly defined qualitative objectives that are intended to provide targeted measures. However, many find doing this well to be difficult. The best objectives are directly tied to the pursuit of the outcomes the customer wants, with key result measures that are both tied to those objectives and have sufficiently difficult targets (to encourage

innovation) that they are reachable only some of the time. In this way the targets form more of a direction to go in and a means to measure progress toward rather than some definitive targets that staff and teams feel they are being evaluated against. Unfortunately, this thinking goes against many of the bad assessment habits many of us grew up with in school. Instead, organizations tend to turn objectives into the same traditional output performance criteria with outputs becoming the key results to be quantitatively measured, thereby losing the value and original intent of OKRs.

Lost Bills

The dysfunction caused by output-based metrics is bad enough when everyone involved is in the same organization. It is even worse when such metrics form the basis of compensation between two different firms.

One large company was looking to streamline the operational support of various backend systems used to support the business. The old operational approach had led to many small teams that felt very protective of their space. Rather than fight with them, the CIO decided to wipe the problem away by outsourcing the entire mess.

The systems, much like the business, were stable and mature. The CIO decided that the most effective way to structure the outsourcing contract would be to pay for the number of incident tickets handled.

One of the outsourcing targets was the billing system. Even though billing was one of the most critical parts of the company, and one of the most interconnected, it was generally very stable. The billing process ran every night from 1 to 3 a.m. As it required locking customer records in both the CRM and billing databases, it was important that the process complete well before the Sales and Customer Support teams needed to access the system during the business day.

After a few months, the billing runs started to fail. The outsourced support investigated and noticed a read error of data in the cache containing customer records being processed. They cleared the cache, restarted the billing process, and everything continued seamlessly. The error began to happen more and more frequently. One of the support people decided to automate the recovery with a script. This eliminated the manual work, sped up recovery, and as it opened and closed incident tickets associated with the error, the outsourcer would reap income from the work. Everyone seemed to win.

Another few months went by. Customer Support started noticing that the number of complaints about incorrect bills had been growing significantly. The problems

were very strange, with customers being billed for services they had never had. Eventually, it got the attention of the executive staff and a team of engineers was sent in to investigate. What they found was startling.

A CRM system upgrade had turned on the ability for Sales and Customer Support staff to insert double byte and other special characters in the customer database. The billing batch processes could not handle these and would throw an exception when encountered, halting the billing process. Such failures are never good and indicate that there are likely awareness and quality issues in the delivery process.

But this particular problem was far worse than that.

The script that the outsourced support had written "solved" the offending error by dropping the customer identification database records that had been loaded before the special character was hit. The now orphaned billable assets would then be appended to subsequent customers as the process was restarted. This caused three types of errors:

Some customers were billed for products and services they never used.

Some customers were not billed for products and services.Some proportion of these customers also disappeared entirely from the customer and billing databases.

As the outsourced operational support staff were paid by the number of tickets they handled, they had little incentive to find the root cause of the problem, let alone fix it. Instead, they were measured and rewarded for efforts that ultimately ruined the integrity of the customer and billing databases and the overall billing process.

Ingredient 2: Friction Elimination

Figure 2.9
Friction elimination needs to be strategic, not random.

As mentioned in Chapter 1, friction is anything that increases the time and effort needed to reach a target outcome. It can arise at any point along the journey. This includes the decision cycle itself. We can be affected by it from the time and effort to gather and understand the information surrounding a situation through one or more of the various steps needed to make the decision, act upon it, and review its result. Friction can be so potent as to prevent you from reaching an outcome entirely.

Eliminating delivery friction, whether in the form of provisioning infrastructure, building and deploying code, or consuming new capabilities quickly, is what attracts people to DevOps and on-demand service delivery. There is also a lot of value in trying to mitigate many of the sources of wasteful friction whenever possible. But as Boyd found, and what so many of us miss, is that eliminating friction only provides value if it actually results in helping you reach your target outcome.

Many confuse friction elimination with increasing response and delivery speed rather than reaching the outcome. Teams dream of instantly spinning up more capacity and delivering tens or even hundreds of releases a day. Organizations will tout having hundreds of features available at a push of a button. We do all this believing, with little strong supporting evidence, that being fast and producing more will somehow get us to the outcome. It is like buying a fast race car with lots of fancy features and expecting that merely having it, regardless of the type of track or knowing how to drive effectively on it, is enough to win every race.

The gap between the lure of possessing a "potential ability" and effective outcome delivery is often most pronounced when there is friction in the decision cycle itself. "Red" side's win over "Blue" in *Operation Millennium Challenge 2002* is a great demonstration of this. The "Blue" side's superior weaponry was not sufficient to overcome its inferior decision loop agility against the "Red" side.

This same decision cycle friction was likely at work in Boyd's analyses of Korean War era fighter aircraft. The superior abilities of enemy MiG fighter jets over the F-86 that Boyd found was often made irrelevant in the battlefield due to the relatively higher friction in communication flow and command structures of Communist forces compared to those of the US. This friction made it far more difficult for Communist forces to quickly understand and adjust to changing battlefield dynamics, as well as to catch

and learn from mistakes. American pilots did not have such problems and exploited these differences to their own advantage.

While the IT service delivery ecosystem does not face quite the same adversarial challenges, these same friction factors do have a significant impact on an organization's success. Teams can have an abundance of capabilities to build and deploy code quickly, yet still have so much decision-making friction that they are prevented from delivering effectively. Such friction can arise from such sources as defects and poorly understood target outcomes to poor technical operations skills. Interestingly, symptoms of decision friction do not necessarily manifest as problems of delivery agility. Often they take the form of misconfigurations, fragile and irreproducible "snowflake" configurations,[3] costly rework, or ineffective troubleshooting and problem resolution. All of these not only impact the service quality that the customer experiences but also can consume more time and resources than former traditional delivery methods did.

Even when delivery agility is a problem, it is not always caused by poor delivery capabilities. I regularly encounter technical delivery teams with nearly flawless Agile practices, continuous integration (CI)/continuous delivery (CD) tooling, and low friction release processes that stumble because the process for deciding what to deliver is excessively slow. At one company, it typically took *17 months* for a six-week technical delivery project to traverse the approval and business prioritization process in order to get into the team's work queue.

Another frequent cause of delivery friction occurs when work that contains one or more key dependencies is split across different delivery teams. The best case in such a situation is that one team ends up waiting days or weeks for others to finish their piece before they can start. However, if the dependencies are deep or even cycle back and forth between teams, such poor planning can drag out for months. At one such company this problem was not only a regular occurrence but in places was *seven* dependencies deep. This meant that even when teams had two-week-long sprints, it would take a *minimum* of 14 weeks for affected functionality to make it through the delivery process. If a bug in an upstream library or module was encountered or if the critical functionality was deprioritized by an upstream team, this delay could (and often did) stretch for additional weeks or months.

3. https://martinfowler.com/bliki/SnowflakeServer.html

Delivery friction can also come from behaviors that are often perceived to improve performance. Long work weeks and aggressive velocity targets can burn out teams, increasing defect rates and reducing team efficiency. Multitasking is often a method used to keep team members fully utilized. But having large amounts of work in progress (WIP), along with the resulting unevenness and predictably unpredictable delivery of work, as well as the added cost of constant context switching, usually results in slower and lower-quality service delivery.

Friction in feedback and information flow can also reduce decision, and thereby delivery, effectiveness. I have encountered many companies that have fancy service instrumentation, data analytics tools, and elaborate reports and yet continually miss opportunities to exploit the information effectively because it takes them far too long to collect, process, and understand the data. One company was trying to use geolocation data in order to give customers location-specific offers like a coupon for 20 percent off a grocery item or an entrée at a restaurant. However, they soon found that it took three days to gather and process the information required, thus making it impossible to execute their plan for just-in-time offers.

These are just a small sampling of all the types of delivery friction. Others can be found in Chapter 4, "Friction," which goes into considerably more depth. Understanding the various friction patterns that exist and their root causes can help you start your own journey to uncover and eliminate the sources of friction in your delivery ecosystem. The next decision-making ingredient, *situational awareness*, not only can aid in this search, but is the key ingredient for building enough context and understanding of your delivery ecosystem to make effective decisions.

Ingredient 3: Situational Awareness

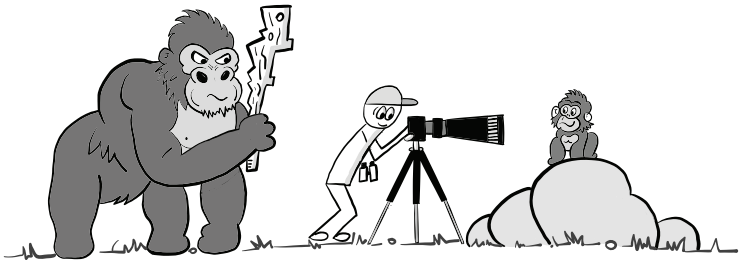

Figure 2.10
Never overestimate your level of situational awareness.

Situational awareness is your ability to understand what is going on in your current ecosystem, and combine it with your existing knowledge and skills to make the most appropriate decisions to progress toward your desired outcome. For Boyd, all this gathering and combining takes place in the OODA loop's *Orient* step.

It makes sense that having information about your delivery ecosystem provides a level of insight into its dynamics to improve the speed and accuracy of the decisions you make in it. It is gaining this edge that has been the draw for everyone from investment banks and marketing companies to IT services organizations to invest in large-scale Big Data and business intelligence initiatives as well as advanced IT monitoring and service instrumentation tools.

But while the desire to collect as much information as possible seems logical, collecting and processing information that is unsuitable or has no clear purpose can actually harm your decision-making abilities. It can distract, misdirect, and slow down your decision making.

What makes information suitable depends heavily upon the right mix of the following factors:

- **Timeliness:** Relevant information needs to be both current and available in a consumable form at the time the decision is being made to have any positive impact. Information that arrives too slowly due to ecosystem friction, is now out of date, or is buried in extraneous data will reduce decision-making effectiveness. Collecting just enough of the right data is just as important.
- **Accuracy:** Accurate information means fewer misdirected or inaccurate decisions that need to be corrected to get back on the path to the target outcome.
- **Context:** Context is the framing of information in relation to the known dynamics of the ecosystem you are executing in. It is how we make sense of our current situation and is an important prerequisite for gaining situational awareness. We use context to not only understand the relevance of information to a given situation, but also to gauge the level of risk various options have based upon our grasp of the orderliness and predictability of the ecosystem. How all of this works, and how it often goes wrong, is covered in considerable depth in Chapter 5, "Risk."

- **Knowledge:** The more you know about the relative appropriateness of the options and capabilities available to you, the greater the chance you will choose the option that will provide the most positive impact toward achieving the target outcome. How much knowledge you have and how effectively it can aid your decision making depends heavily upon the decision-making ingredient of learning effectiveness.

Assuming you are able to obtain suitable information, you then need to combine it effectively to gain sufficient situational awareness. There are many ways this process can fall apart, making it the most fragile component of the decision process. Boyd was intrigued by this, and spent the rest of his life trying to understand what conditions could damage or improve it.

Understanding these conditions is both important and discussed more fully in Chapter 6, "Situational Awareness." For the purposes of this chapter, it is important to point out the two most common places where situational awareness can silently deteriorate, often with chronic if not catastrophic consequences. These are the challenge with trust, and the fragility of mental models and cognitive biases.

The Challenge of Trust

Most of us cannot acquire all the information directly ourselves and instead rely upon other people, either directly or from tools they build and run. If any of those people do not trust those who consume the information or what they will do with it, they may withhold, hide, or distort it. This can be done either directly, if they are handling the data themselves, or by suboptimizing the data collection and processing mechanisms they are asked to build and manage.

Lack-of-trust issues are surprisingly common. The vast majority of the time they arise not out of malice but out of fear of blame, micromanagement, loss of control, or simply fear of the unknown. They are also just as likely to be caused by problems between peers as well as between managers and individual contributors in both directions. Many of these issues can develop out of anything from simple misunderstandings caused by insufficient contact or communication to larger organizational culture issues. Whatever the cause, the resulting damage to decision making is very real and can be far reaching.

The Fragility of Mental Models and Cognitive Biases

Human brains have evolved to rely upon two particular shortcuts to both speed up and reduce the amount of mental effort required to make decisions. These shortcuts, the mental model and cognitive bias, are generally quite useful when we are in stable, well-known environments. However, as you will see, these mechanisms are particularly susceptible to flaws that can severely degrade our decision-making abilities.

Mental models are sets of patterns that our brains build that are internal representations of an ecosystem and the relationships between its various parts to anticipate events and predict the probable behaviors of elements within it. Mental models allow us to quickly determine what actions are likely the most optimal to take, dramatically reducing the amount of information that needs to be gathered and analyzed.

Mental models are valuable, which is why people with experience are often quicker and more effective than a novice in situations that are similar to what they have been through previously. However, this mental model mechanism has several serious flaws. For one, a mental model is only as good as our understanding of the information we use to build and tune it. This information may be partial, misconstrued, or even factually incorrect. Boyd knew that this would inevitably lead to faulty assumptions that damage decision making.

The most obvious threat to mental model accuracy is having a rival try to intentionally seed inaccurate information and mistrust of factual information and knowledge. Studies have shown that disinformation, even when knowingly false or quickly debunked, can have a lingering effect that damages decision making.[4-5] In recent years such behavior has moved from the battlefield and propagandist's toolbox to the political sphere to discredit

4. "Debunking: A Meta-Analysis of the Psychological Efficacy of Messages Countering Misinformation"; Chan, Man-pui Sally; Jones, Christopher R.; Jamieson, Kathleen Hall; Albarracín, Dolores. *Psychological Science*, September 2017. DOI: 10.1177/0956797617714579.
5. "Displacing Misinformation about Events: An Experimental Test of Causal Corrections"; Nyhan, Brendan; Reifler, Jason. *Journal of Experimental Political Science*, 2015. DOI: 10.1017/XPS.2014.22.

opponents through overly simplistic and factually incorrect descriptions of policies and positions.

But incorrect information, let alone active disinformation campaigns, is far from the most common problem mental models face. More often they become flawed through limited or significantly laggy information flow. Without sufficient feedback, we are unable to fill in important details or spot and correct errors in our assumptions. When an ecosystem is dynamic or complex, the number and size of erroneous assumptions can become so significant that the affected party becomes crippled.

Like mental models, cognitive biases are a form of mental shortcut. A cognitive bias trades precision for speed and a reduction of the cognitive load needed to make decisions. This "good enough" approach takes many forms. The following are just a handful of examples:

- ***Representativeness biases*** are used when making judgments about the probability of an event or subject of uncertainty by judging its similarity to a prototype in their mind. For instance, if you regularly receive a lot of frivolous monitoring alerts, you are likely to assume that the next set of monitoring alerts you get are also going to be frivolous and do not need to be investigated, despite the fact that they may be important. Similarly, a tall person is likely to be assumed to be good at playing basketball despite only sharing the characteristic of height with people who are good at the sport.
- ***Anchoring biases*** occur when someone relies upon the first piece of information learned when making a choice regardless of its relevancy. I have seen developers spend hours troubleshooting a problem based on the first alert or error message they see, only to realize much later that it was an extraneous effect of the underlying root cause. This led them on a wild goose chase that slowed down their ability to resolve the issue.
- ***Availability biases*** are the estimation of the probability of an event occurring based on how easily the person can imagine an event occurring. For instance, if they recently heard about a shark attack somewhere on the other side of the world, they will overestimate the chances they will be attacked by a shark at the beach.

- ***Satisficing*** is choosing the first option that satisfies the necessary decision criteria regardless of whether there are other better options available. An example is choosing a relational database to store unstructured data.

When they are relied upon heavily, it is extremely easy for cognitive biases to result in lots of bad decisions. In their worst form they can damage your overall decision-making ability.

Unfortunately, we in IT regularly fall under the sway of any number of biases, from hindsight bias (seeing past events as predictable and the correct actions obvious even when they were not at the time) and automation bias (assuming that automated aids are more likely to be correct even when contradicting information exists) to confirmation bias, the IKEA effect (placing a disproportionately high value on your own creation despite its actual level of quality), and loss aversion (reluctance to give up on an investment even when not doing so is more costly). But perhaps the most common bias is *optimism bias*.

IT organizations have grown used to expecting that software and services will work and be used as we imagine they will be, regardless of any events or changes in the delivery ecosystem. A great example of this is the way that people view defects. Defects only exist in the minds of most engineers if they are demonstrated to be both real *and* having a direct negative impact, while risks are those situations that engineers view as likely events. This extreme optimism is so rampant that the entire discipline of Chaos Engineering has been developed to try to counteract it by actively seeding service environments with regular failure situations to convince engineers to design and deliver solutions that are resilient to them.

Another serious problem with mental models and cognitive biases is their stickiness in our minds. When they are fresh they are relatively malleable, though if you were ever taught an incorrect fact you likely know that it will still take repeated reinforcement to replace it with a more accurate one learned later. The longer that a mental model or bias remains unchallenged in our minds, the more embedded it becomes and the harder it is to change.

Ingredient 4: Learning

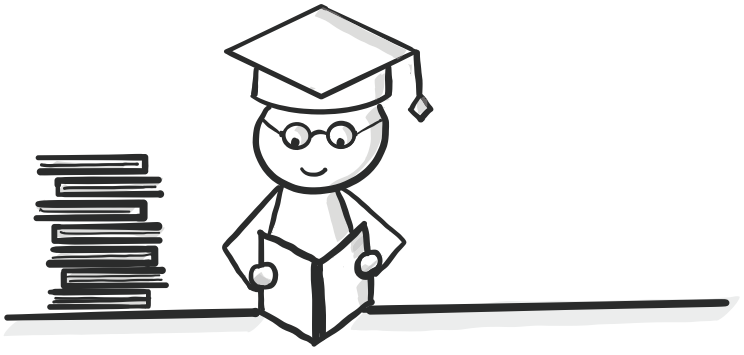

Figure 2.11
Learning isn't just school and certifications.

The final and frequently misunderstood piece of the decision-making process is *learning*. Learning is more than the sum of the facts you know or the degrees and certifications you hold. It is how well you can usefully capture and integrate key knowledge, skills, and feedback into improving your decision-making abilities.

Learning serves several purposes in the decision process. It is the way we acquire the knowledge and skills we need to improve our situational awareness. It is also the process we use to analyze the efficacy of our decisions and actions toward reaching our target outcomes so that we can improve them.

Learning is all about fully grasping the answer to the question *why*. This could be why a decision did or did not work as expected, where and why our understanding of the target outcome was or was not accurate, or how and why we had the level of situational awareness we did. It is the core of various frameworks, including Lean, and it is how we improve every part of the decision-making process.

The best part about learning is that it doesn't need to occur in a formal or structured way. In fact, our most useful learning happens spontaneously as we go through life. This should help us constantly improve the quality of our decisions—that is, if we remain willing and able to learn.

Failing to Learn

Learning is a lifelong pursuit, one we like to think we get far better at it with experience. However, it is actually really easy for us to get *worse* at it

over time. We even joke that "you can't teach an old dog new tricks." What causes us to get worse at doing something so fundamental?

A common belief is that any problems we may have are merely a result of having less dedicated time for learning. It is true that learning requires time to try things out, observe what happens, and reflect to really understand the reasons we got the results we did so that we can apply that knowledge to future decision making. However, the problem is far deeper.

For learning to occur we also need to be amenable to accepting new knowledge. That is pretty easy to do if you have no prior thoughts or expectations on a given topic. But as we gain experience, we start to collect various interpretations and expectations on a whole raft of things that we can use to build and tune our mental models. When they accurately reflect our situation, we can speed up and improve the accuracy of our decision making. But, unfortunately, not everything we pick up is accurate. Sometimes we learn the wrong lesson, or conditions change in important ways that invalidate our previous experiences.

As we discussed earlier, these inaccuracies can create serious flaws in our situational awareness. They also can damage our learning ability. When confronted with new knowledge that conflicts with our current view, we will sometimes simply ignore or overlook the new information because it does not meet our preset expectations. Other times we might actively contest the new evidence unless it can irrefutably disprove some interpretation we hold, even if the view we hold cannot itself stand up to such a test. This high bar creates friction that can slow or prevent us from learning.

The causes of this resistance have roots that go back to our formal schooling days.

"But That's Just a PC"

There are a lot of scenarios where long-held habits and beliefs can get in the way of our ability to learn. Sometimes the outcome we are targeting becomes so ingrained that we fail to learn when our customers' needs change. Other times we get so used to the idiosyncrasies of our environment that we fail to see important problems or risks in the state of our code and repository health, the suitability of our long-held processes, or challenges in skill and relationship dynamics in our teams. These can all

make life far more difficult than it needs to be and even put the business in jeopardy.

An energy company fell into such a predicament. Only after struggling to overcome their established beliefs did they discover that these very beliefs were endangering the core of its business.

The company had long relied upon a critical internally built application written in Fortran. It was designed to calculate how much electricity each power plant should generate across each service area, hour by hour each day over the entire 24-hour period. This information was critical for managing the health of the generation plants and the power grid, as well as the quality and availability of the electricity supply. Even more importantly, the generated results also had to be provided to government regulators. While generation and distribution had some buffer to allow things to coast along for a couple of hours with no data, failing the regulator's requirement meant millions in immediate fines.

Even though this application and the data it provided were so important to the business, few company employees were aware of its existence. For decades the application had run with little fanfare on a large mainframe in the basement of a building at one of the power stations. It only came to light when the building it was in was slated to be demolished.

In the process of trying to figure out what to do, management discovered that all of the original application developers had long since retired. The mainframe itself, while impressively large, was so ancient that the manufacturer had long gone out of business.

The situation seemed dire, but the business didn't seem to fully recognize the risk it had. The setup had been running so seamlessly for the most part that, at first glance, all that was required was to move the equipment. It didn't help that there was a strong belief that mainframes are by their nature failure-proof. That this one had been running since the 1970s only seemed to solidify that thinking.

The team I was leading at the time was tasked with coming up with a workable solution to the problem. What confidence others had quickly faded when we began to understand the complexities of the situation. We quickly swarmed the problem, with one part of the team looking into the problem of moving the existing system, while the rest of us looked for other alternatives.

Along the way we managed to dig up the source code and, while it was ancient enough to require some effort to port to a more contemporary

platform, doing so would be fairly straightforward. We were also fortunate enough that some of us had long ago become well versed in the intricacies of Fortran.

After a few weeks we managed to port the entire application to Linux. Eager to see if it would work, we ran it alongside the existing system. The hourly process took on average 50 minutes or so end-to-end, so we wanted to make sure the ported application would be not only just as accurate but could also fit in the same execution window.

That was when the real fun began.

At its first execution, the new setup executed the entire process in less than one second. After the main system completed its work we compared and found identical results. In disbelief, we continued running both in parallel for several weeks, each time coming up with the same results. Decades of technology innovation meant that what had once required a huge mainframe and nearly an hour of calculation time could now run on a tiny virtual machine instance in almost no time.

Convincing the business that the new setup was a far superior approach that was both cheaper and had the side effect of removing a massive business risk was extremely difficult. People had long grown to believe that mainframes are far superior in every way to a standard PC. Even though there was ample proof that doing so was safe and expedient, moving such an important application to an ordinary computer, let alone a virtual machine, seemed in many people's minds to be far more dangerous than the existing setup. They were simply too set in their ways to easily accept this new knowledge.

After yet more testing and a lot of pressure to decommission the mainframe, the management team finally gave in and migrated to the ported application.

The Shortcomings of the Formal Education System

Overcoming preexisting beliefs about a solution is not the only problem that can negatively affect our ability to learn. Preconceptions about learning and the learning process can also play a major factor.

Most of us spend a large chunk of our childhood being formally educated in school. While this is useful for kickstarting our life of learning, many of the techniques used can lead to unhealthy habits and expectations

that can impede our ability to learn later in life. The worst of these is creating the expectation that there is only one right way of working.

Having only one right method is easy to teach and test for in a standardized way. The problem is that life is not so black and white. Oftentimes, especially in IT, either there is no single right answer or the right answer no longer is right later down the line. However, the expectation of singular permanence makes people resistant to learning new and potentially better ways of working.

This problem is compounded by the fact that standardized teaching is done in a top-down way. It creates an expectation that there is an all-knowing authority that is there to judge our ability to follow instructions. Such an approach is not only unrealistic and noncollaborative, it also discourages critical thinking and innovation.

Hindsight Bias and Motivated Forgetting

Failing to learn is never good. But the problem is far more than simply missing out on knowing some information. It also can make it a great deal harder to improve our overall decision-making abilities. Ironically, the places where learning is often the most dysfunctional are those officially designed to encourage it. This includes everything from incident postmortems to sprint and project retrospectives, end project reports, and lessons learned reviews.

The reason for this goes back to some of the dysfunctions covered earlier. Being subconsciously predisposed to believe that the world operates in a particular way makes it incredibly easy to believe that events are more predictable than they are. This *hindsight bias* causes us to overlook the facts as they were known at the time and misremember events as they happened. This creates false narratives that can make any missteps appear to be due to a lack of skill or a total disregard for what seem to be obvious facts.

When this is coupled with a lack of trust and fear of failure, people can be motivated to avoid blame by actively forgetting or obfuscating useful details. This creates a culture that is not conducive to learning.

The Pathway to Improved Decision Making

We have discussed how the quality of decisions is dependent upon the strength of our understanding of the problems and desired outcome, the friction in our decision process, our level of situational awareness, and our ability to learn. We have also covered many of the ways that each of these key elements can degrade. How can we begin to improve our own decision-making abilities?

As you probably could have guessed by now, the journey is not as simple as following a set of predefined practices. We have to first dig deeper into how we go about making decisions and answer the following questions:

- Do you and your teammates understand your customer's target outcomes? Do you know why they are important. Do you have a good idea for how to measure that you are on track to reaching them?
- What factors in your ecosystem are causing rework, misalignments, and other issues that add friction that slows down and makes your decision-making and delivery processes inefficient?
- How does the information you and your teammates need to gain situational awareness flow across your ecosystem? Where is it at risk of degrading, and how do you know when it is happening and its potential impact on your decision making?
- How effectively does your organization learn and improve? Where are the weak points, and what effect are they having on your decision-making effectiveness?

Effective DevOps implementations not only can answer these questions with ease, but can tell you what is being done to fix any challenge the team faces. Each implementation journey is different, and there is no one answer for what is right.

This book takes the same approach.

If you think you have a good grasp of the service delivery problem space and are ready to dive straight into practical application, you can jump ahead to Section 3. In those chapters I have written a number of techniques and

approaches that I have used in numerous organizations large and small and found extremely helpful for everyone across the organization.

However, if you are like me, you may feel that you need to understand a bit more about the thinking behind this approach to feel more comfortable with it before making any serious commitment. This deep dive begins with Chapter 3, "Mission Command," in the next section.

Summary

Even though we make decisions every day, becoming an effective decision-maker is a skill that takes a lot of time and effort to master. John Boyd's OODA loop shows that the process is iterative and continual, with success dependent upon the following:

- Understanding the target outcome desired
- Gaining situational awareness through observing and understanding your capabilities and the dynamics of the ecosystem you are operating in
- Identifying the impacts of friction areas to gauge what decision options will most likely help you progress toward the outcome desired
- Following through with the action, and observing its impact so that you can determine how far you progressed, and if there are new details or changes that must be learned and incorporated into your situational awareness

Boyd showed that continually honing situational awareness and learning are at least as important as the speed of decision-making itself. Damage to any ingredient harms decision quality, and with it the ability to succeed.

Index

C

D

E

F

G

L

M

N

O

P

Q

R

S

T

U

V

W