The Kitty Hawk Venture

A Novel About Continuous Testing in DevOps to Support Continuous Delivery and Business Success

Jeffrey Scheaffer
Aruna Ravichandran
Alex Martins

ca technologies
CA Press

Apress®

THE KITTY HAWK VENTURE

A NOVEL ABOUT CONTINUOUS TESTING IN DEVOPS TO SUPPORT CONTINUOUS DELIVERY AND BUSINESS SUCCESS

Jeffrey Scheaffer
Aruna Ravichandran
Alex Martins

The Kitty Hawk Venture: A Novel About Continuous Testing in DevOps to Support Continuous Delivery and Business Success

Jeffrey Scheaffer
San Francisco, California, USA

Aruna Ravichandran
San Jose, California, USA

Alex Martins
San Francisco, California, USA

ISBN-13 (pbk): 978-1-4842-3660-4
ISBN-13 (electronic): 978-1-4842-3661-1
https://doi.org/10.1007/978-1-4842-3661-1

Library of Congress Control Number: 2018947390

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Susan McDermott
Development Editor: Laura Berendson
Coordinating Editor: Rita Fernando

Cover designed by eStudioCalamar

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, email orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please email rights@apress.com or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484236604. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Advance praise for *The Kitty Hawk Venture*

"In today's digitally driven world, consumers expect access to technological solutions no matter where they are. Companies that deliver on or exceed these expectations with high quality and velocity will win the battle for customer business and loyalty.

The Kitty Hawk Venture portrays real challenges the modern DevOps professional is faced with. It takes you on a journey from crisis to transformation and demonstrates how powerful continuous testing can be. I found myself regularly saying, 'Yes, I've been there, I've seen that, I can relate to it.'

Though the story revolves around an airline, the lessons and scenarios apply to every industry that must navigate internal politics, influence decisions, and provide digital solutions to their customers."

—Silvia Prickel, Managing Director, Enterprise Quality & Release Management, United Airlines

"*The Kitty Hawk Venture* is *The Phoenix Project* of continuous testing—a great read for anyone unsure or unclear about the importance of modern testing practices in DevOps and technology transformations."

—Nicole Forsgren, Founder & CEO, DevOps Research and Assessment

"*The Kitty Hawk Venture* is a cautionary tale that shines a light on a world where digital transformation and the app economy determine winners and losers every day—either shift left or die. Written in the spirit of *The Phoenix Project*, this novel keeps the reader engaged with real-to-life characters and problems—the names are changed to protect the innocent and guilty. Believe it or not, there are still many organizations that don't test until actual deployment. It is amazing that in 2018, this book still strikes so close to home for so many of us. Great read!

—Alan Shimel, Editor-in-Chief, DevOps.com

"With continuous testing, companies face unprecedented cultural change, with the roles of developers, testers, security, and the executive all rearranging and blending. This book is rooted in real-world examples and experiences. It's not a text book of terms or high-level philosophies. It's a novel that helps tell the story of how the needs of continuous testing can be identified and implemented using proven practices and techniques."

—Adam Auerbach, Vice President of Quality Engineering for Epam

"It's the end of the age of boring books about DevOps. This intelligent novel guides you through Leigh's continuous testing journey. The authors crafted a story sharp as a shark and as exciting as driving a Porsche. In its soft narrative, *The Kitty Hawk Venture* weaves common-sense knowledge for any DevOps strategist together with a fun-to-read narrative. By the end of the journey, the reader learns that quality comes by design, not by accident. What's more, the authors provide a practical guide for continuous testing for success in DevOps. Can't wait to watch the movie."

—João Bezerra Leite, Managing Director, Technology Quality Assurance, Banco Itau, São Paulo, Brazil

"Continuous testing remains one of the most significant bottlenecks to agile DevOps strategies and to competitive transitions to digital transformation. Lack of effective quality strategies straightjackets organizations and undermines execution in visceral ways. This book provides a readable, engaging, and thorough primer for end-to-end execution for continuous testing—from inception through to security and metrics—using a 'real-world' example with relatable personalities. I recommend this book for those seeking to engage in effective quality and DevOps strategies during a time that urgently demands this kind of transition."

—Melinda Ballou, Research Director for IDC's Agile ALM, Quality and Portfolio Strategies Service

I would like to dedicate this book to my wife, Aruna Ravichandran. She is both my partner in life and in writing this book; she is truly the dawn of my every day.

—Jeffrey Scheaffer

It's not often that you get to write a book with your husband. I was lucky to co-author this book with my husband, Jeffrey Scheaffer. I would like to dedicate this book to him for his support, his passion, and his energy, which drives me to excel in anything I do. I also would like to dedicate this book to my daughters, Tanya and Monica Ravichandran, who are the lights of my life.

—Aruna Ravichandran

To my loving wife, for her continuous support and inspiration throughout this amazing life together.

—Alex Martins

To Deliver Continuously, Test Continuously

Testing as a one-shot afterthought won't cut it for fast and short delivery iterations. Testing . . . must enable continuous design, development, and execution of tests at every instance of new code.

The Need for Continuous Testing

Application development leaders in customer-obsessed organizations are looking for ways to build efficient continuous software delivery pipelines to deliver business value faster. Agile, and increasingly DevOps, practices are disrupting old ways of testing software and applications to keep up with the increasing demand for quality at speed. Testing comes earlier in the cycle (shifting left) and is becoming faster, better, and smarter.

"Vendor Landscape: Continuous Testing Services For Agile and DevOps Environments" by Diego Lo Giudice, Forrester Research, Inc., April 14, 2017

Contents

About the Authors

Jeffrey Scheaffer, General Manager & SVP Continuous Delivery, CA Technologies

As Senior Vice President and General Manager for the Application Delivery Business Unit at CA Technologies, Jeff oversees DevOps innovations, which include continuous testing, automated application testing, test data management, service virtualization, and continuous delivery.

He is well versed as an innovation leader in the industry, having spent more than 20 years in IT Operations Management at Hewlett Packard Enterprise, where he served as Vice President and General Manager for the End User Experience Business Unit, which included next-generation SaaS and DevOps products.

Jeff is highly regarded as an expert and a media resource. His writings have appeared in *CIO Magazine*, *CIO Review*, and *Software Magazine*, and he has delivered keynote presentations at events including DevOps Expo and Cloud Expo.

Jeff was recently recognized by his industry peers as DevOps Leader of the Year at the 2018 DevOps Excellence Awards.

He holds an MBA from the University of Colorado and a BS in engineering from Colorado State University.

Aruna Ravichandran, Vice President, Product Marketing–DevOps, CA Technologies

Aruna is a renowned thought leader, a published author, an executive, and an expert in DevOps and digital transformation. She appears regularly in major media, such as *Forbes*, *Wired*, and *Information Week*, and is in high demand as a keynote speaker, having presented at industry conferences such as DevOps Summit Sweden, Gartner, Cloud Expo, CA World, and HP Discover. She co-chairs DevOps Expo for Syscon Media and plays an active role in leading the agenda for DevOps Expo every year. Aruna is also an independent blogger for the Forbes Technology Council, a private invite-only group hosted by *Forbes* magazine.

She created the "Women in DevOps" panel, which invites female industry leaders in DevOps and IT to share best practices, describe their challenges and victories, and nurture future leaders.

Her book, *DevOps for Digital Leaders*, guides readers through several DevOps customer examples and shares the best practices that generate optimized return on investment. In the six months following its December 2016 publication, it had more than 50,000 downloads.

In 2016, Aruna was named one of the Top 100 Most Influential Women in Silicon Valley by the *San Jose Business Journal* and received the 2016 Most Powerful and Influential Woman Award from the National Diversity Council.

Aruna holds a master's degree in computer engineering and an MBA, both from Santa Clara University.

Alex Martins, Chief Technology Officer–Continuous Testing, CA Technologies

Aside from his formidable writing skills, Alex is a veteran of the technology industry, having garnered extensive international business experience in agile software engineering, continuous testing, and DevOps, a large part of this in the airline industry while working for one of the world's most recognized passenger carriers.

Alex has worked both sides of the wall, starting out as a developer and then moving into software testing, which gives him valuable insight into the past and the future of software engineering. His career has seen him rise through the ranks to lead quality-engineering practices across multiple enterprise companies in different industries all around the world.

Focused on both leadership and best practices, Alex has regularly led the transformation of testing and quality assurance and has designed the necessary organizational structures to support culture change vital to agile, DevOps, and beyond. This includes leading worldwide cross-functional teams into continuous testing.

He writes blogs for different CA Technologies social media outlets and delivers keynote speeches and webinars at national and international conferences for both technical and executive audiences.

Alex graduated cum laude with an MBA from the University of Miami School of Business and also holds an MBA and a Bachelor of Science in Computer Science.

About the Contributor

Eric Odell, Director, Continuous Testing Solutions Marketing, CA Technologies

Eric is a storyteller and messaging wizard who applies his 20 years of experience to building product messaging and positioning for SaaS-based technology solutions through CA's Product Marketing division.

His specialty is helping organizations understand the requirements of their audience personas and crafting messages that best tell the story to ensure an optimum match between product and customer need.

People are the focus of Eric's work, and his skill in persona development and story development help transform sophisticated software innovation into truly practical solutions for CA's customers.

About the Technical Reviewers

Stephen Tyler, VP R&D, Continuous Delivery, CA Technologies

As head of engineering for the Continuous Delivery Business Unit at CA Technologies, Stephen is an experienced business and technical leader with over 25 years of software development experience across a wide variety of company types, geographies, industry verticals, and technologies, from mainframe to mobile and cloud.

As a portfolio-level engineering leader, he is experienced in leading agile practice and technical excellence across product teams. His focus is on driving change initiatives in process, tools, architecture, organizational structure, and skills. Stephen's goal is to increase customer intimacy through user research and telemetry, and to enhance core efficiencies that create the capacity to invest in new product ideation and development within budget constraints.

Stephen Feloney, VP Product Management, Continuous Delivery, CA Technologies

Stephen is a product management veteran responsible for CA's Continuous Delivery Business Unit, which comprises service virtualization, application testing, release automation, test-data management, agile requirements designer, and BlazeMeter.

Prior to this role and for the last 12 years, Stephen has been focused on enterprise software at various companies, from startups through to HP, focusing on enterprise software testing.

Margaret Lee, SVP Product Management, Continuous Delivery, CA Technologies

Margaret leads product management, strategy, and customer success teams for CA's Continuous Delivery Business Unit, which makes her responsible for CA's continuous testing and release product portfolio, including agile requirements design, application testing, service virtualization, test-data management, and release management product lines.

She brings 20 years of experience in enterprise software and technologies, including being part of the original development and product management team that started and grew Oracle's Fusion Middleware product suite from an internal start up to a $4 billion per year business. Her career also includes time spent at Splunk and McKinsey & Company.

Foreword by Adam Auerbach

We hear a great deal about digital transformation, a term that somehow evokes the idea of a singular smooth and successful conversion from one complete state to another. Movies have helped support this perception enormously. CGI animation gives us a front-row seat as cars, machines, and all sorts of living and alien creatures turn into other types of beings in about three seconds flat.

The tech sector is not immune to the fascination with and ultimately the fear of change. Those of us who work in technology are amazed by the innovations and improvements that appear daily, yet many of us also hold on to traditions, roles, and tools that we have known since our earliest days. When the industry knocks on the door with promises of the next best thing, we cannot fail to remember that for every new benefit we stand to gain, we are guaranteed to have to leave something of ourselves behind or run into something or someone trying to prevent us from moving forward.

Regardless, software is transforming. But it is not doing so in one fell swoop. It is lurching forward in steps, shaking off the sequential traditions and hierarchies of waterfall and seeking to replace it—eventually—with something more continuous and open. Companies must take care to ensure they are doing this transformation right. Those who have embraced DevOps, the ones who are starting to roll out continuous integration and work toward continuous delivery, may not yet see the red flag problem: the missing link called continuous testing.

Continuous testing is the ability to have all the required testing needed to certify a piece of code automated and automatically triggered by the continuous integration pipeline. Each of the different suites of tests are able to talk to the pipeline, informing it as to whether it can proceed to the next step, without requiring human interaction. It can be considered the missing link because of the way it creates transparency and dependency on the testing process so thoroughly and completely, from check-in to production, rather than waiting until all the changes are aggregated toward the end or for someone to give the go-ahead.

Unfortunately, most organizations haven't gotten to this point yet. For them, there is no clear path, largely as a result of a lack of agreement around how to do testing in a world of agile sprints, very short timeframes, and the need to optimize the lead time to production without sacrificing quality. It also takes a higher level of automation maturity to be able to stop and start a pipeline based on automated tests.

Moving faster without quality is very dangerous. Many companies look at quality based on the success rate of production changes. If you have more releases and you have not improved your quality, you are going to introduce more production incidents. Implementing continuous testing ensures that the quality standards are not sacrificed with DevOps automation. With continuous testing, however, companies face unprecedented cultural change, with the roles of developers, testers, security, and the executive all rearranging and blending.

Ultimately, the app-driven marketplace demands continuous delivery. But you can't get to continuous delivery without continuous testing. This is the call to action that must be heard. And it underscores a key point: that a digital transformation isn't all digital. As with all of business, even that which resides inside binary code, the necessary transformation is ultimately human.

This book is rooted in real-world examples and experiences. It is not a text book of terms or high-level philosophies. It's a novel that helps tell the story of how the needs of continuous testing can be identified and implemented using proven practices and techniques.

Adam Auerbach is the Vice President of Quality Engineering at Epam Systems, a leading global provider of digital platform engineering and software development services. Before joining Epam, Adam served as the VP of Quality and DevOps Engineering at Lincoln Financial Group, where Adam was responsible for introducing and leading the DevOps and Quality Engineering transformation across technology. Prior to joining Lincoln, Adam was the senior director of technology for advanced testing and release services at Capital One Financial Corporation. While at Capital One, he provided leadership for the agile transformation of their quality assurance group and led the enterprise adoption of DevOps and continuous testing.

Acknowledgments

The authors wish to thank the following people, whose wisdom, time, and commentary were invaluable in helping *The Kitty Hawk Venture* take off:

Eric Odell of CA Technologies, who managed the creation of this novel from inception to completion, including background research, story ideation, story development and outlining, persona development, interviews of subject matter experts, editorial review, project management, creative review, and team leadership. Eric also served as chief diplomat.

Adam Auerbach of Epam, who assisted in persona development and technical review and also shared his knowledge, experience, insights, and war stories from the continuous testing and DevOps frontier.

Silvia Prickel of United Airlines, who shared her knowledge, experience, insights, and stories in continuous testing and DevOps as well as provided an insider's view of the airline industry.

Anaf Durrani of Cigna Health, who delivered excellent insight around continuous testing and DevOps, especially regarding the personalities involved in each of its areas.

Stephen Tyler of CA Technologies, who provided advanced strategic assessment of continuous testing and DevOps, along with close scrutiny of the technical material in the manuscript.

Stephen Feloney of CA Technologies, who provided generous amounts of technical review, discussion, and insight, all with a great sense of humor.

Margaret Lee of CA Technologies, who delivered her usual level of indefatigable excellence, providing clear insight into the value of continuous testing and DevOps and its implementation into both pipeline and culture.

Peter Chestna of CA Veracode, who provided essential insight into the evolving world of security in continuous testing.

Dave Karow and Refael Botbol Weiss of CA BlazeMeter for their technical review.

Diego Lo Giudice of Forrester, Melinda Ballou of IDC, and Joachim Hershmann of Gartner are highly regarded analysts and experts who have delivered inspiration to their colleagues and readers through countless inquiries, briefings, and articles on DevOps and the Continuous Testing Journey.

Introduction

This is the story of one individual, Leigh Freemark, and her IT team, who work at Renway Air, a medium-sized passenger airline based out of Boston. Together, they quickly find themselves in the crosshairs of technological change. Struggling to recover from a publicly embarrassing IT issue, they realize that the foundation for a successful cultural transformation to adopt agile and DevOps requires a dramatic shift—one that demands a new and radical approach to developing and testing the software designed to keep airplanes safe, ready, available, and profitable. It's called continuous testing.

Leigh and her team recognize that a continuous testing journey has several steps. You don't just flip a switch. Just like embarking on any type of venture, you must know where you are, what you seek, what the barriers might be, how you measure progress, and how to keep things going once you have arrived. Some of the barriers and obstacles might be technological and physical in nature, but as Leigh finds out, others are very human.

The Kitty Hawk Venture illustrates the beginnings of the continuous testing journey in chapters that approach each challenge in turn. The airline is fictional, of course, but the procedures, people, and pressures within will likely prove very familiar. The book concludes with a flight plan—a summary of the things Leigh has learned in her journey that should prove useful as a small reference guide unto itself.

We tend to assume that IT people live on the bleeding edge of technological innovation if only out of sheer fascination, but in truth many who work to keep an organization's applications running find their energies given over instead to meetings, crises, and politics, leaving little room for keeping pace with innovation. Band-Aids and patches serve to hold things together, and companies lurch into the headwinds of change rather than finding the jet stream and cruising the distance.

But before starting on Leigh's adventure, let's look briefly at the context. The marketplace in which Renway Air finds itself today is similar to that encountered by companies in any industry.

Everywhere you look, the pace of change is accelerating. As consumers, our choices include online shopping, making price comparisons through

our phones, and even paying on the spot without lining up at the checkout. Customer loyalty comes and goes in minutes and is tracked constantly by apps. The era of social media has redefined where people learn about the world around them, shifting from trusted professional newsreaders to online feeds that already match their beliefs and biases. People seldom visit their bank branches anymore, and to be without a smartphone is, for most, a highly unsettling experience.

In retail and industry alike, people expect more. They feel more empowered. Convenience rules, and disgruntled passengers and shoppers have the entire social media spectrum to use as their voice. Companies in every business channel are starting to recognize that the vital link between themselves and their customers exists online and through apps. This is how and where all aspects of the relationship occur.

Apps are the vital connection. They make commerce happen. But speed and quality have never lived together comfortably in any area of business, and app development is no exception. The dominant tradition of designing and testing code has focused on processes overseen sequentially by specific types of experts, and testing of this code has relied largely on a linear sequence that extends beyond the moment of public release. Timelines for spotting and fixing errors were longer, and individual teams' preferred methods of doing things became as much a part of an organization's legacy as the systems themselves.

For some, though, this awareness is coming too late. They stand and watch helplessly as brash startups swoop in and take over, erasing decades of goodwill and steady growth with a tap on a screen. These startups, unencumbered by long-term debts that were agreed upon in a different age, move nimbly into the corners where individual customers await. Other organizations resist, believing in a management and marketing mindset that was formed one or more decades ago. And still other organizations, who do indeed see the future coming, struggle with how to match their legacy systems and corporate culture to pursue these new horizons.

Every company now depends on its IT team for every facet of its existence, from promotion to sales to data and administration. There is no area of operation that is not affected and supported by IT. Yet, the internal culture of IT struggles with the same types of change. The process of developing code, from initial requirements all the way through to production and use by the end customer, still relies on established traditions and techniques around development, testing (functional, performance, and security), and quality assurance. The challenges and risks to IT are both human and technological. Failure and success are both there for the taking.

Change never comes easy, and the need to embrace continuous testing and to leverage its flexibility and quality forces Leigh and Renway Air to consider setting a new flight plan toward innovation. Doing so requires a journey of discovery in which Leigh identifies the steps needed to establish a continuous testing culture, while at the same time confronting the impediments, both human and technological, that threaten her airline's very existence and the futures of all involved.

CHAPTER

1

Cutover

Main Hangar, Renway Air, Monday

"It looks like a shark," Leigh Freemark thought to herself as she stared at the Boeing 737 parked directly in front of her. "A Basker or a White. A fat body with a short nose. Not like a sleek Mako or a sharp-snouted Sandtiger." She knew sharks. She loved them. She had worked closely with them in those special days during her first year of undergrad when she'd volunteered on a university research team, tagging the various breeds that patrolled up and down the New England coast. They would pull the smaller ones on board the boat, take some blood, weigh them, attach a tracker, and let them go. As the junior research volunteer, she always ended up perching on any surface she could find, usually a Coleman cooler wedged against the gunwale, with a toughened, waterproof laptop on her lap, entering the specs that were called out to her by the senior researchers, who carefully held and inspected the beast and equally carefully let it go again.

It wasn't just the shape of the body that made these planes shark-like to her. It was the cockpit windows too. They formed the face of the plane, and although they were not on either side of the body, like sharks' eyes were, they were just as empty, cold, dispassionate. A plane without a crew looked blind, without life, without a soul.

The body was smooth, designed in both cases—shark and plane—for frictionless motion through the seas that they respectively called their homes, one of salt, the other of wind. For both, balance was maintained by a tall tailfin—the signature that informed the world of their presence.

J. Scheaffer et al., *The Kitty Hawk Venture*, https://doi.org/10.1007/978-1-4842-3661-1_1

Those wonderful afternoons rocking in a tiny boat—the shouts of the crew, the cry of seabirds, the sense of strength and power that came from the waves and from the muscular twisting of the ancient creature, fearless even when pulled temporarily from its element. This all flooded back to her as she looked up at the plane in front of her. She had left the ocean behind once her studies had become more serious and her career more real. Now, it was planes, like this 737. She was in a hangar about twenty feet from the forward landing gear. The sense of strength and power still surrounded her. The air was heavy with the smell of tire rubber and oil, of metal and a lingering wisp of jet fuel. And god, she was tired.

As Senior Director of Quality Assurance for the IT department of Renway Air, there were far more airplanes than sharks in her daily schedule, although seeing a plane up close like this was still a rare occurrence. The IT department did not have offices at the airport. But here she was. It was 7:05 Monday morning. Two hours ago, she had received a call and two text messages from the assistant to Renway COO Helen Humphries, instructing her to go straight to Hangar 7A. She had made it there by 6:00 a.m. sharp, only to be told to cool her heels while Ms. Humphries spoke to the media.

Leigh sipped her coffee from a thermos. At least that way it looked less like she had stopped off on the way, which, in truth, she had. Because, well, coffee. And besides, there was apparently none on offer here in the hangar. She checked her phone for messages. Nothing she hadn't already read this morning. She looked at the plane again. Like most commercial airliners, its livery was mostly white, but with clean dark blue and purple pinstriping that ran the length of its body, ending as a stylized letter "R" on the tail. It told the world this plane was part of Renway Air, the second-largest regional passenger airline in the Eastern United States.

Three metal folding chairs had been arranged here, in a row, in the hangar, in front of the plane's nose. A wood and canvas director's chair had been placed to face the three. Leigh guessed they had been arranged this way by one of Helen Humphries' assistants to maximize the drama of the lecture that was about to happen. This could have gone down in the boardroom back at the office, but Leigh knew that wasn't the Helen Humphries way.

Occupying one of the metal seats was Alicia Aiken, CIO—Leigh's boss's boss. Seated on another was Tyler Wiggins,VP of Corporate Communications. He was talking to someone on his phone. Leigh sat down on the third chair and watched how the lights of the hangar reflected so perfectly off the fuselage. It was a nice distraction from what she knew was coming. Planes always looked so clean. Last week's PR disaster could not take the shine away from its perfect skin.

The media had called it a computer glitch, and for two days the story had occupied some sort of no man's land between real news and "on the lighter side." It had happened just over a week ago. Renway's reservations platform

operated on a 32-bit mainframe, and the goal had been to convert all the programs to 64-bit via a hard cutover. To do this, many systems, including the main site at renwayair.com, had been made unavailable from 1:00 a.m. to 5:00 a.m. eastern time of that Sunday morning. The entire IT staff had been on deck for the switch, and they had spent a number of those hours doing the testing in production—a customary practice in legacy organizations like Renway, which was itself the product of many decades of slow and patchy technological evolution in the airline business.

But it had not worked. Not as planned, anyway.

The following Monday morning, breakfast TV show hosts and morning drive-time radio jocks had all put it something like this:

> "If your travel plans include flying anywhere on the East coast this holiday season, you might want to check with your travel agent or online to make sure you actually have a plane waiting for you! Boston-based Renway Air announced today that an outage affecting its reservation system is preventing people from booking flights and might even affect the availability of flight crews to pilot the planes during the critical holiday travel period. Helen Humphries, the airline's Chief Operating Officer, released a statement saying the outage is the result of a software upgrade and will be resolved shortly. 'At no time,' she says, 'is public safety at risk. We expect the system to be back up and running within twenty-four hours.' Customers looking to book flights for today through Wednesday are advised to call Renway's customer service hotline . . ."

That was a week ago. The problem had been quickly fixed. A mere papercut in the history of the airline's IT operations, but this time it had resulted in an unexpected virulent infection. Experts and social media memes were calling Renway out as yet another symbol of a broken system filled with delays, exorbitant fees, excessive security measures, and shrinking comfort. Renway was in danger of becoming the final straw after decades of passenger frustration.

Leigh knew that the "dungeon" where so many of her IT crew worked was humming right now, and she wished she was back there. She *should* be back there. But here she was in Hangar 7A, waiting for Helen Humphries to wrap up the last of six live on-camera interviews with those same breakfast TV news media outlets. It was one week since the cutover, and she was still doing damage control. The story had not stopped. It was standard practice

for Renway's own corporate communications people to set up the camera and lights in front of one of the planes and then feed the signal directly to the TV stations, whose hosts could do interviews without leaving their anchor desks. That's why Tyler was here, obviously.

The crew was wrapping up now. The lights snapped off. Helen Humphries removed her earpiece, and one of her assistants handed her a coffee. Apparently, there was some somewhere. She smiled a tired smile.

"These past seven days have been excessively busy and unpleasant for me," she said. "I have been up since very early this morning. Our CEO, Bob Cartwright, called me at 4 a.m. and asked me to do a charm offensive with the media today. One full week after the reservation system came back online, I am still having to explain what the technical glitch was and whether it means our planes are unsafe."

She sighed deeply, like a schoolteacher listening to a late-assignment excuse. "I asked you here, Alicia, because obviously as CIO, this whole thing happened on your watch. And Ms. Freemark, you are here on Alicia's recommendation, as well as to fill in for Derek, who is still in the hospital. A great deal of damage has now been done. The media is talking, and passengers are still complaining. We cannot afford this type of stain on our brand and reputation. It never entirely fades away.

"What I need to know now is how this happened, who should have caught it when, and what we are going to do to make sure this does not happen again. I am charging you with the responsibility of finding this out, and the clock is ticking as of now. I am not promising amnesty here. Somebody—or bodies—in your IT circles might no longer be here when this is all over. That's how it goes. I do not want a blame game, and I do not want excuses. I want to know why our system, whose quality is supposed to be assured by your department, failed us, and failed our customers and shareholders. Then, we will let the chips fall where they may."

CHAPTER

2

Why Testing Is Imperative Now

Leigh felt the power of her old Porsche's rear engine as it pushed its wide tires from the merge lane onto the highway. The Hangar 7A meeting was over. Now, she was heading away from the airport back toward Collingwood. That's where Renway Air had its administrative offices and data center, with her office being, as per the norm for QA and IT drudges, in the basement. The dungeon.

She checked the clock on the console: 8:15 a.m. The car had been a gift to herself. She had bought it third-hand, eight years old, but it still looked and felt new. It was an admittedly selfish acknowledgment of her promotion to Senior Director, Quality Assurance just last year. This marked a milestone of sorts, after five years of slogging it out as a QA manager, and one year as director, all following her graduation from a Boston-area university with a joint master's degree in mathematics and economics.

Hers was an unconventional background for a career in IT, made stranger still by her undergraduate degree in marine biology. But sticking to the straight and narrow had never been her thing. She loved numbers. She loved logic. She loved precision. But she also loved randomness and fate. The two forces seemed to form the dance of life itself: the strict rules of physics and chemistry versus the arbitrary nature of hurricanes and earthquakes. The perfect binary atmosphere of computing versus a human race obsessed with creating viruses and bots.

J. Scheaffer et al., *The Kitty Hawk Venture*, https://doi.org/10.1007/978-1-4842-3661-1_2

The safe career route had seemed too easy. This was a quirk, she knew. She had always loved placing her logical mind in the hands of fate. Her rise through the ranks at Renway had not been planned, but it had been helped along by the three years she had worked in IT at an airline catering firm, FoodFlight, during her undergrad years. Her love of travel had originally gotten her in at Renway as a temporary reservations agent, which paid infinitely more than volunteering to tag sharks ever did, so she had switched from a boat to a desk.

She had quickly figured out the workings of the Renway reservations system, as well as of global systems like Sabre, and discovered she had a talent for data and computing. So, ironically, her passion for travel had kept her inside the walls of Renway, slowly tunneling through the dark depths of IT until she had surfaced inside the office of the Senior Director of QA. Consequently, with no time available to go backpacking in Europe, she had bought herself a fast car instead. On those rare occasions when she could truly floor it, it indeed felt like a plane accelerating for takeoff. So, call it market research. It was a nice car, and she was proud to own it.

Anyway, the Porsche was her second office. She spent more time in it than anywhere else, including, it seemed, home. There always seemed to be a crisis of some sort either out at the airport or at one of the company's two outlying buildings that were situated inexplicably on opposite sides of the city. And when there was no crisis, there were always meetings. She had grown used to pulling into any convenient mall or empty street-side parking space and hopping from the driver's seat over the console—without hitting the gear shift—to the passenger seat, where she could use her laptop comfortably. At least half of her meetings were done like this.

This time, however, she wanted the team together in person, and as she drove away from Renway Hangar 7A, she told them so.

Her phone beeped. It was Dinesh, Director of Marketing. She had known Dinesh for almost a decade now. When she had first started working in reservations, he had been her boss. He had continued to pursue his own postgraduate studies in computer science, only to find his passions lay more with customers than with computers. He had quickly offered his help in coordinating the meeting and rounding up the team.

"How far out are you?" he asked, forgoing any usual morning pleasantries.

"Thirty-five minutes, max, plus five for the elevator," she said. "I should be there just past nine. Has everyone confirmed for 9:30?"

"Yup," Dinesh replied. "The media is recirculating Helen's bits from this morning, but they're making it look much bigger than it ever was."

"Yeah, well," Leigh replied, "as you know, when the media makes it look bigger, it becomes bigger. And there will be a cost."

"I would suggest our bigger problem here is dotPlane," he replied.

dotPlane, whose logo used a small "d," a dot for the letter "o," and an @ sign for the "a", as in d•tPl@ne, was Renway's biggest competitor for short- and medium-haul flights up and down the eastern seaboard. There were larger airlines, of course—JetBlue, United, and others—but dotPlane was a point-for-point direct competitor, smaller than the majors but bigger than the puddle jumpers. It was less than five years old, a brash startup specifically aimed at younger customers and business travelers.

dotPlane had been started by the current owners of Carberry, a London fashion house with a history stretching back almost 200 years. Carberry offered fashionable products that did not impose a six-month wait between the fashion show and the retail floor. It had expanded into other forms of media, with musicians, songs, and entertainment falling under its brand. These acts then performed at Carberry fashion shows in Paris, Milan, and New York, which became trending social media events. Carberry used data to listen to its customers and then sold them what they wanted, right then and there. They used mobile apps to transform the shopping experience and pioneered on-demand personalized deliveries using services like Uber and Lyft. They were even making strides in adopting blockchain technology to manage all aspects of their supply chain. In short, its owners had brought a two-century-old industry forward to the leading edge of digital commerce.

When dotPlane's CEO was asked how they could get into the airline business without a deep knowledge of the industry, she had replied, "I don't know. Let's take an Uber to Richard Branson's house and ask him how *he* did it." She was referring, of course, to Branson's brilliant off-the-cuff founding of his airline by crowdsourcing a charter to the Virgin Islands. Her mention of Uber was equally strategic.

Within hours of Renway's outage hitting the media, dotPlane had launched a major campaign offering seats to any Renway passenger who had been caught short:

> Lost your vacation? We found it. It's over here!

Downloads of the dotPlane mobile app skyrocketed, and every dotPlane passenger was given a points bonus, just because. On the Monday of last week, as the cutover fiasco news had spread, anyone booking a flight on dotPlane to or from Boston got 25 percent off their fare no matter which way they booked it. It was a loss leader, sure, but they were already trending on Twitter: #dotPlaneJustBecause.

"Man, that was fast," Leigh exclaimed as she passed through the industrial belt and headed on into Collingwood unobstructed. Traffic was light. That was one of the benefits of driving the opposite direction of the Boston-bound rush hour. She could already see the Century building, where her team would be waiting.

"You mean the traffic?" Dinesh asked.

"No, I mean dotPlane's work last week. They turned this snafu around on us on a dime."

"Yes," said Dinesh. "I think it's time we bring out the dog."

"Agreed," said Leigh. "I'll see you at 9:30."

Immediately after hanging up with Dinesh, she exited the highway and pulled into a roadside picnic area. She picked up her phone and pressed the app for Veneto Burrito. They delivered the best Italian and Mexican brunches and lunches anywhere, hands down. She placed a brunch order for nine people to be delivered at 9:15. Although staff meetings were not usually catered, she wanted everyone's best talents this morning, and this was the best fuel she could think of.

* * *

At 9:15, Leigh was in the 16th floor boardroom connecting her laptop to the projector. Mary, the floor receptionist, opened the boardroom doors and said, "Leigh, your order is here." Two delivery people in red overalls and red ballcaps wheeled in two carts, each laden with three insulated boxes. The carts, the overalls, and the boxes all had the black and red logo of an upended fork, looking like a rocket, with flames shooting out of its tines. Alongside it was the name: Veneto Burrito.

"Perfectly on time, as usual," Leigh said to the older of the two deliverymen. "On the table along that wall would be perfect." They proceeded to unpack the boxes. Each of the meals was also tightly contained in heat-retaining packages, which they laid out along with condiments, plastic cutlery, and two portable boxes containing coffee. She noted their tagline, silk-screened onto each of the boxes: *Point. Click. Burrito*.

Leigh went back to the work of setting up her place at the head of the boardroom table. She had switched the computer screen over to a news channel, and, sure enough, there was Helen Humphries, speaking with calm authority directly into the camera. The 737 formed the backdrop to her interview, the Renway logo clearly visible. She reached for the remote and unmuted the screen.

"... at no time is public safety at risk, David," Helen said to the news anchor. "This is a routine upgrade that will make our passengers' travels even more convenient and enjoyable."

"Sounds like you guys didn't test enough," said a voice. It came from the back of the room. Leigh muted the TV and looked to where the comment had come from. There was no one there except the two deliverymen, who were finishing up their setup and stacking the insulated boxes back on one of the carts.

"Excuse me?" asked Leigh.

The two men continued working. The older of the two spoke but did not turn to face her. "I said, I guess you didn't test enough. It wasn't ready. It sounds like you guys don't know what you are doing."

Leigh pressed her left hand to her forehead and pinched the skin a little in disbelief. She was getting heckled by a fast-food delivery guy, although she thought the voice sounded a little familiar.

"I'm sorry," she said, "and you are?"

Finally, he walked across the boardroom toward her, removing his cap and extending his hand in greeting. Leigh felt a rush of recognition as he said, with mock formality, "I'm Steve, from Veneto Burrito."

"Steve Fibonacci!" she exclaimed, recognizing her old boss from her undergrad years.

"It's nice to see you again, Leigh."

"I'm sorry, I'm a little shocked," said Leigh. "You were VP at FoodFlight when I was there."

"And you were a very smart programmer," Steve replied. "I had wanted you to stay on, but you were full of ideas and dreams if I remember correctly. I had to let you follow your own path. Now, you're an airline bigshot."

"Hardly a bigshot," Leigh replied. "Senior Director of QA."

"Another rung on the ladder," Steve replied. He turned to the other deliveryman, who was much younger. Probably another undergrad. "Jerry, take the truck back to the office. I'll catch up with you later. You can manage both carts?"

"Yes, sir," said Jerry, "no problem."

As Jerry expertly maneuvered the carts out of the boardroom, Leigh said, as diplomatically as she could, "So, you're still in the food business, obviously?"

Steve smiled warmly again. "Don't let the uniform and nametag fool you. I sold FoodFlight to a multinational. Could have retired then and there. But I don't know how to fish, and I don't want to drive an RV. So, I bought this company. Veneto Burrito. It had forty franchises back then, and now I have added two hundred more across North America. I'm the boss. I own it all."

He continued, "I make it a point to monitor some of the incoming orders and go out on deliveries at least once a week with my crew. How else are you really going to stay in tune with how the business runs on the street level? You can't know your customers from the corner office. When your order came in through the app, I thought to myself I just had to pop in and say 'Hi,' you know? I had been meaning to catch up with you these past few years. But now, with

your little PR problem, well, how better to do it than with a cartload of *huevos rancheros italianos*? It's a perfect parallel."

"Parallel?" Leigh asked.

"Yes," said Steve. "Your IT problem is like our *huevos rancheros italianos*. Scrambled. And your internal proceedings aren't exactly internal right now. They're all over the public breakfast table."

He looked at her more closely. Leigh relished seeing the twinkle in his eyes after all these years.

"Look," said Steve, "Renway has a problem that I think I know a lot about. You know that. We had the same types of issues back at FoodFlight. When I was considering buying Veneto Burrito, their software development process was the first and the last area I explored. And I explored it very thoroughly. I shouldn't get involved with the politics or the confidentiality of your group, but hey, that's why I sneaked in. Let's get together sometime in the next couple of days. Let me share with you a little of what I have learned since you left. Time may be of the essence here."

He looked at the people starting to enter the boardroom. "What's your plan for the meeting? Not a post-mortem, is it?"

"No," said Leigh, "that was done last week. I have to find a way to change the culture of testing going forward. We have to change the way software is created and tested. More precisely, our CIO, who is my boss's boss, wants me to spearhead this."

"Good!" said Steve. "Make sure that imperative is underscored. Get them to start thinking about this brave new world where everyone wants new things built quickly but with quality. Get them to start talking about doing more testing all the way through the software development life cycle. Get them to see that testing is everyone's responsibility, not just the testers and not just at the end. That's the imperative. You got that?" Then he added, "it's just a suggestion."

"Thank you," said Leigh. Her head was spinning. Steve fished around in his apron pocket and brought out a business card holder. He handed a card to Leigh. It was a photo of half a walnut, with his name appearing as if it had been laser-engraved on the side.

"When I pass away," he began, "I want my ashes to be placed inside one of these, so that when people come to see it at my memorial, they'll all say, 'That's Steve in a nutshell!'" He then broke out in a paroxysm of laughter. He winked. "My number is on the back."

Indeed, it was, along with the Veneto Burrito logo and all the other details. His email address was 1235813@venetoburrito.com. "Nice touch," she thought. As anyone who worked with numbers would know, this was a Fibonacci

sequence, revered by mathematicians for thousands of years. There was also a QR code for downloading the app. He smiled, gave her a small salute, and left the boardroom so she could start the meeting.

* * *

At 9:30 a.m. sharp, Leigh called the meeting to order. They all had their laptops open. They knew the drill. Leigh's meetings always started when promised, always ran quickly, and always ended on time. They would be done by 10:30 at the latest. It's one of the reasons they always showed up punctually. Most of them were still choosing from the Veneto Burrito buffet, but that was OK with Leigh. "As long as you can hear me, we can begin," she said.

She looked around the room and confirmed to herself that everyone was present. Dinesh, Arthur Fergus (VP of Application Development), Jessica Blantyre (Release Manager), Owen Saunders (VP of IT Ops), and Giles Taylor (Senior Director of Mobile Applications). One other figure was standing at the refreshments table, fixing herself a coffee—Alicia Aiken, CIO.

Leigh took her place at the head of the table. On the screen behind her was a photo of a very happy-looking dog. A golden lab. Its brown eyes and natural smile leapt from the screen, but there was something unusual about the photo. The dog was grinning at the camera from a seat inside a large passenger airplane.

She began: "I don't want to talk specifically about what happened. I want to talk about what *is happening*, and what continues to happen. Let me tell you a story." She half turned in order to point at the image of the smiling dog.

"This photo, as you might notice, was taken aboard a plane, where dogs don't usually sit. It came from a rescue effort that happened after a forest fire destroyed a town up in Canada a couple of years back. Once the runways and airports were declared usable, civilian flights resumed. But what the people there discovered was that the major carrier for the region, its computerized revenue management system automatically charged a premium for a same-day booking. That's the norm, obviously. People who were used to paying $150 for a flight were looking at $600 or even $800 per person one way. These were not vacationers, remember. These were people whose homes were about to be destroyed, or already had been.

"The media had a field day with that one. It said the airlines were gouging. People turned to Facebook to troll the airline pretty mercilessly. But we know they weren't gouging. That was just how their pricing system worked. They didn't have any features to override their standard algorithms with special pricing for an emergency situation, and they couldn't add that feature quickly enough to avoid the PR disaster."

She looked over at Dinesh to continue the story, which he did.

"One competing airline, AirMax, a charter, sent some planes up to give anyone who wanted it a seat extra cheap right then and there. And once all the people had gotten out safely, AirMax sent the planes back for the pets. Dogs, cats, turtles, hamsters, you name it. They got them all on the planes and got 'em out of Dodge. Wonderful thing to do, and a huge PR victory to boot."

There was silence in the room.

He continued, "Point one here is, AirMax was able and willing to get their planes—and their crews—back there in a hurry, outside of the normal schedule, and they made huge returns on the effort through unprecedented positive media coverage that turned into bookings and loyalty nationwide. Point two is that their reservations system kicked in a special circumstances pricing routine that gave these people access to cheap seats right away. No need for refunds, credits, or apologies."

Arthur Fergus spoke up. "That has no relevance here," he said. "They pulled some strings to make their planes and crews available on short notice, but that's not an IT issue *per se*. You're trying to make some point that connects to our cutover problem, obviously," he said, "but in our case, it was rectified quickly. We got it fixed within hours. The thing on the news has been way overblown, and I think someone has been fanning the flames on this a little."

"That may be," interjected Alicia, who had remained standing at the window side of the boardroom rather than sitting, "but the point isn't that we fixed this in a few hours. The point is, it should never have happened. We cut over too soon—we used a risky all-or-nothing procedure and then spent hours doing testing in production. The AirMaxes of the world are coming up fast with a far more nimble solution."

She turned to face the rest of the group. "Again, as Leigh has pointed out, this meeting is not about dissecting the cutover. Rather, it's illustrative of a larger point, which is that the public expects a high level of service and instantaneous adaptability from everyone they buy from, and that includes airlines. And, as things go, we as an industry haven't exactly moved with the times. We have not truly embraced any sort of digital transformation."

"In other words," Leigh chimed in, "what we consider good enough, especially with our development and testing cycles, is *not* good enough. On all levels, things need to move faster, but they also need to be *ready and perfect* faster. Quality is everyone's responsibility." She drew in a breath to start making her next point, but Arthur jumped in, ignoring Leigh and instead addressing his CIO.

"With respect, Alicia, as I said, the problem was fixed quickly. We're back on track."

People fidgeted a little in their seats. Alicia did not approve of such escalations. She was cool—an ally of everyone in IT. She had risen quickly in the business,

first at a huge global software company, and then, for the past three years, here at Renway Air. She had done so by being brilliant and open, sharing credit and bearing responsibility in a way that Leigh found inspiring. She was tough and driven, but she was not manipulative. She also managed to make time for family and social events—a concept that was alien to many of Leigh's colleagues as well as herself. Arthur's comment should have been addressed directly to Leigh.

Alicia smiled without a hint of malice or annoyance. "I agree, Arthur. But as Leigh and I just pointed out, this is not about specific past events, but about our future. Even with all the constraints we face as a carrier, we have to catch up. Not to other carriers, but to other companies. In retail. Services. Media. Netflix. Uber. Amazon. That's our competition now."

Arthur continued. "I think we may be making mountains out of molehills here. Things happen. We test. The testing team runs thousands of functional regression tests every time we release the reservations app. They get code coverage of 65 percent, which is very good for a large legacy application. There's almost four million lines of code, performance tested to three times our peak production load."

"Yes," said Leigh, "but it is now apparent we are having production issues with our software more frequently. This might have been OK in the past, but there was once a time when punch cards were OK too."

She hit the Enter key on her keyboard to bring up a new PowerPoint slide. It was a screengrab of a Google search page dated last Monday, when the cutover repairs were still ongoing. The search results were for Renway Air. But the ad on the page was dotPlane:

Lost your vacation? We found it. It's over here!

"Renway's testing needs a reboot," she continued. "However good you think our testing is, it is fundamentally failing to prevent production problems from happening, it is failing to detect and correct issues rapidly when they do, and it is failing to understand the impact on user behavior and business metrics of each change that we release. We need to think very differently about what, how, and when we test, not only to mitigate release risk better and have our applications work reliably, but also to create meaningful competitive advantage. Folks, this is our imperative."

She pointed at Giles as a cue for him to speak. Giles stood up.

"As we all know," he started, "over the past few years, the role of testing in DevOps has become more critical. It has had to change. Once, it was a waterfall thing—regression tests and performance testing toward the end of the cycle. You know, find defects, and then go back and forth.

"Then, as we adopted Continuous Integration (CI) we added a lot of automated tests on nightly builds to detect regressions sooner. Then, as we started to adopt agile, we tried to shift left with acceptance testing. We put out software in smaller chunks but with the expectation that each sprint would deliver software that was not just coded but also tested. We invested in test automation in both QA and CI. There was some success, but typically it just came down to automating regression. Even with quarterly releases, there were constraints, and these became bottlenecks to the process."

Leigh stepped in. "Let's just slow things down a bit here, for everyone's benefit. Not everyone at Renway—or elsewhere, for that matter—is on the same page when it comes to DevOps. It's still a relatively new term for a lot of people. Can you take a moment to explain it from your bleeding-edge perspective?"

Giles raised his eyebrows quizzically. His British accent seemed to emphasize the significance of his message. "It's hardly bleeding edge," he said. "DevOps is about breaking down walls between development and operations to ensure that our software, which is being built through agile methodologies, can be deployed quickly to the users as soon as it is ready, without sacrificing quality." He looked around the room to ensure everyone was still following him. "For that to happen and to mitigate business risk, there has to be a lot of testing from the beginning of the software development lifecycle all the way to production. That's the only way to ensure business stakeholders are comfortable that the users will receive something that has the high quality everyone expects."

He checked the room again. It looked like people were waiting for more. "As they should," he thought to himself.

"The act of doing a build, the act of doing setup and deploying to production—testing is a huge part of all of this," he continued. "Some would say that it shouldn't be called DevOps anymore. Maybe DevTestOps. Oh, but we can't forget security or the business itself. So maybe BizDevTestSecOps. The problem is, it's not evenly distributed. And it doesn't adequately address change."

"Success factors," Giles continued. He walked over to the whiteboard on the wall opposite the PowerPoint screen and began to write a title: "Success Factors." His handwriting was remarkably neat—a rare talent among programmers. He then wrote out five bullets:

1. Throughput
2. From committing code to deploying
3. From months to hours

4. Low performers need weeks or months to move from commit to deployment.
5. Fast recovery rate

Alicia chimed in. "I think what you can derive from this is that our own software needs to be developed and delivered faster. But it also needs to be built to be flexible enough that it can be changed frequently. That way, when the market demands change, or when a natural disaster or an opportunity happens, the company can react to it on a dime and look good with both the press and our customers."

Leigh spoke up. "Here are some numbers I pulled up a couple of days ago I saw from at a DevOps conference just a few weeks back." On the screen was a giant number that filled the whole slide:

63

"Sixty-three percent of software development delays are occurring in the Test-QA practices across the cycle. Huge bottleneck. So, even where DevOps exists, companies like ours are still experiencing a trade-off between quality and speed. But if the app does not work, people will leave it behind. Development has to be fast enough to react to the way the customer behaves. This means testing needs to evolve and be spread out across the software development life cycle (SDLC). You cannot just have tests run manually—we must think differently from now on and be advocates for the customer. Here are some more numbers for you."

Leigh hit the keyboard again.

70

"Seventy percent of testing is still manual."

56

"Fifty-six percent of critical dependencies are unavailable."

50

"Fifty percent of time is spent looking for test data."

64

"Sixty-four percent of defects occur in the requirements phase—"

Arthur got up and started pacing around the back of the boardroom. "And you're going to tell me, again, that doing the things that Giles described, and having developers working with testers, will improve all these numbers? I don't think we can do that right now. We're not equipped. We're not staffed. And we already have enough work to do."

While Arthur spoke, Leigh's phone vibrated on the desk in front of her. Ordinarily she would have ignored it, but her screen was face-up. It was an email from Steve Fibonacci. The subject line read, "This might help. Put it up on PowerPoint!" The message ended with the initials SF and a smiley face emoji. She alt-tabbed across to her email, checking back just once to make sure the PowerPoint presentation, and not her inbox, was on the screen. She opened the email and looked at the image Steve had attached. "Perfect!" she thought. She quickly copy-pasted it to a blank slide in the PowerPoint deck, which updated automatically. All she had to do now was click to it.

Alicia was speaking. "But that's precisely what we—as a culture—have to do. Our people need a better understanding of how developers work. That way, we can add value earlier in the lifecycle by testing earlier than we do traditionally. Yes, that means shifting testing to the left of the lifecycle, which ultimately enables the testers to provide feedback to developers faster. And that is huge. Because developers would love to know that when they check in their code they won't ever have to look back, wouldn't they? It just works better."

Leigh hit Enter on her keyboard to call up the newly inserted slide. A cartoon appeared. It was an image of two Stone Age men laboriously pushing a cart with square wheels past a roadside stand selling round wheels. The gist of the cartoon's joke was that they didn't have time to look at the new wheel since they were too busy pushing the cart.

Arthur shook his head in obvious disagreement. He apparently failed to see either humor or relevance in the image.

Giles smiled at him. "I understand your concern, Art." He knew Arthur hated being called Art. That's why he did it. "Some developers and testers can make that shift-left transition and like it. Others can't handle it. From either camp. I know that. But that doesn't change the fact that we have to change who is able to do the testing."

Leigh's phone sounded again. This time it was her alarm clock app.

"OK," she said, "time to move the meeting toward closure. Our objective was to discuss the future rather than the past. The message here is that we need to change the culture and process around the development, testing, and quality of our products; that a better form of testing is our new imperative; and that the quality that comes from that testing is everyone's responsibility from this point forward. Our competition and our marketplace are outpacing us. Let's do a go 'round. What can we list as takeaways from this meeting?"

She paused. She counted silently, as she had been told long ago to do. "Count to six," someone had once told her, "or no one will think of anything to say, and the meeting will just have been a speech."

"Three . . . four . . . five . . . six."

Jessica Blantyre spoke up first. She was the youngest member of the team and was eager to involve herself in the discourse.

"A new form of testing is imperative if we are to stay competitive, profitable, and safe."

"Great," said Leigh. Owen was next.

"I guess we need to stop thinking about testing as an event. It's not something to be done at a specific point. It must be done by everyone all the time, even before code is developed. It has become more imperative because it has become more visible. The release cadence has greatly accelerated. For example, when we were working to a 12 to 18 month cycle we budgeted a certain number of weeks into the calendar for testing, but now with more frequent releases, testing is becoming one of the first things kicked off the checklist. There's conflicting needs of more frequent releases with less time to test."

Arthur coughed loudly. Was that a comment or just coincidence? Leigh dismissed it quickly from her thoughts. Giles spoke up.

"We have to think about shifting testing further left. We need defects and problems to be nipped in the bud, not caught—or even missed—as the product comes out the chute into the customer's lap.

Dinesh spoke next: "dotPlane is all over us. I think that says it all."

She closed down the PowerPoint projection. "As promised," she said, "this meeting is now done. Send your questions or comments to me by 5:00 today. As of now, quality is everyone's responsibility, and your assignment, class, is to send back to me every bottleneck and challenge that you currently see, have seen, or anticipate in getting our products to market in perfect shape. And help yourself to the leftovers!"

Leigh and Alicia watched as everyone scooped up their laptops and the leftovers. Jessica and Dinesh, Leigh noted, each took an empty paper bag that had the Veneto Burrito logo on it. They each flattened out their respective bag and placed it on the keyboard of their laptops, which they then closed, and left the room.

Leigh watched them leave. She couldn't shake the feeling she had missed something big here. It was—

Her train of thought was derailed at that moment by Alicia, who was also taking a Veneto Burrito paper bag as a souvenir.

"I'm glad you brought DevOps out in the open back there," she said. "I want to add it to your official responsibilities going forward. We have a lot of changes to make. Given that QA already has its hands all over the whole IT organization as a shared service, you already know all the teams, the applications, the nuts and bolts of it all. What you will need to do is to press upon these folks the need to break out of their waterfall complacency. In addition, I think you should start a knowledge base—a wiki of some sort—to keep this all under control. And please link me to it."

She locked eyes with Leigh with unusual seriousness.

"There is an urgent need for this," Alicia said. "These folks don't know what's coming, but I can see a drama unfolding that we will all get swept up in. Think of Helen Humphries as the first act." She tucked the Veneto Burrito bag into her computer case and walked out.

CHAPTER

3

Challenges to Achieving Quality

Leigh was back in the seat of her beloved Porsche, which was parked in one of her most treasured places in all the world—one of the best and most exclusive places of all. She was parked on the shoulder of the service road that ringed Logan Airport, right at the foot of runway 4R/22L.

Very few people were allowed here anymore. Enhanced security measures post-9/11 had put a stop to the fun for many a plane spotter. Before that time, some airport service roads used to have hot dog vendors to feed the dozens of plane lovers, but now these roads were strictly off-limits to anyone not affiliated with the airport in some way. Leigh's Renway credentials gave her access to a limited amount of space inside the fence. She liked to watch the hawk and drone technicians who worked tirelessly to keep birds from straying too close to the runways, and she watched the security teams as they watched her.

Every few minutes, a plane would come screaming in, barely 100 feet above her, but still travelling at between 100 and 150 miles an hour. To hear and feel the enormity of these planes never ceased to be a thrill. The experience had been magnified one-hundred-fold when she had brought her father, Warren,

J. Scheaffer et al., *The Kitty Hawk Venture,* https://doi.org/10.1007/978-1-4842-3661-1_3

a retired aerospace engineer, to see two extremely rare occurrences. Their first had been the takeoff from Logan of the Antonov 225, the world's largest civilian transport plane. It was big, heavy, and brutally beautiful. It always seemed it would run out of runway before its front wheels ever left the ground, but regardless, it reliably heaved its massive self and its cargo into the air just in time, thanks to the simple rule of physics that allowed all planes to fly: lift.

The second thrill had been bringing her father to watch the landing of the Concorde during its 2003 farewell world tour. She was 25 and a student at that time. He drove, and they had watched it from the parking lot of a coffee shop on Bayswater Street, along with hundreds of other people. Warren had worked as an engineer for a company that made the turbofans and turboshafts for jet engines and helicopters. Her passion both for flight and for quality engineering had come from him; however, she hoped her career would not end as drastically as his had, as a victim of across-the-board layoffs in a sudden downsizing.

At present, she was sitting in the passenger seat with her laptop open to a.) the Slack channel that she had assigned to her team at yesterday's Veneto Burrito meeting; b.) her personal Twitter feed; c.) her email; and d.) a flight-tracking app that showed the real-time movements of every large civilian plane in the world. Currently, she was watching the cluster of tiny airplane icons swarming around Logan Airport like bees approaching a hive. Her car's Bluetooth was hooked up to the music app on her mobile phone, which was playing Tom Petty's "Learning to Fly." She thought about the lyrics as she watched the planes' landings and takeoffs and contemplated her own continuous testing journey that was beginning to take flight.

Her phone rang and interrupted the moment. It was Alex Ng calling precisely on schedule. Alex was a friend who had sprung out of one of life's chance encounters—they had met as side-by-side seatmates on a flight back from an IT conference in Denver two years ago. They had both attended the conference but had not met there. It took the boredom of a defective seat-back TV screen to spark the conversation that led to their friendship. He had been involved in airline-related software development for years, and the two had taken to keeping in touch as mutual mentors. He seemed to know a lot of people and a lot of interesting stuff about the industry. Leigh also liked the fact that he always sounded very happy and content with his work. It was refreshing to experience that.

An inbound Emirates 777 roared overhead, temporarily blotting out the sound of Alex's voice. Leigh was always careful to mute her own phone when working from here.

"When I was leading the Quality Management Office—the Testing Center of Excellence for Fugue Air in Ireland," Alex said, picking up the threads of the conversation they had been having a few days back, "the biggest challenge we had was that our departments operated completely in silos. All of

them. Reservations, airport operations, flight operations, maintenance and engineering, cargo, customer loyalty, HR, and so on. They all had managing directors reporting to the CIO. They all had their own little worlds. I remember the typical output of the testing team in the reservations department was a record locator. Be it through the call center application or the dot-com, a reservation was created. Meanwhile, the airport operations testing team needed a record locator as a prerequisite for their test execution to start. After all, they needed a reservation so they could check a passenger in. What they used to do was to go into the reservation system and generate a new set of record locators based on their testing objectives. That was very time-consuming, and it wasn't even part of their test case; it was a prerequisite for their tests.

"So, I—my team and I—we said, 'Let's work together!' We told the department managers we would like to align the airport operations testing prerequisites to the record locators the reservations department created as part of their own testing. That would cut a lot of effort from the airport operations team, as they could start their testing cycle by leveraging test data newly created by the reservations department. But you know what they did? They pushed back! They said no! So, because I wanted to do the right thing, I had to pick some of my best guys and architect a way to enable that vision to work without involving the departments' management. That was one of my challenges—having a testing Center of Enablement, without actually calling it one, that would be able to leverage the synergies between those departments.

"We knew there were heroes out there—team members who were trying to do things in the right way to ensure quality was assessed somewhere in the lifecycle. So, you know what we did? We just got together during lunch and talked about how the business units were going. After a few months, we were able to create a cross-departmental framework. We used brown-bag sessions. We would bring a projector to the cafeteria at lunch and invite people to observe how we tested across departments. How each team was able to leverage each other's data and testing assets, including automated tests. Then, we started becoming the Center of Enablement, where people would come and walk up and ask for access. We were generating interest from the bottom up. From that, we finally got some support and funding to make the CoE a formal entity that enabled testing teams across departments."

Leigh's computer beeped. Input from Dinesh.

"Alex, do you have a couple more minutes?" Leigh asked, "I'm getting some input from Dinesh. At yesterday's meeting I asked the team to send me a list of barriers that they feel stand in the way of achieving quality from their perspective. He's sending me some stuff, and I would love your feedback."

A UPS Boeing 767 filled her windshield as it descended incredibly quickly to the runway. The landing gear tires coughed up a puff of blue smoke as they contacted the tarmac.

"Deadlines," she said. "The first barrier to quality. Squeezing capabilities into a release."

"OK," Alex said, "I can see that being an issue. But if you do smaller and more frequent releases, you should be able to help with that."

"Roles," she continued. "Inadequate delineation and identification of roles."

"Been there," he said.

"The next one is metrics." She quickly muted her phone. A Lufthansa 747 passed overhead. She thought for a moment of how yesterday's meeting in the hangar had made her feel like she was standing in front of a shark. When you're under a 747 heavy, it's like a blue whale power-diving over the top of you.

"Metrics," she repeated. "The quote next to it reads, 'Management has no idea of what quality is. How do we know we are doing a good job? If no metrics, then we have no idea what quality is.'"

"OK, that's three," said Alex. "Any more coming?"

"He has four more," said Leigh. "I knew these would be on here. The next is testing."

"Naturally," said Alex.

"Then, there's assessment."

"Next?" he asked.

"Legacy tools. And QA."

"And finally?" Alex asked, knowing precisely what it would be.

"Change management."

"Of course," said Alex. "Well, let me know what I can do to help. I know this is all confidential inside stuff, but these are problems everyone has faced, and not just in this decade. These problems were around in one form or another long before personal computers were on the scene. But, as you can see, except for the legacy tools this is not so much about software as it is about culture. But you know that already. It's a matter of what risks management is willing to accept and what their managers are measuring their performance against.

"In most cases, unless there is a compelling business reason being pushed down IT's throat, managers will hardly have the motivation to change their organization's culture. In your case, as it was with mine, changing that culture means looking at software quality as something you build in from the beginning of the development lifecycle. And in your case, the driver for that change is so that dotPlane stops eating Renway's lunch."

There was silence, both in the conversation and in the air above the car. Both were fleeting.

"I have to be on a conference call that started five minutes ago," Alex said. "Let me know when we can talk again."

His voice was drowned out by a Renway Air 737 dropping out of the sky and gliding perfectly along the runway toward the taxiway, apron, and gate, where the ever-efficient ground crew would guide it in with those wonderful orange wands. Leigh made a mental note to arrange to spend a day on the apron with them. She could call it QA research. The truth was, she really just wanted to give it a try. Or at least be allowed to stand out there and watch one of those beautiful beasts come to rest with such precision and grace.

Leigh reviewed Dinesh's message. There was a good amount of detail there. She quickly copied and pasted it into her wiki.

She checked her email. Fourteen new ones:

- Two emails to Alicia, cc'd to her, sent an hour apart from JP, a project manager, escalating to her that testing is behind schedule on the dot-com site for a new feature that allows passengers to multiply their miles for an additional fee. In a red, bold font, he had written, "This will cause us to miss our release date. We need to narrow down the number of tests so they make the date, while ensuring quality by keeping test coverage optimal." Leigh marked this one as unread in order to return to it later. It was a high-profile release that would directly impact a new revenue-generating channel.
- A budget meeting request for next Tuesday. No, not a request exactly; one of those annoying email invites that basically confirms you are coming regardless and makes it your prerogative to decline.
- Something with the subject line "Kitty Hawk"—no idea what that is—just "Kitty Hawk" in the subject line and the words "Eyes only. Green light" in the body, no signature.
- An invite to a Dubai conference
- A "howdy y'all" from Marnie, who was away on maternity leave
- Seven annoying emails from software vendors trying to sell her tools for test automation, test management, test data management, pipeline orchestration, and mobile device testing
- A recall request for the Kitty Hawk email

She closed her laptop, gunned the 911's engine, and headed once again toward the highway and the office. But as she was doing so, she changed her mind a little. There was something in what Alex had said—"This is not so much about software as it is about culture"—that made her realize there was someone else she needed to see first. She merged, cut across sharply to the fast lane, and punched 1 on her console speed dial.

* * *

She pulled into the driveway of a quaint New England–style house in a leafy suburb of Thornbury, itself a smallish town not far from Collingwood. It was neat but not ostentatious, well cared for, and, yes, her childhood home. Her father, Warren, was busy sweeping the first of the fall leaves from the driveway. He waved and stepped theatrically aside to allow the car to pull in.

"Dad," she said, "we bought you a leaf blower for that. Don't go exerting yourself just to keep the driveway clear."

"Pah!" said Warren, brushing off her scolding with a wave of his hand. "Just rev your engine a coupla times. That'll get rid of them all." He rested the broom carefully against the wall. "C'mon in, Bru," he said. "You want some coffee? Or something stronger?" He had called her Bru all her life. It was short for Brutus, which was what the Boston General Hospital Labor & Delivery staff had nicknamed her on the day of her birth, coming in at a stocky nine and a half pounds.

"I only have a few minutes, actually, Dad," she said as she followed him into the house. "Got to get back to the office." She looked at his hands as he pushed a coffee pod into the machine. Nice and steady, she thought. No sign of trembling or arthritis. She looked at the propeller tattoo on his forearm, the blue ink now softened and diffused by age. It was a symbol of his love of all things aeronautical, especially the confident purr of a piston-driven Spitfire from World War II. Warren had been born in 1948, but *his* father, Wallace Freemark, had worked for the Met Office—the British weather service—assisting in the preparation of flight plans for the RAF. The sound of a perfectly tuned Vickers or Mustang engine was viscerally pleasing. Leigh and Warren had been to many airshows together just for that experience.

Warren poured the coffee. "When you were with Mainland Technologies did you have problems getting people to listen to you?" she asked.

"Why, is that happening to you?" he asked with instinctive paternal concern.

"Not yet, at least not that much," replied Leigh, "but soon, I'm sure."

"Pah!" Warren exclaimed again. "All the time. I was a senior test engineer. That meant I had to keep telling everyone around me, above me, and below me things they did not want to hear. That there were microcracks in the turbine blades and that we couldn't make shipment. I'm sure that has a lot to

do with them 'setting me free' so early. They could blame it on the economy all they liked, but no one likes it, Bru. They don't like being told 'no,' and they don't like having to change."

Leigh thought for a moment about that day she had been allowed to go to work with her dad—Take Your Kid to Work Day. She had been 15. She had gazed with absolute wonder as they drove their golf cart slowly through the machine shops, where huge green lathes carved precise metal blades and other things, their hot points of contact being constantly drenched by a stream of milky liquid. It was like an industrial version of Disneyland. Soon after came the department where her dad worked. The metal components were dipped in vats of bright green goo that glowed in the black light and showed where the defects were. There were x-ray rooms, chemistry labs, and a whole lot of grey and green.

"Those were the days before the internet and the computer, you understand," said Warren. "We had computers, but they were up in back rooms. We had to do stuff longhand."

Leigh smiled to herself. Dad was playing the old man again.

"The problem was that my definition of quality was always too stringent for many in the executive and sales departments to bear. When they had already promised delivery of goods by a certain date, they never wanted to hear 'no' from some engineer on the line. Instead, they got legal to insure the shipments against defect lawsuits—even wrongful death. They had the math figured out. Lawsuits back then were much cheaper than retooling the production line to build a better product from the get-go."

He smiled.

"Let me tell you this, Bru," he said, "people don't change. Their good and bad points don't change much no matter what decade we're in. If you're starting to make changes with your hotshot computer people up there in Collingwood, be prepared for pushback. Your goal might be fixing your product, but your target has to be the people. Look after them, Bru."

"I will, Dad," she said, and finished her coffee. She gave him a quick kiss on the cheek. They walked back out to the driveway, and she climbed into her Porsche. She lowered her window to say goodbye.

As she backed out, Warren pointed at the front of her car and shouted after her, "You ought to check under your hood sometime. I heard they don't always give you an engine!"

"Don't worry! I've got a spare one in the trunk!" she shouted back. It was a very old joke. She waved and turned left toward Collingwood.

* * *

Back at the Century building, she took the elevator to the 17th floor, where the executive offices were. People called it the "oak door floor" on account of the executive trimmings—nice lighting, nice décor, fancy artwork, all designed to impress customers and soothe the executive mind. She occasionally ventured up there to seek out an unused meeting room to get some work done without interruptions.

She found a small room whose door was open, and which seemed ideal. That is, until she heard a familiar voice emanating from it. A British accent. Giles. She edged her way in and found Owen and Giles watching a soccer match on the projection screen.

"Working hard, gentlemen?" she asked. "On the executive floor, no less?"

"As a matter of fact, yes," said Giles. "This is work, and bloody hard work, I might add."

"Do tell," she replied.

"Well, this on the screen is just a UK match, but it's got me thinking. The World Cup. Why are we not flying there?"

"Oh, that's easy!" said Leigh with mock sarcasm. "It's in *Russia*. We don't fly there."

"No, but dotPlane does," replied Giles.

"What? No, they don't," said Leigh.

"Have a look," added Owen. He alt-tabbed away from the soccer game to a web page.

On the screen was an ad for dotPlane. "Moscow or Sochi. We're there, you're there. Da!"

"Exactly," said Owen. "They're already in bed with a transatlantic. A long haul. And they've got alliances with British and Spanish carriers, and their branding is all over it."

Giles spoke up. "OK, so the main thing about this is a lot more people than normal will be travelling to Russia, and primarily to the eleven host cities. We should have been on this a long time ago. Distribution is going to be different, not just volume. Whoever gets a lock on that can effectively freeze out the competition. So late may mean never in this case."

Leigh cut in, "I don't think senior management ever considered doing an international route. We don't have the alliances. I don't think this was ever on our radar as a company. Besides, it's probably too late."

"Yes," said Giles, "you're right. But if dotPlane can get in on it, then we should be able to. This is about taking a large number of people across to London and further afield to Moscow thanks to whatever partnerships can be set up.

I think someone on this floor should get a team on this and see how we can get involved. This is one big baby, and somehow our competition is already onto it. The exposure alone is priceless; this is where the money is. Maybe I should give up software and work for Dinesh in marketing.

"Alright," said Leigh. "I will be talking to Alicia tomorrow morning. She has the ears of Bob and Helen Humphries. Do you mind if I sit here and take care of a few things? I just need to cross a few things off a big, long list."

"Help yourself," Owen said, as he and Giles returned their attention to the soccer game.

She opened her laptop and alt-tabbed to her wiki page.

Leigh's Wiki: Managing the Challenges to Achieving Quality

She entered a collection of bullet headings:

- Roles
- Silos
- Deadlines
- Metrics
- Testing
- Assessment
- Legacy Tools
- Quality Assurance
- Change Management

She started copy/pasting text from Dinesh's message. Then, she stopped and looked at the two geniuses watching soccer. They were one level senior to her, but what the hell, she thought. Let's get our own ball rolling here.

"We have a problem with our testing culture," she said to Owen and Giles as they watched the game in the name of research. "We need to change the way we test, making it continuous from the very start. This is more than just shift left, and it is way more than automation. To make this work, we have to see what the bottlenecks and roadblocks are."

Giles wheeled his chair around to face her. "Let me caution you a little on that," he said. "This is where so many companies get off on the wrong foot, as it were. It's all well and good to go charging at the bottlenecks as a silver-bullet solution, mixed metaphor notwithstanding." He looked back to check his research on the screen. Arsenal 2-2 against Chelsea.

"Granted," he continued, "identifying these bottlenecks and removing them is how you get a system that does what it should, and does it with better flow,

shorter lead times, and in a generally more productive way. There are aspects of our old-school testing that are in that boat for sure. But there are also aspects of our testing that are simply insufficient and not fit for that purpose. What I'm saying is, simply removing every bottleneck and achieving perfectly laminar flow won't necessarily solve the problems we identified in yesterday's meeting."

There was a pause.

"First, you have to think about the tools," he started. Leigh looked at the *Legacy Tools* bullet on her wiki page. "Testing processes need to be automated to improve speed and increase test coverage. But once the processes are improving, the traditional testing tools no longer support that fully automated form of testing. They were not built for that; they're from a different time. That's why open source has been gaining so much traction—it's more flexible, extensible, it's easier to experiment and fail fast, then scale. And developers love it, so it's easier to achieve adoption by developers as we try to shift testing left. Legacy tools are an impediment for testers to test at the speed of agile. You need tools that won't get in the way of speed and quality. You need tools that enable quality to be built into the process and not 'just' test the application, as in the past. I read somewhere—yesterday was it?—63 percent of mature adopters of agile and DevOps still report testing as a cause of "bottlenecks in their software development lifecycle."

"The main thing driving open source into the testing space is the expansion of testing responsibilities from testers to developers, not generic process improvement *per se*. Unlike testers, developers will gravitate to creating and consuming open source solutions both because they can and because they are not funded to buy tools anyway even if good ones were available. You have to question whether this is relevant to us in our current situation, whether it is germane to the issue of instituting a cultural change.

"Don't overlook the role of the app. Application lifecycles are changing. The methodology to develop apps has changed. Go to any development conference and you'll hear the same things consistently. It's not about building what we as a company think our customers will want; it's really about building a working concept, a minimum viable product—an MVP—so that we can get it in the hands of our customers to validate our business hypothesis. If that's proven, then we stay the course and double down on making that MVP more robust, and we continue adding features as a hypothesis that must be proven before committing considerable resources to building it at scale.

"But then again," he added, "everyone in our IT organization is pretty traditional. They never go to those conferences. They've been doing the same thing the same way for decades. That's too bad for them. They're missing out on pub crawls and tons of bacon. And too bad for us, because there's no waterfall in mobile or IoT app development.

"This requires a new way of testing, as there's a lot more that can go wrong. We'd be releasing much more frequently and in much smaller chunks. By the way, these chunks must be merged more frequently and tested for proper integration across multiple technology stacks. And that has to happen much faster. Given how we do things here today, it sounds impossible, right?

"Modern companies use A/B tests, canary builds, blue-green deployments, and other techniques to support this new way of developing and testing applications. Is that something we're ready for? Probably not. But that's what we should be shooting for. It'll be a journey to get better, but at least we'd be aiming in the right direction so that we can support our business goals. Which, at the moment, all seem to revolve around beating dotPlane to the World Cup."

Giles grabbed the remote and turned off the projector. The soccer research had returned a 2-2 draw. He looked at the comfortable office chairs around the table. He stood up and pulled one away from the table and drove it closer to two other office chairs.

"Ok, then, here's a great example," he said. "Let's look at seats. The individual capabilities of our app to do a seat change. This is the customer interacting directly with us. 'Is my seat booked? Is the seat next to me open?' That piece is OK. The bottleneck comes when systems get integrated together. We have to send and receive info back and forth to places like Expedia or Awayz. These integration points are just like soldering points. They're most likely to break. Not so much because of the technology, but because of the interaction.

"We can tell Expedia how many seats are available, but the customer wants to know *which* seats are available. And what about performance? Can we do this with 40 or 4,000 concurrent users? What about 40,000 rabid football fans all looking to fly to Moscow in June?"

Leigh interjected, "This is a requirements issue, right? This comes from having too many business analysts with their hands on the throttle. They can introduce issues, but they don't think holistically. They say 'Wouldn't it be nice if the app could do X,' but they don't specify that they do not want the app to do Y or Z. No one tells the developers about any negative constraints from a requirements perspective."

Giles replied, "Yes, and I would add to that a couple of points to help you ponder how this new way of developing and testing software would impact other critical aspects of application quality. First, you ask, how good is my testing? What is my functional coverage, not just my code coverage? What are my acceptance criteria? Am I actually testing for them effectively? Many testers don't know.

"The word you're looking for here is *insight*. Understanding how good the testing is when you're doing it and then trying to structure that in a way that is cost effective to run and maintain."

"OK," said Leigh. "So, what's the second point?"

"How quickly can I execute them—the tests?" Giles said, without skipping a beat. "What and when? I get into all types of tools and process domains that will vary from application to application. For example, the vast majority of regression testing—70 percent—is manual, going through fixed test environments. It's also being executed in serial fashion. We can't just magically automate that and declare victory. We need to be smarter and get as much of the testing as possible out of being dependent on these scarce fixed environments, and get it happening in parallel, leveraging ephemeral on-demand environments, complete with test data and virtual services, to resolve key dependencies.

"The bottom line again is that we need to get to a point where testing can become parallelized, and then shift left and get it all happening continuously."

"So, in summary?" asked Leigh.

"Move from fixed to ephemeral," Giles said, then corrected himself. "Actually, from serial in fixed environments to parallel in ephemeral environments. Not full-stack, but on-demand ephemeral environments that are just enough to get your testing done."

Owen broke his silence. "Don't forget security. Is that on your list? There's a real lack of robust testing in general. From where I sit in Ops, security always bites us in the butt when the pre-production guys don't do proper security testing before deploying the system in production. We have to treat security as a first-class citizen. These have to become specialized testing practices that get incorporated into our continuous testing process."

Leigh added another bullet to the wiki page, titled *Security*.

She waited for Owen to say more, and he actually shouted. "Performance too! Ops is always burned with performance-related issues! Once the application is in production and my team has to maintain it, we usually find out that the application is not performing under the actual user traffic. The quick fix for this is to increase horsepower, which means more hardware. In a cloud-based infrastructure this would be less of a problem for us. But obviously our infrastructure is not in the cloud, so we need to increase physical hardware in our datacenters in order for a software application to perform. That's a lot of money, and it's not sustainable. I'm getting beat up by our CIO monthly because I've been unable to lower our hardware and hosting costs. How can I? If I do it, users won't be able to use our applications, especially the revenue-generating ones. We have to ensure the application performance is tested prior to production, starting all the way to the left, like we've been talking about with functional testing. And we must continue monitoring performance in production. Again, we need the right tools and processes for that."

And with that his phone buzzed. He checked it. "Gotta go," he said, and left quickly.

Giles, too, had tuned out. He was still there, sitting at the table, but he had absorbed himself in whatever was on his own laptop screen.

Leigh pushed her chair away from the table a little and stretched. Her email inbox chimed. A message from Adam. The subject read, "Can you still make it for our semi? 5:30 today at Shakers?" Adam was the one true friend she had made during grad school. They had studied together, wrestled complex assignments together, and eaten buckets of ramen noodles together. Upon graduation, they had promised each other to meet up twice a year to reflect, reconnect, and support each other's challenges and successes.

"5:30?" She checked her watch. It was just coming up to 4:00. Shakers was the pub on the ground floor, so there would be no trouble getting there, save for having to deal with the urgent messages from JP. And the fact that she hated going to after-work pubs. She replied "Of course!" and hit Send.

She looked across to where Giles was still working, but he had his headphones on and was typing furiously. She tapped her chin with her pen. For the second time in two days, she couldn't help feeling that there was something more here than simply a coding issue. There was something else she was not seeing.

She let her mind wander and looked around the room. Such a dull place, especially for the oak door floor. Just office furniture, dry erase boards, and starfish-shaped conference phones. Not a room for visitors, obviously. It was an inside room, so there were no windows. She looked at the flipchart next to the screen. The front page of the big paper pad was blank, but she could make out some writing on the sheet behind it. This was quite a normal occurrence—people often prepped flipchart notes for the next day's meetings. But the faint lettering caught her eye. She was sure she could see the words "Kitty Hawk." Just like this morning's email.

She looked over at Giles, but he was engrossed in his screen, typing quickly, and was facing the other way.

She edged over to the flipchart and carefully lifted the blank sheet. The title of the sheet below did indeed say "Kitty Hawk." There were many bullets and much bad handwriting. She scanned it quickly, checking once more that Giles was not watching. Words like "shareholders y/n," "prospectus," "first round," and some large dollar figures leapt out from the scrawl. The word "MidWest" was there with a large arrow pointing to the word "Renway," which had been circled numerous times.

She looked at it again.

She replaced the flipchart's cover sheet and then sat back down and alt-tabbed across to her email inbox. She looked at the "Kitty Hawk" email still sitting there in her inbox, and at its clumsy recall notice. The email had come from the office of Lester Greene, Renway's head legal counsel. Just yesterday, Legal had sent a note to Leigh and 22 other managers regarding updates to GDPR

compliance projects. The Kitty Hawk distribution list was the exact same names. It had been sent out to the wrong distribution list. "Eyes only. Green light." This had obviously been intended for the senior executives.

Leigh put two and two together. Renway was buying MidWest. An airline that was already failing.

* * *

Shakers was at the far end of the basement retail complex that Renway's building shared with the one next door. There was a small food court down there, some stores, and a gym. The pub was at the far end. It was 5:25. She was five minutes early. This meant she could choose to sit facing the door. There was no special reason why she liked to do this; it was just something she had learned from watching her favorite movie, *The Godfather*, about a million times. That, and 'always order the cannoli.'

Adam Dobrovodsky must have liked the same movies because he, too, arrived at 5:25. So, with the power play evened out to a draw, they located a table together and sat down. Leigh still succeeded in getting the seat facing the door.

After they had exchanged a little small talk, Leigh told him about her investigations into continuous testing.

"That's a tough road to travel," Adam replied.

"At the moment, I'm trying to pull together a list of the obstacles," Leigh said. "We need to build quality into the SDLC, but there are bottlenecks everywhere. I mean, I know all about this on a project-by-project level, but what I need to do is instill this into the corporate culture. Change management is so much a human thing, which is more difficult than writing code."

"Yeah, I remember," Adam said. "Psych was never your favorite subject back in the day, was it? Dr. Padgett wasn't exactly your favorite prof."

"I always preferred the purity of numbers, to be honest, D," she replied.

Adam smiled. A woman who prized logic and numbers above all, but who let fate alone draw her into the world of airplanes. By contrast, Adam—D to his friends, the first letter of his last name—was much more a people person but had planned his way into the IT world of banking and finance with intricate precision.

"Well, for what it's worth," he started, "at Consolidex, we realized we needed both top-down and bottom-up change initiatives. It always starts with great leadership. We created an experiment specifically to build a continuous testing culture. We set up a monthly audience with testers and developers and focused on building energy and building a plan together. We had to make them part of the solution by involving them in the initiative rather than simply dropping it in their laps one day."

Leigh was taking notes on her phone. A waiter stopped by, and they both ordered beers. D also ordered a shot of Stoli. "For my mother's memory," he said.

"Oh, no!" Leigh exclaimed. "Has she passed away?"

"Certainly not," said D, "she drinks Stoli because it helps her remember. Especially the gossip."

They laughed. The drinks arrived. They toasted the past, the future, D's mother, and Leigh's dad.

"This is evolution," he said, "peeling the onion. At Washington Financial—my previous employer—we built a team. We worked with them to reduce regression testing time—which sometimes took five days—by moving to acceptance level and API testing. Regression almost disappeared. It was still there, but it was just more API testing than UI testing. Regression is no longer just random tests, but rather targeted tests based on change. There were fewer production defects, a shortened time to market, and smaller production teams. You still must have unit tests. You have to have thresholds set up. But you have to break testing into components."

Adam paused and watched her take down all these notes. "You don't have to write this stuff down, Leigh," he said. "Just take it in. This isn't about code; it's about people. Even the VP level cannot make the full transformation work, but they need to influence. Look at shifting left. We need developers onboard. They'll have to change the way they work, they'll have to welcome testers into their day-to-day routine, they'll have to be friends and not enemies. For the shift left to work, you need testers to learn new skills so that developers can see them as people."

He laughed. "I mean, people who can help the team succeed. That is, release quality software at speed. If testers continue to be seen as people who will point out flaws in what a developer wrote, it'll be hard for the shift left to actually happen. By working together, both personas can start testing earlier in the lifecycle, at the unit level—even before that, at requirements definition time. And you achieve that by letting developers and testers, who ideally would have similar engineering skills, experiment and try new things. Give them the chance to fail . . . and learn from it.

"And always remember the biggest elephant in the room, Leigh: there will be a bunch of people who will worry that they will lose their job if the shift left happens, or if other types of new and better testing happen. They have to be shown that they can—and should—be able to do more. It's a matter of saying to them honestly, 'Your job is not going to go away.' The role of testing must change to become a new form of CoE: a center of enablement. Their role is not diminished. It is becoming that of a service broker."

They talked for a little while longer. 6:30 became 7:00. The composition of the bar had changed from the "let's meet for a drink after work" crowd to the smaller "I don't want to go home" crowd that made her feel distinctly uncomfortable.

"Let's not wait another six months," Adam said. "I'm doing an OpenSpace on Thursday. Come on by. I'll send you an invite."

"Thanks," said Leigh, "I'll try to make it." They embraced, and Adam left. She paid the check and sat motionless for a moment or two, gazing into the middle distance of the bar. She contemplated things for a moment and then came to a decision. She typed an email into her phone, not sure if the recipient would be in San Diego or Germany. "Oh well," she thought, "here goes nothing."

Seconds after pressing Send, the phone buzzed. It was a text message from Steve Fibonacci. "Got intel from the sneakernet. I got some QA stuff for you. Call me."

* * *

"The sneakernet? What the hell is that now? A new web app?" Leigh asked through her speakerphone. It was only yesterday that she had reconnected with her old boss, Steve Fibonacci, in the Veneto Burrito–catered meeting, but for whatever reason, and completely against her better judgment, she now felt free to talk with him as a peer. His style was just as comfortable, no pretense, especially for a company owner who should have better things to do.

"No," replied Steve, "it means I used my shoes and walked over and talked to someone in person. More people should try doing it. It's friendlier, and someone usually has snacks. You won't get that with email! Anyway, I was just following up from our chat yesterday. I heard you are compiling a list of challenges to achieving quality, and I thought I would send you my notes. I'm recording the call, so you don't have to memorize this or anything. Just have it transcribed later. OK?"

Leigh wondered who he could have talked to. It must have been someone at Renway. Who's to say where he walked to after he finished setting up the food. And he probably didn't need the snacks. People like him just seem to make connections that continue to grow in value. His surname truly suited him.

"OK," said Leigh.

"OK," Steve began, "point number one. I feel like your side—the QA side—has been acting like order takers for the past I-don't-know-how-many years. When we think about quality or an issue in production, we'll say, 'Why didn't we find this in the testing phase?' and then QA will say, 'We received the code late and tested what we could.' You should be asking, 'Why aren't QA and Dev working together throughout the development process? Why is

there a waiting period?' But it seems like QAs everywhere have some sort of inferiority complex. You're introverts."

"I'm not an introvert," Leigh said in mock objection.

"You like numbers more than people?"

"Yes."

"Science more than socializing?"

"Yes."

"You named your dog after a scientist?"

"Copernicus. Yes."

"Magic 8-ball says, 'All signs point to introvert.'"

"OK. Anyway . . ." said Leigh with mock sarcasm.

"Anyway, QA can't be a second-class citizen anymore. If your applications are being designed for an outside consumer—like your customers who want to check their flight status—now you have the whole world interfaced with your IT. That mentality of second-class citizen is upside down. You are better off with a suite that offers less functionality and never fails than a Cadillac that never works properly.

"Point number two," Steve continued, "QA has to involve senior leadership. You gotta ask who's minding the store. Once senior leadership starts to focus on quality and not just delivery, that's where things start to happen."

He continued. "Point number three: requirements. There are always gonna be reasons why quality is failing. If developers don't fully understand what they should build—if they don't have proper requirements that are unambiguous, complete, and testable—this needs to be fixed. Otherwise, how would someone expect them to build the right things? You need the folks that write the requirements to be open and communicative and integrated into the application team, not hidden at the top of a silo somewhere. Remember, Leigh, there's a substantial difference between building things right and building the right thing. Remember the Stonehenge sequence in *Spinal Tap*?"

"Yes," said Leigh. "I should show that to my team. It's on YouTube, certainly."

Steve continued. "Exactly. So, we can have code that is perfectly written, but is not doing what it is supposed to do. That's where requirements come in; they are the guiding principles for the developer to write their code. And that's the primary measure for gauging application quality. Is the application doing what it was intended to do based on requirements? In other words, did developers build the right thing? Or did they just build it right? You still there, Leigh?"

"Yes," said Leigh, "drivin' and listenin' like a good introvert."

"Good," said Steve, overlooking the sarcasm. "Sixty-four percent of defects occur in the requirements phase; of course, you know that, right?" Leigh nodded, forgetting she was on speakerphone.

"Yes," she said. She had mentioned that exact number in the Monday meeting.

"Sixty-four percent," he repeated. "That's almost two-thirds."

"Thank you," said Leigh, "my master's is in mathematics and economics. They taught us fractions on the first day. That's the two on top of the three."

"Correct," said Steve, deflecting the sarcasm yet again. "Continuous testing is going to fix that, but it's a culture challenge. The people who have their fiefdoms—they want to keep control. That's a challenge. No tool in the world is going to help you if you can't overcome those cultural issues. Has anyone broached legacy tools with you yet?"

"Not fully," Leigh replied.

"OK, well, point number four, challenges with tools. Legacy tools. They might have sufficed for waterfall, but they don't work well in continuous testing. Developers refuse to use tools for non-coders, like traditional testers. They don't want to use anything that is overly complicated or requires a lot of clicks. Often, they can write something better by themselves fairly quickly. Hence, the rise of homegrown and open source tools for some things in the developer-implemented testing domain. Most legacy tools were not designed for the developer's skillset or their workflow. Developers don't want to be stuck in something proprietary. It will not help their career. It's a human thing. And it's a multiple UI thing. You got room for one more?"

"Yeah. Fire away," said Leigh.

"OK, point number five: the feedback loop. It used to be we were able to wait through an 18-month release cycle. We could poll the customer when we had time. But now, when we're releasing every month, we have to include the customer perspective in the development process, even at that fast pace. You need proper feedback loops. You need to get those tools set up as well. This is not just something for the Ops guys. This needs to be a joint endeavor by development, testing, and Ops."

"Anyway," Steve concluded, "those are my five points. I'm sure they'll make fine bullets for one of your 'Death by PowerPoint' slides."

Leigh thanked him and told him it would be a wiki page.

"That's better," said Steve, "and remember, 'Point. Click. Burrito.' That's a mindset that will take you far."

Steve clicked off the call.

The Porsche's tail lights disappeared into the fading light of dusk.

CHAPTER

4

What Constitutes Continuous Testing?

Arthur was pacing again, and it was only just after nine on Wednesday morning. He was in Alicia's office, along with Giles, Owen, Leigh, Jessica, and Dinesh. Jessica had bumped into Leigh at Starbucks earlier that morning and, upon learning of the meeting's topic, had asked specifically if she and Dinesh could attend. Jessica knew something big was happening, and she wanted to be part of it.

"It doesn't work to say everyone is responsible," Arthur said, essentially picking up his tirade from where he'd left off on Monday. "When everyone is responsible it means no one actually takes responsibility."

"What do you mean by that?" asked Alicia.

J. Scheaffer et al., *The Kitty Hawk Venture*, https://doi.org/10.1007/978-1-4842-3661-1_4

"Well, I think everyone here has quality already built into their DNA. We have some great people here: developers, testers, everyone. They're very good at taking individual responsibility, and they can also work collectively. But we strongly believe in self-governance by the team and its members, and we certainly don't need interruptions."

"Interruptions?"

"Yes," he repeated, "interruptions. It is far better and more efficient for self-governing agile teams to work with few interruptions to produce high-quality software that is well tested, rather than to have constant interruptions and context switches as reactions to defects coming back from downstream testing."

He paced back and forth. "We still rely on written requirements. But we have an ideological disconnect with testing. We are two distinct types of people. Take a login page, for example. Developers can create a successful login, but the testers will fail it if they do not see a "Login Successful" page pop up, even if the login is successful. I've been at this business a long time. There's a process in place for a reason. I can't see how shaking things up and pushing testing into the whole lifecycle is going to save anybody any time or money. I can't even see how this makes QA any better than it is now."

"Owen?" Alicia asked. Owen was sitting on the edge of his seat, obviously eager to make a point.

"Well, at Monday's meeting—all this talk of shift left and making quality everyone's responsibility—I mean, I see the motivation, but it can also be said that we have these separate jobs for a reason. Testing is not something just anyone can do. For example, I have always wanted to take issue with the concept of testing automation, since we're going down this road. Testing is just as creative as development is. It's just different. It's a craft. It's not something that is just stamped out like some sort of assembly line machine. It requires thought. And it requires a definite skill to be able to identify what needs to be tested and how to go about testing.

"Automation is just the 'how,' which is fine, but with the focus very much on the 'how,' we seem to have overlooked the importance of the 'what,' and I think this demeans the craft. It's not just about writing code. We—testers, that is—we have to think through what needs to be tested.

"Test automation looks great on paper, but it takes away from the experience and the touch that testers bring to the process. The same with performance and load testing. What about the critical thinking you need to establish acceptable performance benchmarks?"

"Duly noted," said Alicia. "Giles?"

"In my previous organization," Giles started, "we didn't have a QA department at all. We found that by not having one, it forced developers to understand

that if stuff comes out of development and has bugs, the people who are going to find the problems are the customers. That forced developers to ensure quality. It makes everybody an engineer, and intentionally blurs the lines for the skill sets.

"Then, we invested in teaching those teams how to write testable code and test it. Our engineers—I call them engineers, not developers or testers—had to build in testability so their code could be easily tested within the sprint, and they had to test new code written by other team members. Nobody just codes. Everybody codes and tests. The team is individually and collectively responsible for engineering and delivering high-quality software."

"Did you have to enforce that?" asked Leigh.

"We put it into our definition of *done*," Giles replied. "In agile, it's collective responsibility, using retrospectives to inspect and adapt the process and its outcomes. Basically, we recognized that your definition of *done* might not be the same as my definition of *done*, so we redefined the criteria that had to be met before anyone in development got to call it done."

"So, this fits with Arthur's point," Leigh suggested.

"Mostly," replied Giles. "It's partly through the definition of *done* that the team members enforce, and it's also partly through light-touch management reviews on some key metrics."

Leigh was typing quickly.

Giles continued, "Basically, my team—at my previous location, and for the past year here, believe it or not—enforced their definition of *done*, which included full-coverage acceptance testing, no failures in the automated regression tests, and no new defects or any severity introduced by the change. This means they had to test that the change worked and that it didn't break anything, and if they found any bugs they had to fix them before they got to call it 'done.' Simple. Then, on a regular basis, there was a managerial review, where each team's representative presented their progress and quality metrics that showed that they were adhering to the basic quality policies that every team was expected to meet for their new story development work."

Arthur sighed loudly. "Spoken like the love child of Mr. Spock and Sheldon Cooper, if such a thing were possible."

"Quite apt," replied Giles, unfazed. "Some of you might be surprised that there already is some modern app development and testing going on at Renway. I have told you all about it at least twice. It might not yet be fully mature, but it's a hundred times better than anything else we have here."

There was a stony silence. Giles was never one to hold back, but his tone contained just enough whimsy and genuine enthusiasm to stop people from actually wanting to kill him.

Alicia said, "Well, I don't think we're going to get rid of QA just now, but we're still facing a major challenge. I have no proof that another reservations snafu won't happen again, which is precisely the request I received from Helen just last night." She turned her monitor around to make it visible to the three of them, including Arthur, who it seemed just couldn't think without pacing around.

"This is the dotPlane website as of this morning," she said. "They're pushing their loyalty app as hard as they can. 'Buy five seats or trips, get the sixth one free. Buy a premium seat—it comes with an iPad for inflight movies. When you deplane, you keep the iPad."

"How can anybody do that?" exclaimed Arthur. "Margins are thin enough already. Who can afford to give away computers like that?"

"It's data," said Leigh. "They're building some incredible profiles of customer preferences and tastes. It's not just about aisle or window seat anymore. They can spin this into hotel reservations, car rentals, Uber preferences, movies. Anything. dotPlane is taking their territory back from the online booking agencies. They're trying to become what Amazon is to retail.

"I can't see it," said Arthur. "It's not possible they have the wherewithal to do that."

"Well," interjected Alicia, "they're doing it regardless. What that means for us is we simply cannot afford these types of upgrade screwups to happen again. We cannot live in a break-fix bubble when our competitors are getting proactive and highly disruptive."

Jessica stood up. She moved across to Alicia's dry-erase board. She addressed Alicia formally. "Ms. Aikens, may I use this?" she asked.

"Of course!" said Alicia, intrigued, "and you can call me Alicia."

"Well, I know I'm new here, and my position perhaps doesn't warrant my being here this morning, and also Dinesh is in marketing, but that's kind of the point. We feel this isn't just an IT issue anymore. Dinesh and I have been going to a bunch of user groups focusing on DevOps. We kept reading about how great Netflix is, and unicorn companies like that. Capital One, also. They don't do hard cutovers. They do it slowly because they continuously test—they have a continuous testing framework. We kept asking each other, why couldn't we do that? It's like an evolution from waterfall to scrum fall to agile to business agility, to what we're discussing here: the merging of agile and DevOps."

Dinesh stood up. "May I?" he asked Jessica, who nodded. "We heard this incredible story—it was a case study directly from commercial aviation. There was a merger between two well-known airlines—they didn't say what country, but one bought out the other, and they had to merge their entire infrastructures. Anyway, one of the airlines had a Testing Center of Excellence

officer, but no one ever went to him to leverage his office's shared services. Anyway, long story short, when the merger happened, both companies tried to keep their silos, and they refused to use his department, until one day he just quit. And that seemed fine until they realized just how far apart the two sides were when it came to software development and testing. They had to reinvent their own wheel by brown-bagging it for a year and rebuilding a knowledge base that he could have helped with from the very start. It set the merged company back years."

"That's a long way from Kitty Hawk," Alicia said.

Leigh almost dropped her coffee cup. Did she just say that? Was it a turn of phrase? Was she comparing today's airline business with the simpler days of aviation? Was it a hint? A mistake? Alicia was looking straight at her, but that didn't mean anything. She felt the pit of her stomach drop, but she kept her composure.

"Why didn't you tell me about this?" she asked Jessica, "about the user groups, I mean."

"Well, the punchline to this story, if you want to call it that," said Jessica, "is that the QMO officer who quit had heard about continuous testing, but he did not believe in it either. He kind of sided with Owen on this. He saw testing as a craft. So, if they had gone to him, they might have gotten it wrong anyway."

"Which kind of shows," Dinesh added, "that this is an ongoing learning process for all of us."

Alicia sat back in her chair, impressed. "So, what do you have for the whiteboard?" she asked.

"OK," said Jessica, "from what we got from the user group, and from Monday's meeting, and from some research Dinesh and I have been doing, here are the things that seem to be emerging:

"First, like I said on Monday, a new form of testing is imperative if we are to stay competitive, profitable, and safe. We have to stop thinking about testing as a singular event, done at a specific point. Testing must be done by everyone all the time, continuously, even before code is developed. And like Giles mentioned, this means shifting further left. We need to move further into a continuous testing mindspace in place of the current release-management process and manual test cases."

Dinesh spoke up. "I know I'm in marketing now, but I actually have a postgraduate IT degree, and I am in constant contact with a lot of my university buddies. They all agree that a separation of duties might have worked well in the past, based on how fast software needed to be built, but now we have to give our developers quick feedback to ensure their code is working so that it can move through the release pipeline quickly. We—Jessica and I—want to put it to you

like this: The way we are testing is what's causing the problem. It is making the airline out of touch. It's taking too long."

"Next," added Jessica, "we have to create or find data to feed all the functional and non-functional tests running both before production and in production. Ideally, the data should be synthetically generated. Then, we'll need to monitor those tests running in production as well as what real users are doing in production. And you must learn from their experiences—the users', I mean—otherwise, you just delivered useless lines of code."

Arthur interrupted. "Wait . . . did you say tests running in production?"

"Yes," said Jessica, "and then there's non-functional testing. We talked about that, too. People usually only think of functional testing. Continuous testing is not just about functional testing automation. Non-functional testing, like performance and security, is equally important. People usually view these as one-time activities before production that happen once a quarter or maybe once a month, but they're wrong. And the skills needed are different than those used in traditional functional testing.

"What the user group was saying was that with performance testing you should also shift it to the left. Don't start your performance testing just before releasing to production. And certainly don't test 'just' in pre-production environments. You should think about continuously running your performance tests in production to provide synthetic transactions for monitoring how your application is performing." She added, a little humbly, "I hope I haven't said too much of what you already know."

"Are you crazy?" Arthur shot back. "Push load to production? That's what caused the screw-up last Sunday. And doing more performance testing? We already have a Center of Excellence that can't keep up with demand, despite having a great engagement process."

"Yes." It was Alicia interrupting this time. "But having a central team that handles all the requests from Renway's IT organization sounds limiting. And you've just proved the point by saying they can't keep up. That won't work for an agile team."

She leaned forward and rested her elbows on her desk. "This is about COE test teams with limited time. You know what happens? Most problems don't get fixed before release. This means a six- to eighteen-month turnaround. Now, this weekend, we proved once again that you've got to find the problem pretty much as it occurs. What's the expression out there on the street? If you see something, say something. That's what we must become now. We are all testers."

"I agree," said Leigh, "but I think we need to recognize that my QA team is not up to this task and neither is Arthur's Dev team or Owen's Ops team. We have to work more closely together to define a new role of QA within the IT

organization to realize this vision, and that means redefining the roles of all the departments, Dev and Testing included."

Alicia surveyed the group. "I must say I am pleasantly surprised. I had no idea we had such a range of talent in the room. I think we must agree that nowadays a customer can switch to a competitor way more easily than before. They've got the Netflix mentality now: instant, up-to-date, and perfect. And that's why we have to do what we have to do. Easy. Peasy. Lemon Squeezy."

"Point. Click. Burrito," Leigh said quietly to herself.

* * *

It was now noon on Wednesday. Leigh was at her desk in the IT dungeon reviewing a report on the cutover failure. She knew she should step out and get something to eat, but, as usual, there was too much to do. She hoped the 10:30 bran muffin would hold her for a while.

She held in her hand a printout of the initial post-event report following the cutover failure. It was pretty much what had been said during the hangar meeting: the mainframe OS was being upgraded from 32-bit to 64-bit, which required all programs to also be converted. This is a manual process that takes time. People must manually inspect the conversion process and then do manual tweaks to ensure the programs are properly converted. It took longer than expected.

She was having trouble concentrating. Partially due to hunger, perhaps, but also because of Alicia's mention of Kitty Hawk. Did she know? She was CIO. She *must* know. But that didn't mean anything. But MidWest? It was only operating thanks to healthy tax breaks and a questionable "too big to fail" mindset that had wrapped its roots tightly around the governor and his cronies in its home state. It was too big. And it was failing. If Renway were to swallow this fish, it would choke and drown. And even if it staggered along for a while, any type of merger would mean there would be duplicate roles. A lot of people were going to lose their jobs. They would become deadwood, Renway people or MidWest people or both.

She sat back. This was a tough business. She thought of the old saying she had heard at an aviation conference once: "If you want to make a small fortune in airlines, start with a big one."

She realized it was more important than ever for Renway to be in the better position to survive. They would have to integrate the MidWest legacy IT systems, especially the reservations system, and still come out as a superstar. And, apparently, they would have to do all of this in time for the World Cup.

She checked her email. There was the response she had been hoping for. The one with the headers all in German. She read it carefully. Six-thirty p.m.

today … Logan departure lounge … yes, she could do that. She smiled. "Dinesh should be in on this," she thought, and quickly typed out a text to him.

Her desk phone buzzed. It was Sam, the security guard in the lobby. Sam, with the Richard Pryor mustache and piercing eyes that saw every face and forgot not a one; the retired cop who, at 60, could still chase and bring down a bad guy on the run, but who was so charming, the rumor was he had never paid for his coffee in his 12 years at the desk. People always seemed to have an extra one just for him.

"Er, Leigh?" he said, in a strangely curious voice.

"Yes, Sam?" she replied.

"There's a … uh … a robot here pulling a trailer. It's got a guy's face on it. Says his name is Steve Liberace—"

"Fibonacci," said a voice in the background.

"Fibronashi," said Sam, mangling the name further. "He … uh … says he's from a burrito place—"

"Veneto Burrito," said the voice.

"Yeah, anyway, I can't have him sign in 'cause he … uh … has no hands, but he said you ordered some food?"

Leigh suppressed a smile. She hadn't ordered anything, but she would be happy to receive some unsolicited food at this point. "Yes, Sam," she said, "I can vouch for him. Just use my ID number."

"OK," Sam said, his authority restored. "I'm sending him down to you. Course, I have to get up to press the elevator button, cause, you know …"

"Yes, no hands. Thanks, Sam. I'll get you a coffee later."

"Much obliged," said Sam, and hung up.

Moments later, Leigh could hear a mild whirring, and a Foray trundled straight past her door and continued down the hall. It was a virtual presence robot, like a small Segway with a monitor where the handlebars should be. This one was towing a small cooler with rubber wheels. She waited a few seconds. She could hear the whirr change its tone, and she gathered it was backing up. Sure enough, the cooler appeared, pushed by the Foray. Apparently, whoever had developed and tested the device had thought far enough ahead to ensure the robot-trailer connection used a solid lynch pin and not a soft cord.

This time, the robot turned left and entered her office, towing the cooler behind it. On the screen was the smiling face of Steve Fibonacci. At the base, between the wheels, was the Veneto Burrito name and the rocket fork logo. On the vertical mast, between the wheels and the monitor, was the tagline: Point. Click. Burrito.

"I brought lunch," he announced.

"You make quite an entrance," she replied. "Thank you. Does your droid have a debit card terminal so I can pay?"

"That's not the droid you're looking for," said Steve. "This one just does deliveries. It's a prototype. And besides, lunch is on the house."

"I appreciate it very much," said Leigh. "May I?" She made motions of getting up to go and open the cooler.

"Of course," replied Steve.

"You know," said Leigh as she opened the cooler and extracted the heat-retaining box within, "I can't quite understand why you're doing all this. The advice, the food. I mean, I appreciate it, but you must have much more to do, running your empire."

"I do," said Steve, "but long ago I learned two major lessons that helped me get to where I am today. The first is the value of maintaining a great network of great people. You were always a part of that network. We just lost touch. My bad. And the second is the importance of paying it forward. I had a mentor or ten in my early days, and I have them still today. Knowledge is a wonderful thing. You can give it away, yet you never lose it, and it always doubles back to become more. So, I just wanted to give you a couple of snippets of knowledge to help you put continuous testing into perspective. You probably have a battle in front of you."

Leigh sat back behind her desk. She opened the heat box and pulled out a succession of paper bags, each with the rocket tine image.

"Ah, the paper bags," said Steve wistfully through the Foray's speaker. "They balance moisture so much better than plastic," he said, "and besides, there's a sentimental family connection."

"Can you share that?" asked Leigh.

"Of course," said Steve. "My great-great grandfather was the first Italian cowboy to be thrown in jail in the Wild West. He was too poor for real western clothes, so he wore cardboard chaps. He was arrested for rustlin'." He laughed out loud at this. "You can use that one at your next meeting if you like."

"Thanks, I will," said Leigh, as she started in on the food. The mention of the bags jogged her memory back to the staff meeting, when Dinesh and Jessica had both taken a Veneto Burrito bag with them as they left.

"Anyway, to business," said Steve. Leigh returned her focus to the screen. "Let's look at what actually constitutes testing as it has evolved into the world of DevOps. I built most of the Veneto Burrito app architecture, so I have some experience here. What you need to know is this:

"Over the past ten years or more, there has been a dual evolution. One part organizational, the other tooling. As we have evolved more toward automation, the types of applications that need to be tested have changed. For example, there was once a time when there was no such thing as mobile apps at all. The architecture has changed and evolved. Also, users and the percentage of those users who are customers have expanded. That means that testing today is not just for functionality or for ease of use and scalability.

"Organizationally, this evolution has gone from 'developers who forward to a group of testers who sit separately in a Center of Excellence' through to agile. The COE has become something of a double-edged sword. Yes, people in COE became more professional, but by virtue of being part of a COE, it became difficult to diffuse the knowledge—it became a bottleneck unto itself. Now, companies are trying something new, which is not separate groups or a COE, but instead they are embedding the tasks of testing into the entire team. There is still a COE, but now the acronym stands for *Center of Enablement*. That group is meant to be the gurus that will enable all agile teams with practices, tooling, and reusable frameworks. The actual testers became fully part of the agile teams. Did you get that, Leigh?"

"Yes, thanks," said Leigh, "I was taking notes."

"And, finally, if you want to talk about how continuous testing helps with change, have a look at our app. Maybe show it off at the meeting. The food-ordering app as it stands is a process in production. But how do we make changes to our production process? How would we deal with, let's say, introducing a vegan burrito? Or a Halal selection? Or a new form of payment? The issue here is the capacity to introduce change into the process. This needs testing to happen earlier and more often. Testing can't be thought of as a single area that is done somewhere toward the end of the process. You don't just hand over your beautiful design to a group of testers anymore. That's going away fast. You can't expect QA to work with a black box and come up aces. Change happens, and the product must move with it in a more constant fashion. You get what I'm saying?"

"Yes," said Leigh, "I do."

"You have a good crew on your hands, Leigh," said Steve, "it just needs to learn how to rig the sails again. Drop me a line if I can be of any more help. I will be your beacon in the darkness." He saluted proudly. The Foray and its trailer turned away from Leigh's desk and drove straight into the wall.

* * *

"Thank you for coming," she said to the group, who had convened for a quick meeting in her office at the end of a busy Wednesday. It was 4:03 p.m. Admittedly, motivation to attend had been bolstered by the fact the invite had been co-signed by Leigh; her boss, Derek Zajdner; and his boss, Alicia. Given

that Derek, who was still recovering in the hospital from surgery to torn quads in both legs, was on equal footing with Arthur, Giles, and Owen, and given that they all reported to Alicia, it made the decision to accept somewhat easier.

Leigh had a PowerPoint slide up on the screen. It read: *What Constitutes Continuous Testing?* Beneath it were three items:

- PEOPLE: Empower developers with testing capabilities
- PROCESS: Automate everything
- TECHNOLOGY: Leverage open source and cloud

"So, following up from Monday," she said, "this is just a real brief 15-minute touch base to build up our definition of continuous testing. Let's go around the room. Comments please. You've had some time to think about it. Now, what do we at Renway consider as continuous testing? What have you learned from people on the outside?"

Owen raised his hand to speak first. "I see continuous testing as a kind of automated assembly line with testing at every step—automatically triggered—from first code check-in all the way to production. All the testing required for certification would be triggered automatically. It reduces the mean time to validate a change. This would include all aspects of testing all the way up to integration and performance testing and monitoring in production."

"OK," said Leigh.

Giles was next. "Continuous testing would be how you test applications in a world of continuous integration and continuous delivery. An effective and efficient application-centric test regime that is end-to-end in scope—across CI, CD, and production."

"Great," said Leigh, "and—"

Giles continued. "The goal is to be as efficient as humanly possible to get changes into production in the shortest possible lead time with the highest quality and top customer experience. It's gotta be a system."

Giles wouldn't stop, but that was OK. "Just having tests at separate phases is one aspect. But there are constraints—it can take up to a week to have the environment prepared and available for the testing you want to do around here. I can't wait a week to do a smoke test. I need a way to test in other environments that can simulate some of the services we can't easily and cheaply duplicate. And I need the ability to create and provision test data into those environments easily and quickly. Really, I don't want to be submitting service-desk tickets for this stuff like we do today. It needs to be self-service. I need a tool to reset environment data."

Jessica spoke up. "What about feedback? Some people think as long as you can test faster, you're doing continuous testing. But that's not right. There's a lot of discussion about balancing testing against the intended business impact, and not just to functional and non-functional requirements. You need feedback for this."

Leigh added *Feedback* to the bullet list.

Owen spoke up. "When developers started adopting agile, there was this misconception that testing was already being taken care of, so they placed their focus on development and delivery. They started learning new capabilities, such as deployment to different environments, but there was clearly a quality piece missing as team velocity increased and so did defects."

He sipped his coffee. "Finally, crappy testing speeds up, well, crappy testing. And regression tests? Their test coverage is highly variable. For some apps, it can be as high as 85 percent. For older apps, it is lower, and 20 percent is not uncommon. Continuous testing is about testing the right thing to validate that what you're giving to the customer is what they expect. Not just doing the same thing faster, but something fundamentally different. It's about quality, and it's everyone's responsibility. I heard a great expression once. A driving analogy. 'If continuous testing is not built into continuous delivery, you just speed up to the next red light. You're not doing continuous testing well if you have to stop again. That's why feedback is so crucial.'"

"OK," said Leigh again. "I think we have the start of something at least." She looked at the slide on the screen. It read:

Continuous Testing Defined

- Automated assembly line with testing at every step
- Reducing the mean time to validate a change
- How you test applications in a world of continuous integration and continuous delivery
- Getting changes into production in the shortest possible lead time with highest quality and top customer experience
- Feedback

Leigh looked at the wall clock. It was 4:20. "OK," she said, "that's our 15 minutes. Seventeen, actually. This is good for now. I know you all have other things to do." She thanked everyone for coming, and sat down at her computer, trying to figure out how to compress 15 minutes of dialog into three or four sentences.

* * *

Fifteen minutes later, she walked back into her office, a fresh Starbucks latté in hand. She had ducked down to the food court following the meeting and had thought about enjoying some time outside for a change, but the fall breeze was just a little too cool for that.

She entered the QA department, waved quickly to Stacey at the desk, and then stopped. Arthur and Owen were sitting in the seats outside her office, obviously waiting for her. Owen was thumbing something into his phone. Arthur was leafing through an old issue of *National Geographic*.

"Is this an ambush, or are you taking me out for my birthday?" she asked.

"Is it your birthday?" asked Arthur.

"No."

"Then it's an ambush," he said.

"Well, no," interjected Owen, "it's not an ambush *per se*. But I think you need to become aware of something that might throw a wrench into your continuous testing initiative, as great an idea as it may be."

"Then please come in," she said. She was determined to keep the upper hand, so she decided to start out by questioning them instead.

"Tell me," she started, "why do we have a QA department?"

They both leaned forward, eyebrows raised as high as eyebrows can go.

"No, I don't mean 'why do we have QA'; I mean, why do we have a physical department? Why do I have an office and receptionist and physical walls and such?" She waved her arms around. "This shows just how traditional and siloed this IT organization is. We shouldn't need this. An IT organization's QA resources, like testers and test automation engineers, should be embedded into the agile teams. Not just online, but physically. There really should not be a 'QA department area' of the floor."

"How would you know where they are?" asked Arthur.

Leigh just stared at him. "The point," she said, "is that these positions shouldn't be so sharply defined. There needs to be a cultural change. I need to give up a lot of my staff so they can be embedded into the agile teams. I should keep some top guns around to set the testing direction, tooling, reusable frameworks, and to coach agile teams how to best go about ensuring quality across the SDLC in each application. But I don't need a huge QA staff anymore."

She looked at both of them. "So, how can I help you?"

Arthur began. He started pacing. "Well, in light of what you just said, and also what you said at Monday's meeting, as you know, each department within our big airline family started their own agile and DevOps journeys independent of one another. This has been the case for years. They all have vastly different

levels of maturity, and that means you will have to upskill your own team to properly support each department's DevOps journey. Does your team have the skills required to oversee continuous testing across the board?"

"No," Leigh sighed, "as I mentioned to Alicia, they do not."

Arthur pressed on. "Do you have the budget to hire staff with those skills?"

"No," Leigh sighed again, "I do not."

"Well, that's probably a good thing," added Owen, "because every department I have spoken to is now saying they can't have testing done outside of their own teams, as that would certainly be a bottleneck and be against all agile and DevOps best practices. So, we are wondering what your plans are regarding your continuous testing initiative. You know, whether it really has a place in the company?"

"This isn't 'my little initiative,'" Leigh answered back, trying to hide how exasperated and betrayed she felt at that moment. "This is progress. This is taking an airline—an industry—out of the 1970s and turning it into a service business. Because that's what it's becoming!"

She felt ashamed. Losing her cool was definitely not in her personal playbook.

"You've both been at all the meetings this week."

"Yes," said Arthur, "and as I said to you on Monday, the reservations upgrade has been fixed. We found the problem, and we fixed it."

"Anyway," he continued, "I don't want to ruin your day, but you're chewing on a pretty big elephant here. We'll do what we can to help, but I know my people in development are permanently busy right now. They don't have time for big changes. They don't need any extra work either. I've seen these things come and go, and at the end of the day, it will still be the same as ever. Too much to do, not enough time, and always someone screaming. That's why work is called work. It's hard."

Owen spoke up. "Look, Leigh," he began, "we're not trying to be tough on you here. We are all part of the same team. But even on a sports team, people play different positions. You know better than anyone that anyone in Testing will be less than thrilled about losing the work they have to a bunch of people who really don't value the testing craft."

Arthur added, "Every few years something comes along that everyone thinks will be the next latest and greatest. And yes, we go from punch cards to floppies to the cloud, but in the end, it's all the same thing. Someone has to build it, and someone else has to test it. And you guys need to make sure it's ready. It's as simple as that. Anyway, that's all I wanted to say."

Owen and Arthur left the room. Leigh slumped back in her chair and just stared through her office doorway into the bullpen beyond. Steve Fibonacci's robot Foray cruised by, obviously still lost. This made her laugh for a moment, but then she realized how similar she felt, lost in the depths of the Renway IT dungeon. She looked at the photo on her desk of her, her partner, Casey, and their dog, Copernicus. She suddenly longed to be able to step inside that picture frame and transport herself instantly home.

She checked her email. Scanning through them, she selected one from Dinesh. It read, "Hey! Jessica & I have been working on an explainer video. We got a friend of ours to do the voice—real broadcast like. Here's the audio track. LMK what U think."

She plugged in her headphones, clicked the link, closed her eyes, and relaxed back in her chair. She wished Copernicus were there. He always knew how she felt, and silently, with fervent dedication, he would show how much he truly cared in a way only dogs can. The audio played:

"Up until recently, the concept of continuous testing in software development was not truly achievable. It had always remained far easier to have people perform build-and-test. There was no compelling need to motivate change.

"But as business moved into the age of social media and the cloud, we saw the application economy come to the industry and its supply chain. Legacy businesses started to see IT no longer as a supporting arm of the business, but as an enabler of new revenue streams. As that changed, people started to see a competitive advantage.

"Where infrastructure solutions had traditionally been seen as extremely expensive, cloud technology emerged as the easier alternative. This became the next step toward business agility.

"People started to realize the need for change as the continuous delivery mindset evolved. With the emergence of agile, there was another big push, bringing testers into the team, but still companies tested the same way, using manual test cases. Performance testing and security testing were still being done outside of the agile team, just prior to release to users.

"The market was starting to demand a change in the way testing was done, but industry did not change quickly enough. It was thought that changing and re-educating thousands of testers would be too costly.

"The emergence of DevOps took the first steps toward automating the testing process, delivering business value to the hands of the users and eliminating a key bottleneck. DevOps became the ultimate motivation to change testing, and fully introduced the notion of continuous testing as a procedure that must happen throughout the software development lifecycle.

"This development helped companies stop thinking about testing as an event that was to be done at a specific point along the line. People began to recognize that testing was something that must be done by everyone all the time.

"This required a new set of tools that would help automate the process and shift it. It had to be shifted left, back to the developers all the way along the lifeline, and it also had to be shifted right, closer to production, and sometimes in production, learning from failures and incidents and bringing it into the lifecycle.

"The change had to drive the following message: that I as a developer must be responsible for the brand. It's not just the job of marketing. From there, we started to see the emergence of purpose-built testing tools for supporting the shift left/right movement.

"With DevOps, testing and quality assurance are now moving to the left—shifting left further upstream in order to scrutinize how the design works from that starting moment. The objective is to ask, 'What is the opportunity to inject quality all the way through?'

"That's the role of continuous testing: to find unexpected behaviors and fix them as soon as they are injected. It is not a scheduled activity. The intention of continuous testing is to uncover the unexpected behaviors across the entire software development lifecycle. This requires a different mindset and new technology to support it. This is the new role of testing in DevOps.

"A credit card is a typical software development challenge. In today's marketplace, a cash transaction system must process a credit card quickly. But what is 'quickly?' If a developer's definition of 'quickly' is three seconds, but the marketplace defines it as three one-hundredths of a second, then there is a defect in the process.

"Companies must shift left to understand this new requirement in the same way and validate it in order to prevent this type of non-functional defect from ever getting into the system. The verification of the coding style guide by developers must happen automatically to validate if the code was written correctly. Then, the testing activities that follow must focus on whether the right thing was built. With continuous testing, quality becomes everyone's responsibility. Everyone in the lifecycle, beginning to end, has the opportunity to find and fix."

The audio track ended abruptly. Leigh opened her eyes and pushed the onscreen rewind symbol back 20 seconds, but the player showed this was the end of the recording. She checked her watch. 5:45 p.m. Already a long day.

She updated her Outlook calendar to reflect the half-hour spent with Owen and Arthur, along with the half hour spent with Dinesh's voiceover recording. For the entire period, she simply entered, "Continuous Testing Imperative Now." She scanned tomorrow's schedule. The hour between 11:30 and 12:30 had been set aside for Adam's DOS Prompt. "Great," she thought, "Jessica should see this." She forwarded the invite and added a brief explanatory note. She smiled as she thought about getting home to Casey and Copernicus.

CHAPTER

5

Taking It to 11

"What do you mean, dotPlane is selling off its fleet?" Leigh asked Dinesh as they drove toward the airport in the fading light of Wednesday's sun. "Are they buying bigger planes?"

"That's not what it says here," replied Dinesh, "just that they're starting a selloff process, looking to get rid of eight of their 737s in two years." He flashed his phone at her. He was on the *Aviation Week* website.

Neither of them spoke. All that could be heard was the distinctive drone of the Porsche's engine.

"They want to get bigger," Dinesh said finally, still reading from his phone's browser, "and they're going to do so by shrinking their fleet. It's like they want to become an Uber of the skies or something," he said, jokingly.

They arrived at Logan and circled around to the employees' entrance. It was out near the fuel-tank farm, where the Jet A-1 and B were stored. This was an alien and hidden landscape for most civilians, loaded with huge tanks that fed pipelines that terminated at each gate. Parked nearby were some of the dispenser trucks that pumped fuel from the gate pipelines directly into the planes' wing tanks, as well as some of the low-profile tanker trucks that took care of off-stand refueling and defueling.

They pulled into a space in a lot that overlooked a main runway. "What I would give for an hour out there," she said to Dinesh.

"Have you ever really pushed this car?" he asked.

J. Scheaffer et al., *The Kitty Hawk Venture,* https://doi.org/10.1007/978-1-4842-3661-1_5

"I went to a Porsche owners' club meeting once, at a motorsport track. Lots of curves. They gave each of us a few laps with a professional Formula One racing coach. I had this French guy. He sat where you are and criticized my entire run. Told me I had to accelerate into the curves, not brake. Then, we switched positions, and he showed me what this car could actually do. I was never more terrified and amazed at the same time in all my life."

"It's interesting, isn't it," said Dinesh, "when you get someone who shows you how to accelerate into a curve, into the unknown. Into the future. To floor it. Such an amazing concept. I'll have to get me a Porsche," he declared, "or maybe a Tesla."

They parked, locked the car, and headed to the terminal. This, too, would be an unknown, but Leigh felt she had to do this, if only to beat down that nagging feeling that she was missing something big. Something world-changing.

They headed to the departures level and located the TSA gate at the far side of security. They presented their airline security credentials and entered the Departures area, where they checked the departures screen and found their target: Lufthansa 425, departing at 8:30 p.m., two hours from now, gate E10. They headed over and scanned the collection of travelers who were biding their time until the boarding call 90 minutes from now. Some families, lots of restless and tired kids, some backpackers, a couple of people sleeping on the floor, and almost everyone else engrossed in their phones. No one Leigh wanted to talk to. She felt momentarily defeated. She looked around and saw a pub. A faux English-style pub with fake wood, lots of Union Jacks and plenty of ads for Guinness. Worth a try she thought.

She and Dinesh entered and looked around. At a far table she saw a group of four exceedingly corporate-looking people—three men, one woman. She looked at their hand luggage. Success! One of them had a tag with the red, yellow, and black logo of AbenteuerLuft, the adventure airline. Known on social media as AbLuft, the airline operated out of Hamburg and flew to several destinations in Europe, North Africa, and Russia. And they were just about to launch a twice-daily transatlantic run out of New York. She took a chance. "Guten Abend! Wie geht es Ihnen?" she said, in her best high school German, knowing full well all of them were likely able to speak perfect English.

"Ah, yes, Ms. Freemark," said the elder of the three men, "so glad you could join us. Please!" He drew up two more stools, and introductions were made all around.

Leigh launched into her pitch. She expressed hope that they had enjoyed the International Civil Aviation (ICAO) seminar in San Diego and repeated the message she had included in her email to them last evening, that she was not here in any official Renway capacity. But off the record, as it were, she wished to take advantage of their two-hour layover here in Boston to learn what AbLuft might look for in a World Cup partner. Yes, it was late in the game, and

yes, this was not the usual protocol, but they were new to North American shores, too, so let's just call it all market research.

Beers arrived. Dinesh and Leigh ordered German beer. The Germans ordered Stellas and white wine. Go figure.

* * *

On the drive back, they dissected the meeting.

"They seemed interested in us, in theory," Leigh said. "But they seemed concerned as to how many aircraft we could make available to JFK on a daily basis. I think they would have been more comfortable if we could have promised them a larger number." In the back of her mind, the image of the MidWest fleet appeared. She bit her lip again.

"They brought up the cutover fiasco in the politest way, I thought," said Dinesh. "They were happy to see it resolved, of course. I have to hand it to you," he added. "These clandestine operations are not the typical fare for a specialist from the IT department. Are you sure you don't want to come over to the dark side and work for me in marketing?"

Leigh smiled. "I know. I can't help it. I don't want to get anyone's nose out of joint on the oak door floor, and I certainly don't want to get fired. But you know what it's like, Dinesh." She turned to look at him as much as was safe to do while driving. "This is a different world now. People need a broader vision. They can't stay pigeonholed, or they'll lose sight of everything. It's not like I was going to bring out a contract for the Germans to sign. But this is the Uber thing. The dotPlane thing. People like Arthur just can't see that business is different now. It's not traditional. That goes for marketing the same way as it does for software development. It's like digital newspapers and paywalls. Everyone's struggling to make yesterday's technique fit into today's technology. And while they're doing that, some nobody company skates in on a pile of VC money and invents a whole new industry. I don't want to see our planes go the way of taxicabs, and I'm not sure we can wait until the person with the right title in the right department figures this out."

Dinesh smiled and looked out his window.

* * *

It was Thursday morning, 11:05. The air was uncommonly warm for early autumn. Summer did not quite want to say goodbye yet, and that was fine. Leigh was parked in the guest lot at the George Hedley Building, about five miles from her Renway office. She shifted to the passenger seat, opened her laptop, and pored through her email. The one that caught her eye was from Alex: "Saw You at Logan! Missed You. Coffee w/ guest?"

She opened it up.

"Saw you in the pub at Logan Gate E yesterday. I was in a Homeland Security briefing across the floor. You looked quite involved, so I didn't want to interrupt, but my study buddy did. You probably know her. Sylvia Finch from Aurora? Any chance you can get back to Logan this aft? – Alex"

Sylvia Finch? She was a legend in this business. She was Managing Director of Quality Engineering at Aurora Airways, an airline famous for being number one in both customer and employee satisfaction year after year. They seemed to do everything right, not only in person but online too. Their apps seemed to work flawlessly. Each employee was issued shares in the company on day one of being hired. Leigh shook her head. Why would she have time for me? she thought.

"Pretty slammed this morning, but could do 3:00, if that works for you and Sylvia," she wrote. She hit Send and then checked her watch. 11:20. She looked around and saw Jessica arrive. She drove a Jeep. Leigh liked that. Jessica's demeanor made her seem like someone who would go for a practical compact or hatchback and great gas mileage. Just goes to show that you never can tell. Leigh closed her laptop and got out.

"Glad you could make it," she said.

"I wouldn't miss this for anything," replied Jessica.

* * *

They entered the Hedley Building and looked around. Sure enough, there was a poster on an easel with an arrow pointing to the elevator banks. "Adam Dobrovodsky — DOS Prompt — Continuous Testing Disciplines — 11:30–12:30. — 12th Floor." They took the elevator and stepped out into the twelfth-floor vestibule. They could hear the murmur of an assembled group down the hall. Another easel pointed them to the twelfth-floor cafeteria, where they could see about fifty people seated on folding chairs laid out classroom-style in rows of ten. Others were helping themselves to coffee and pizza. Some had brought their own lunches. Adam waved and walked over to them.

"So glad you could make it," he said. Leigh introduced Jessica to Adam. "Please call me D—for Dobrovodsky," he said. "I'll catch up with you guys after the session."

"It's amazing," Leigh thought, "how difficult it can be to get people to show up to a meeting on time or to reply to your email, but pizza? They'll come out of the woodwork for that." They chose two seats in the back row at the end, by the aisle. There was a projector with the *de rigeur* PowerPoint screen, which read "Disciplines of Continuous Testing." At exactly 11:30, Adam introduced himself.

"Good morning, everyone," he began. "I am Adam Dobrovodsky, VP of Quality and DevOps Engineering here at Consolidex. Welcome to DOS Prompt. That

stands for Dobrovodsky's Open Space. If there is anyone in the room who remembers what an original DOS prompt looks like, please treat them with great reverence, because they are very old."

"And if they're still in IT, they will have a haggard and crazed look about them!" said someone close to the front.

"Well, from my perspective up here," said Adam, "that would include all of you." Laughter rippled through the crowd. "Anyway," he said, "to business. As we continue our learning process around continuous delivery initiatives, it seems that one of the items we as a community have overlooked somewhat is quality. We're certainly accelerating our code-release process, but we haven't thought enough about how to ensure that quality keeps pace with all the speed we're starting to enjoy. We can't do testing like we used to before continuous delivery. And manual testing is too much of a bottleneck in the application pipeline. But if we don't address this, it will ultimately defeat the purpose of the whole initiative."

He looked into the audience, trying to make eye contact with as many people as possible.

"To embed quality in our applications, we will need to add more modern testing practices in the form of small quality checks performed throughout the application pipeline. This would enable our code to be constantly tested in small batches, all the time. This is part of a new emerging discipline called continuous testing. That's our topic this morning, and I am joined here today by a friend and colleague of mine going back way too many years, Wahid Annan, from Awayz." The PowerPoint screen showed the Awayz logo.

"Travel bookings extraordinaire," she whispered to Jessica.

"They know more about air travel than we do," Jessica whispered back. That made Leigh pause for a moment. She made a mental note to remember it.

Adam continued, "Wahid is leading a transition toward continuous testing at Awayz by transforming their Center of Excellence into a Center of Enablement. These are terms you will likely have encountered yourselves. Wahid has been sharing his journey through his LinkedIn posts, which I invite you to check out. Hopefully, we can avoid many of the bruises and scars they received in the process."

That comment got a wave of sympathetic laughter and some heads nodding in agreement.

"Wahid plans to cover ten key items around continuous testing in the next hour. So, buckle up and feel free to ask questions. Wahid?" Wahid stepped to the front of the room to energetic applause.

"You're probably all familiar with the idea that continuous testing leads to the need for measurable results at speed," he began. "Testing and QA obviously

pose significant challenges to achieving this. Most testing organizations are just not set up to do it right. Even as development goes agile, the end-to-end pipeline remains in waterfall.

"Testing has to become part of the fabric of how code is written, released, and operated. It has to start before the code is developed and must continue after it has been released. That's a pretty tough thing to embrace when you have teams of developers, testers, and release managers, each doing their own thing independently, and none of them thinking about how to efficiently test and manage risk end-to-end."

He pointed his remote at the screen behind him to reveal an image of a tricked-out 1970s-era Ford Econoline van, with two people in bell-bottom jeans standing in front of it. Someone at the back of the room shouted "Yeah!" and then sank back into silence upon realizing how young the rest of the group must have been.

"The way testing is done hasn't really changed much in the past several decades," continued Wahid. "Other disciplines, like business analysis, development, release management, operations, and others—they've all evolved considerably, but testing, not so much. In most organizations, it is still very waterfall, planned and executed by a siloed organizational unit. It still involves lots of assets that are maintained at high cost, independently of the code bases to which they relate."

He advanced the slide to replace the Econoline with the now classic photo of Elon Musk's Tesla and Starman heading toward Mars. This brought a noticeably larger response from the group.

"We must break QA stagnation!" he exclaimed. "Testing must evolve and get off the critical path to release software with quality at speed! That's what we realized at Awayz, and we were able to get the entire IT organization to support a massive change in the way we looked at software quality."

He looked across the audience. "Let me ask you: how do you think you should accelerate embedding testing to support all the speed you're shooting for?"

"Test automation!" someone shouted from the back of the room. Was it the Econoline guy? No one could tell.

"For sure," Wahid responded, "but let me ask you something else. Is that the first step?" There was silence in the room.

"What if I told you that test automation is not the first thing to be tackled?" he asked them. "It sounds kind of counterintuitive, but what I learned is that, after we had all our API tests and GUI tests automated, integrated into a continuous integration agent, and ready to run for one of our pilot projects at Awayz, those tests had dependencies that had to be fulfilled before they could be executed. Can anyone guess what those were?"

"Test data," said someone in the front row.

"Bingo," said Wahid. "And also interfaces and environments. We have to interface with third-party global distribution systems and partners, as well as with our own internal-interfacing systems, such as revenue management. All of these have their own test environments, so we must coordinate with the third-party and internal-interfacing systems before we can run our tests. And, more often than not, we end up being blocked and having to wait for these environments to be made available for our testing.

"When we performed a value-stream mapping on our end-to-end SDLC, it became quite clear that even if we had the test data to feed to the automated tests, they would fail if we hadn't coordinated with all those groups to ensure their environments were configured to be consistent with our test data. That's how we determined the bottleneck we should tackle first was not actually test automation.

"So, based on the outcomes of value-stream mapping, here's what I learned." The PowerPoint screen changed to:

Where to Start?

Remove environment constraints and free up your developers and testers to do their thing, even if manually. The most common and scalable strategy is to virtualize any and all interfaces you don't own or control.

Automate test-data provisioning and management to feed your tests, manual or automated, on demand.

Automate your tests so they can run against your virtualized interfaces using the appropriate test data.

Have your pipeline orchestration engine set up in such a way that there is no manual intervention on the preceding steps.

Now that you've got the basics working, focus on the complementary disciplines that almost no one talks about.

"Let's dive into these in more detail," Wahid said. "These are the ten items D warned you about." The PowerPoint screen changed to:

1. Virtualize Environments

"Why virtual environments?" asked Wahid. "Continuous testing means testing more frequently, and that means testing against multiple test environments. All the time. When you set them up virtually, you remove constraints, as they'll always be available. You control them.

"For example, at Awayz, we were constantly releasing code for our dot-com site, but users found a ton of issues in production. That was a major blow to our flawless reputation. It was pretty clear we did a great job in accelerating code release, but now we needed to ensure that the code had the quality our users expect from us.

"The problem was that the dot-com site touches countless environments, from reservations to ticketing, from revenue management to customer loyalty. Some through APIs, others through other types of messaging interfaces. Some had modern architectures, others were monolithic legacy client/server or mainframe systems. When we tested our code related to a release of the dot-com website, there was no way we could coordinate our testing with those multiple environment owners. We had to break those dependencies by virtualizing them so those wouldn't slow us down.

"The concept of ephemeral environments became top of mind for us. We needed to have the environment available on demand. And we wanted our pipeline orchestration engine to manage all that. Because we wanted to fully control all environments we touched, the easiest way was to start using concepts such as infrastructure or environment as code. That was another new concept for us. We had to learn it from scratch. By treating the environment as code, we had the ability to easily and seamlessly spin up the environment we needed, when we needed it, all orchestrated through our pipeline, with the configuration of the environment coded in a way that was associated with—and versioned with—the source code to which it related. Completely automated and unattended.

"Ephemeral environments that can be provisioned on demand whenever a tester needs them is a huge leap forward. You can only get there if you can also virtualize away the service and system dependencies that are causing testing to be queued up. Currently testing needs access to fixed environments that have all the dependent services available and configured." This ability to leverage service virtualization to enable a shift from scheduled, serial testing in fixed environments to on-demand, massively parallelized testing in ephemeral environments is a foundational capability that can revolutionize how and when testing gets done in a truly efficient pipeline.

"Once environments were no longer a constraint, test data became the next target for us. How many of you have to deal with going to production environments, grabbing data, subsetting it, hopefully scrubbing it, and making it available in the test environments?"

"That's my day job . . . and it's painful," a woman in the middle of the crowd said.

"And why is that?" Wahid asked.

"I have to provide test data to the business for their own experiments as well as to the development and testing teams. Everyone had learned to accept that

it takes time to get the data in the right condition, as our application landscape is too complex. But in the past few months, all of a sudden no one can wait, and I'm being asked to automate the manual steps it takes me to condition the data. I have no clue how to do that . . . but no one listens."

Wahid nodded, smiled, and thought for a moment before responding. "Sounds like you're at a point where colleagues are seeing you as a bottleneck. That's a good thing, because you're in the spotlight, a perfect time for you to ask for pretty much anything you need in order to make it happen. Especially if you need help to learn the new skills you said you don't have. We had to go through the same thing, and I am not ashamed to admit it. We just didn't understand how to crack the test data management problem. But we got there in the end. Here's how we approached it."

The PowerPoint screen changed to:

2. Test Data Management

"One of the key challenges for us," Wahid began, "was devising the ideal set of test cases needed to fully test a system, which helps shorten the test cycles across the SDLC. But for that you need to know your test coverage. And I mean actually know it, not just a gut feel. Otherwise, you end up manually designing more or fewer tests than you need, spending a lot more effort and time that won't really identify more or fewer problems. And you still won't know what you missed.

"When we started our journey at Awayz, we had this metric thrown at us by consultants that 50 percent of testing time is consumed by trying to find or create test data. I had a suite of tests that had to run every day. This is critical to continuous delivery.

"So, we started creating Excel macros to condition the data before testing, then it became an Excel app. Then, we evolved it to a dot.net application. That tool today handles 100,000 requests every month, all in the cloud. We chose to build a system rather than buy one, but if you do that, you must be careful not to replicate business logic into your applications. Otherwise, you'll end up with a maintenance nightmare as your application logic changes."

"Synthetic data?" a voice from the crowd asked.

"Yes," Wahid replied. "Synthetic data is needed for continuous testing because you must have the right data and the appropriate diversity of data for positive and negative scenarios. You won't find all that diversity in production; that is critical. Also, you won't be able to bring data from production, condition it, and provision it at the speed your application pipeline requires. Synthetic data is the best option, all orchestrated through your application pipeline."

An individual at the back of the room stood up, waved for attention, and said, "That's great to hear, but for that to happen you have to know your data and how it is stored, the database, data models, and so on. And there's no one on our team who knows that. They all left the company a long time ago. It's just not possible for us!"

"I feel your pain," Wahid said, without losing any of his positive attitude. "Every enterprise organization I talk to suffers from the same issue. The subject matter experts who originally set up our legacy applications are long gone, and no one wants to touch those applications because of the fear of breaking them. I get it.

"But for you to tackle the test data management bottleneck, you have to take the first step. And it will be small. But then you will take the next one, and the next one. For us, we decided we wanted to continue bringing down existing reservations from production for a few complex scenarios, which we thought we weren't able to replicate ourselves synthetically. For the traditional scenarios, which were basic reservations for domestic, international, multi-segment, and other types of reservations that we get daily, we decided to try to synthetically generate the data," Wahid explained with a naughty smile on his face.

Adam saw his friend's expression and interjected, "C'mon Wahid, we're all friends here. Give us the unfiltered version."

Wahid didn't hesitate. "Of course, we had skeptics on the team. Without us knowing, they continued bringing down a subset from production with all the scenarios we used to leverage for testing. We didn't even know. So, when we showed the team the synthetically generated reservations, which turned out to be 95 percent of what we needed for testing, the skeptics didn't want to trust it. Then, we compared the synthetically generated reservations with the similar ones from production and found they matched the required test scenarios. Still, we had to use both sets of data in parallel for three releases before everyone on the team learned to trust our synthetically generated datasets."

A man in the front row asked, "What about the more complex scenarios? Did you just continue pulling them from production?"

"Only for those three releases," Wahid replied. "Then, we decided to tackle that and synthetically generate them. That took more effort than we had imagined, as we had to learn from scratch about the data and its relationships across tables and external systems. Obviously, the people who knew about those relationships left our company a long time ago. But after another two releases, the effort paid for itself, as we were able to synthetically generate all data, on demand, through our pipeline orchestration engine, reducing the test data provisioning time from one week to a few minutes. Now . . . was it easy? Absolutely not. Was it worth it? Yes. One hundred percent. Our business never imagined we could do anything like this. When they saw we could generate

any type of data, whenever we wanted, they started asking us to support some of their experiments by generating very specific and uncommon datasets, as they were trying to reimagine how the whole travel experience should work for our customers."

* * *

"This is going pretty well," thought Leigh, looking around the room. "It's a full house. Way to go, D!"

For a moment her thoughts drifted to Alicia and her mention of the phrase "Kitty Hawk" yesterday. Did she know? If so, what did she know? Was the MidWest merger plan even true? Maybe Leigh had just misinterpreted a flipchart that was meant for something else. And if there was going to be a merger or a buyout, wouldn't it be a much bigger event than just a boardroom meeting? Well, yes, but these things must start somewhere. Just an idea being floated around, maybe?

Then, she thought about the big data breach that had happened last year at one of the country's largest wealth management firms. The senior executive sat on the story for months and then all sold their stock—or most of it—days before releasing the news of the breach to the media. Things like that happened all the time. Rules were only for those who chose to follow them. Her dad had said that. You can have all the compliance requirements and audits in the world, but there always seemed to be a separate game plan for some people, and those people always seemed to come out on top.

Her mind snagged on Jessica's comment from before: "Awayz knows more about flights than we do." It was dawning on her. This was the future for the airlines. People would be buying plane tickets from Awayz, from Amazon, from Walmart even. They were the ones with the retail experience. They were the ones with the future. It probably wouldn't even matter what name was on the side of the plane if Walmart got you and your family the best price to Disneyland. Loyalty programs—they wouldn't belong to single airlines anymore. They'd belong to whatever online outfit brokered the best deal, whether it was a seat on a plane or a camping tent or a gallon of gas.

That's why dotPlane's getting rid of its fleet! That's a sunk cost for them! Why use your own planes when you can buy from anyone? Point. Click. Vacation. No brand names in the equation.

She realized at that moment just how much another snafu like last week could ruin Renway. It was no longer just a matter-of-fact technical mess-up. It was an iceberg. People were not going to be loyal to an airline or any brand. They would go with whoever was ready, right now. Renway, she realized, should be planning for a very different future. But how can we do that when we can't get a simple cutover straight?

Then there's the merger or buyout or whatever. Bringing in another airline that is already failing. The integration between two legacy organizations that big! She had read the case study about the TransPacific–CentralAir merger. The way each operated had been different in every aspect. Operationally: like boarding from the front of the plane or the back; policies, unions, scheduling of crews—each company had done everything differently from the other. The reservation system, ticketing, loyalty programs! So many problems waiting to happen.

She realized at that moment that the need for continuous testing was about to get exponentially more urgent. They needed to get to a completely different level if MidWest was coming on board, either to accommodate them or to beat them with a better system. Some Renway jobs were at stake here.

Leigh stood up discreetly and signaled to Jessica that she was going to the back of the room to get some pizza.

"They're covering a lot of ground," Jessica whispered once they had made it to the tables and found a pizza that was still reasonably warm.

"Yes," whispered Leigh back, "it's a lot to take in, but it helps that it's mostly his own case study. I have to find a way to digest this down for the wiki."

They watched as the PowerPoint slide advanced once again.

3. Test Automation

"Big changes needed here for us!" Adam exclaimed. "The way we're doing test automation is not suited for today's agile and DevOps world. Our scripts are built for the waterfall age, and they need babysitting as they run because they're brittle. That won't work in a fully orchestrated application pipeline, because scripts will fail as a result of things not related to the application code. So, no one will trust the results. We need reliable test automation for today's world. That's the only way for us to achieve continuous testing."

Wahid added, "Spot on, D! As part of our journey to continuous testing, we learned it requires a change in the way we think about test automation and who does it. That means developers automate tests themselves and help testers by making their code testable with hooks that help more modern test automation tools be more effective, break scripts less, and run the tests faster. Testers get more technical so they can talk to developers and read their code, potentially augmenting it to make test automation more reliable.

"Automating the automation! Get scripts to be created and maintained automatically so you don't have to do it. Seventy percent of tests are still manual because GUI testing isn't maintainable! And once the scripts break people don't go back to fix them. They just go back to manual."

Wahid continued, "But one thing I would caution, based on my experience, is that it's very easy to dive into a tools conversation. And before you do that, you should analyze where the bottlenecks are in the application pipeline. For us, we had taken care of the test environments and test data bottlenecks. Test automation was the next bottleneck. What we learned is that if we automated the non-GUI tests, such as unit, component integration, and API, we would find a lot of defects before the GUI was even ready. Later, we learned that the industry calls what we did 'shift left.' Moving testing activities upstream to find defects as they happen instead of waiting for the GUI to be ready to do a more integrated test."

"Wait!" Leigh bit her tongue, but it was too late. Wahid and D were looking at her. She felt she had to continue her thought. "You mean we shouldn't run tests on the GUI?" she asked, still holding half a pizza slice in her left hand. "That wouldn't work for us. If we take our dot-com site, we need to ensure all the business logic and complex business rules around flight reservations are working as expected. We can't just *trust* that they were implemented correctly at the GUI level." Leigh instinctively checked herself. Was she being too aggressive? Not personally, of course; she did not care about that. But in terms of Renway. Was she revealing too much? No, she decided. This was how progress happened. "Have you had to deal with something similar at Awayz as you went through your continuous testing journey?"

Wahid's face broke out in a large smile. "You're Leigh, right? D has talked a lot about you! He said you might be coming."

D interjected playfully, "I only shared the bad stuff about you Leigh." Wahid looked out at the audience.

"I hope this whole presentation helps you understand what you will probably face, and I hope it eases some of your pain. My scars haven't healed yet, but I'm feeling much better than I was a year ago. In any case, to answer your question, Leigh, yes, we certainly had a lot of applications in a similar situation as the one you described. For us at Awayz we realized that by shifting left, when the GUI became ready, we set the expectation to ourselves that all defects related to logic and functionality would have been found and fixed already. That way, you don't really expect to find defects through the GUI; instead, you test user journeys across business processes to ensure users are able to achieve value as they use the application. With that frame of mind, you're continuously testing the application across all layers, from unit to integration. Obviously, that expectation we set for ourselves wasn't met at first, but over time we were able to achieve it, and when we got the GUI deployed, we had already been continuously testing everything before it. We still automate the GUI testing, but we are not reliant on it for all of our functional testing."

Wahid looked at the entire audience again and continued. "Don't underestimate what I'll say next. Test automation requires a complete overhaul to be valuable and to support the speed and quality required by users. It can't be a separate project or funding; it has to be part of the development process. Acceptance test completion, maintenance of the automated regression testing suites, and adding more tests to the regression suite, have to be part of the agile *definition of done* for each story in the backlog."

Leigh looked over at Jessica, who was taking notes. The PowerPoint screen changed again.

4. Pipeline Orchestration

"Yes," said Wahid, "this is the backbone. Everything is tied to it. This must be integrated with your automation suite. This is the next critical piece. You must understand how it works, how to interpret results, and how to make it scalable. Basically, in terms of continuous integration and continuous delivery, you're making sure that you set up your continuous testing in a manner where you integrate continuous testing attributes within a pipeline. But you validate the pipeline to make sure that, if there's an audit, you are compliant with organizational standards. Make sure it's transparent and within bounds. Think of it as an automated workflow tool that will run all of your automated tests, fully integrated with code deployment activities, as the code moves through the pipeline.

"Frankly, you cannot get to continuous testing or DevOps without the reliability and speed of a standardized and automated pipeline. The pipeline orchestration must be treated like a production application. It must be monitored, supported, and resilient. You must have team members that own it, maintain it, and teach others how to leverage it."

A hand went up in the group. Adam acknowledged the hand. The participant stood and faced the group.

"With respect," she said, "I might suggest that this is a bit of an overused term. Every time you have a new build, a new pipeline is created. When you have continuous development and continuous deployment, you need to understand where your release is. There might be multiple applications and dependencies. Look at microservices, for example. In theory, they should be independent, but in reality, there are still dependencies. There have to be. And in large enterprises there are also dependencies on other applications. The faster you release, the less is being released in each cycle. You need a pipeline, but you still have to ask, 'Where are my bottlenecks?' This is one of the key points in agile as well as DevOps—it's constant improvement to the release pipeline."

Wahid, with his calm face and humble demeanor, said, "This is definitely an area where we are all constantly learning. And what worked for me won't necessarily work for you. Even within Awayz, each team had to take a look at the pipeline templates we created and customize them to fit their individual needs."

He continued, "Leigh would probably start her journey by creating a pipeline per application. Soon, she'd realize acceleration and other benefits for each application. But then she'd also realize that, in each application pipeline, there would be a major bottleneck. That would happen when, for example, the reservation application receives a new build, and that build would progress through the reservation application pipeline until it reached the actual production deployment stage. At that point, it would need manual intervention for approvals. Leigh, correct me here if my assumption is out of line. But I would bet no one can release to production without going through a Change Advisory Board—the good ol' ITIL CAB—am I right?"

Leigh sighed, "Yes . . ."

Wahid got re-energized. "Don't feel bad; we were like that too. Most companies I talk to are like that. So, going back to the pipeline story, that build that was going through the pipeline smoothly is now stuck waiting for manual approvals. That is necessary because as you deploy a code change to something that changes the way reservations are made, that will impact the airport operations applications that actually check passengers in, gate applications, bag-tracking applications, and so on. My point is, unless you create a unified pipeline containing all applications, properly orchestrated, it will be very hard to truly eliminate all bottlenecks."

Leigh was getting anxious thinking about how this could work at Renway and whether she'd been wise to take on the task of leading the charge toward continuous testing there. Again, without actually wanting to say it, she said it out loud: "That sounds like an impossible task. It's too complex. How did you prioritize? How do we start?"

Wahid smiled again. "For that to happen, obviously, it would take time as you decompose applications to remove hard dependencies and mitigate release risk. But as you think about building your pipeline, think about the most important thing in the beginning: visibility. The pipeline would help you clearly see where the bottlenecks are, and then you'd be able to invest time and resources to eliminate those bottlenecks, one by one. There's no magic here. You just have to start. That's what worked for us."

"Very good point, thank you," said Adam. He checked his watch and advanced the slide.

"Those were the basics and what everyone talks about in conferences. My personal experience showed we can't stop there, as it wouldn't be enough for the business and your customers to realize the value they expect. We have to go beyond. And here's the next step we took." The next slide showed:

5. API Testing

"Although I briefly talked about it as I was touching on shift left and testing before the GUI, API testing is an area you need to go much deeper on, as it will enable alignment with the principle of the testing pyramid," Wahid said.

A person in the front row murmured, "Pyramid?"

Wahid continued. "Yes, the testing pyramid. Google it for more details later, but the gist of it is that it describes how organizations should strive to test as much at the unit and API levels as possible and minimize reliance on UI testing, because UI tests are so much more brittle and expensive to maintain. This is not the reality in most organizations, but it needs to be. To achieve continuous testing, you have to embrace the concept of the testing pyramid properly, strengthen unit and API testing, and reduce reliance on UI testing, especially for business-logic testing. Automated UI testing should focus on the UI functionality and visual behavior, and testers will do exploratory testing against the UI, but the component functionality should be tested thoroughly at the unit and API levels, where tests are easier and cheaper to create with high coverage and much easier to maintain. If developers aren't required to test, you will see little to no unit test coverage, and if applications don't have APIs, you can't test them. If both are true, you fall back to UI testing, but this must be addressed."

"OK," said Adam, "that makes sense. As we know, API tests that are automated tend to be more stable, which makes them more conducive to short release cycles and frequent changes. Yes?"

"Exactly!" said Wahid. "With APIs, systems tend to be more decoupled, where each layer can go independently of each other and can be tested independently. We find this to be more reliable because we can monitor at each level. It's a big foundation item in continuous delivery. Many testers don't know APIs well. As I mentioned earlier, if you're only testing from the GUI, troubleshooting becomes problematic. It takes a lot of resources to figure out where the root cause is. Is it in the GUI components, business-logic component, another API that's returning incorrect data? It's just too hard and time-consuming to get to the bottom of it if you find a defect through a GUI test. So, testers had to learn how APIs work and how to properly test them because the benefits of finding problems at that level were going to be huge."

"Thank you," said Adam.

Wahid added, "And you have to consider the maturity level. Most testers 'get' GUI-level testing, but they don't have a great understanding of APIs and what they do or how to test them.

"At Awayz, we had a third party that was making changes. It took two sprints to make changes. Then, after that came regression testing. We called it a

hardening sprint. So, if this took three sprints, then it would have taken between nine and twelve weeks before getting to production. With internally developed APIs, we found we could shorten the window by just testing the APIs. And also, don't forget that large companies often have an architectural review process that takes time. That's what we used to have too." Wahid laughed and continued, "That wasn't very DevOps-y. I'm glad we were able to convince our business and IT leaders that we had to implement automated controls so that we could kill that process. Lots of my scars are from that battle. But that's a topic for another time.

"Anyways . . . what we also learned is that as there were new APIs being built or just changes being implemented in existing APIs, we were still taking quite some time to build new API tests or to update existing tests, as we were doing it all by hand. We learned that continuous testing was pushing the boundaries of not only automated test execution, but also automated test creation. That was eye-opening for us. The ability to automatically create API tests and then run them in an automated fashion was simply too good to be true. We were able to try that out and further improve our lead time as we were no longer creating or maintaining tests by hand. We let technology do that for us. That's something for you to consider very seriously, as it's probably something you didn't even think about."

Leigh checked her watch. It was noon. Were they going to get through all ten in the next half hour? She and Jessica discreetly moved back to their seats. This is why she always sat in the back row.

On the screen, the next topic appeared.

6. Performance/Load Testing

"Now, you may have thought I was only going to talk about functional testing, right? To be honest, when we went through our journey, we only considered functional. As we got to a point where we were releasing quality code at speed, we realized the user experience was still problematic based on what we were monitoring and on feedback." Wahid looked around the room at blank stares, as if people were thinking about what it actually meant: functional versus non-functional testing. More specifically, performance, in this case.

Wahid continued. "As we got to the bottom of it, we found that while our applications were functionally good, in certain cases the user was having to wait several seconds for a response to a user action. Being in the travel business, where there are many options for the user to choose from, that is unacceptable, and it was leading to many dropouts in the user journey that we were able to root-cause to poorly performing application components. That translated to a high six-figure-dollar impact to our business in missed revenue, which clouded all the amazing progress we had made toward achieving

continuous testing on the functional side. Anyone want to guess what our next priority became?" Wahid asked the audience.

"Performance testing!" many shouted together.

"That's right, but that wasn't an easy challenge. We had to radically change the way we thought about performance testing," said Wahid. "We used to do these long-running performance tests that included workload profiles for load, stress, and endurance. That took days to run, and that's in a stable environment with lots of data. Fixes required significant effort and re-testing."

A man stood up. "My name is Bryan, and I led my performance testing CoE," he said, "and we are proud of what we accomplished. That's how we do it here, and it works great if you know what you're doing and if the teams follow our test-request process. We produce some serious metrics with all the statuses of how the application is behaving across the tech stack. It took us years to perfect our practices and tools in the performance testing CoE, and we're finally at a point where everyone trusts our results and relies on us."

Wahid took that in and thought hard on how to react. After a long pause, he said, "I totally agree that works great for certain environments. For us, however, that wasn't enough. We optimized costs and gained efficiencies when we centralized our performance-testing skills and technology, but when continuous testing became everyone's goal, that model would not scale. Our centralized team, like your CoE, has a fixed capacity. As agile teams started talking about shifting performance testing to the left, along with functional testing, our central performance-testing team simply wasn't able to support the barrage of requests for performance tests to be created and run. That happened because as you shift testing to the left, you're testing earlier in the lifecycle, with less application and infrastructure components, which means the tests are smaller by nature. But they're also much greater in volume."

Wahid paused and looked around the room. People were locked into what he was saying, as if they knew exactly where he was going next. That gave Wahid a boost of confidence, and he continued with an excitement he hadn't displayed until now.

"Just like we did with functional testing, we had to democratize performance testing. Make that capability accessible to everyone across all agile teams. That meant all developers and testers alike had to be able to create a performance test and run it by themselves, without having to send any requests to anyone. The challenge with that was having the right skills and technology within each agile team to actually do that effectively."

"That makes no sense. You can't throw away years of knowledge and experience that were gained as you created your central performance testing team," said Bryan, who was still standing.

"You're absolutely right," replied Wahid. "And what we did was the opposite of that. Recognizing the unique value of that team, we decided to turn them into an enablement team. Or a Center of Enablement for performance testing. That team became responsible for educating the agile teams on the best practices around performance testing, as that had to be shifted to the left. They defined the practices, the platform, methodologies, and tooling and packaged it up in an on-demand, self-service platform that anyone could consume and customize for their own agile team's needs. They were also available for Q&A as well as to help with proofs of concept or to kick-start performance testing within a complex technology stack that was not trivial. In other words, in addition to an enablement center, they became kind of like a SWAT team, in and out, problem solved, while others learn how to do it next time." Wahid stopped for another pause. Leigh was impressed. She felt he had hit a nerve with the audience members and had given them a path to follow.

"Okay," Wahid said, "but there's one thing missing. Anyone care to venture a guess?"

Silence.

"Hey, D, I think I've managed to get everyone to sleep during lunch time," Wahid joked.

Adam fired back, "Yeah, you have that effect on people." Everyone laughed.

"In all seriousness," Adam continued, "I think we're missing monitoring capabilities."

"You're really paying attention, huh?" Wahid said.

"I have to; I'm paying for everyone's lunch," Adam replied.

"Okay, as you all probably know, monitoring is a critical component of any performance-test execution. However, monitoring in pre-production environments is an ongoing practice and not just an ad-hoc activity that people don't really think about. If you are democratizing performance testing and people are testing earlier in the lifecycle, in many different development and test environments, you should also think about monitoring all those environments. That enables team members to clearly and quickly identify the root cause of any performance problem that is encountered in the high volume of tests that are run as you shift left. Without monitoring, when a test detects a performance issue, it's like trying to find a needle in a haystack to get it root-caused." Wahid took a breath to deliver his final pearl of wisdom.

"At Awayz, we transformed performance testing like that. It allowed us to see degradation and to troubleshoot very quickly. We were able to move from months to hours in finding and fixing the root cause. We could move away from long-running tests at the end of the cycle." Wahid concluded and looked at Adam.

The hour meeting was drifting toward the end now, and Leigh could see people getting nervous about leaving without causing offence. Work was piling up back at their desks. The information was great, as was the pizza, but the day was moving along.

"We're running a little out of time," Adam said, as if reading Leigh's thoughts. "If anyone needs to leave, please do so, and I hope you'll check the hashtag for some additional comments." Looking at his phone, he said, "I see we already have quite a few. Wahid, can you throw us a couple of fast points for the last couple of topics?"

"Sure," he said. "OK, acceptance test-driven development."

7. Acceptance Test-Driven Development and Behavior-Driven Development

"This is one of those key capabilities we had to learn from scratch. For the sake of time here, I'll just say we basically took the acceptance criteria for each user story and created tests to ensure those were met. Obviously, that doesn't cover all aspects of functional and non-functional testing we would normally think about, but it was a great way for us to keep our testing focused within the sprints. Over time, we noticed the teams would define much more detailed acceptance criteria, which required tests to also be more comprehensive. It was a learning experience for everyone on the team—product owners, developers, and testers." Wahid spoke in a humble way.

He continued. "I know I'm short on time here, but we're currently exploring behavior-driven development as a way to unify our assets under a ubiquitous language that is understood and usable by everyone on the team. That language would be Gherkin for us. As we were marching toward that path, we realized we'd be solving the collaboration problem between the team members, and everyone would speak the same language, have the same understanding of a story. But over time, we'd still have to deal with the traditional problem of maintaining a vast library of Gherkin feature files as we made changes to our stories or had new stories that impacted existing functionality."

Wahid moved to the next slide, which read:

8. Automated Test Generation

"Which brings me to this topic, which was an entirely new thing we didn't even know existed, and I wish we had more time to talk about it. But you have my contact info now; please do reach out," Wahid said.

"For us, as we realized the kind of speed we needed to achieve and the level of quality everyone expected, we tackled the issues one by one as we identified them through our value stream mapping exercise. However, what we didn't consider at first was the fact that the very activity of manually designing and writing manual tests or automated test scripts was a bottleneck in and of itself. Does anyone know about this concept I'm talking about?" Wahid asked and saw Adam pointing to his wrist watch.

"Okay, I'm getting D's mad-man look. So, let me wrap this one up. Basically, developers and testers would look at the stories for the sprint. Developers would start doing their work, while testers would start thinking about what tests would need to be created to cover those stories. Toward the end of the sprint, as the new code had already been manually tested and automatically regression tested, we'd start our effort to automate the manual tests for that sprint and eventually add them to the automated regression test suite for the next sprint." Wahid took a sip of water to catch his breath.

"The revelation to us was when we found out that instead we could automatically generate the new acceptance-level tests for manual and automated testing in the beginning of the sprint, so when developers started checking in and merging the first working pieces of their code, we could start running our tests in an automated fashion. That was a huge game-changing capability for us, because effectively it gave developers the ability to get immediate feedback on the quality of their code as they checked it into source control. And that feedback included the results of acceptance-level functional, performance, and security tests that were generated automatically and executed automatically through our pipeline orchestration engine. Think about that for a second; you've probably never seen that before, like I hadn't back then," Wahid said, full of excitement.

"With that capability we were finally able to truly achieve continuous testing because test design time was drastically reduced and writing the actual application code was what was making others wait. Can you imagine that? Coding, not testing, *coding* is the bottleneck in our SDLC. That just feels like the Twilight Zone . . . and it's awesome!" Wahid moved to the next slide.

9. Requirements Engineering

"While many teams, including my own, start the continuous testing journey by looking at enhancing development, testing, and operations practices like I've been describing so far, we have learned at Awayz that there will always be a disconnect if we don't truly include the entire group of SDLC stakeholders in the journey. And that means finding a better way to communicate and collaborate on the requirements. The way requirements are specified today

continues to generate different interpretations by product owners, developers, and testers." Wahid took one second to read the room. There were some people standing up, but not walking away. They were still hooked.

"The earlier you include requirements engineering into your continuous testing initiative, the smoother the journey will be. Team members will be working with the same understanding from the beginning, without the need to go back and forth to the product owner to clarify what was meant in a story when a test fails and the developer says it's working as designed while the tester says it's not satisfying the story acceptance criteria. This also feeds into the Acceptance Test-Driven Development and Behavior-Driven Development disciplines I talked about earlier, as well as automated test generation." Wahid looked at Adam, but his facial expression said it all: "I need to get these people back to their desks!"

Wahid sped up. "What is the first thing your product owners do as they are explaining to you what a user story is and what it has to accomplish?"

Someone who had just stood up and was walking toward the front on his way to leave said: "They usually draw a diagram to explain the process that needs to be created or changed."

"Like a flowchart?" Wahid asked.

"Yes, a flowchart is the most common type of diagram they use. Everyone 'gets' it, no need to know any special notations," the person said and stopped walking, intrigued.

Wahid continued. "Great, so you have half the battle won already. The key to better requirements engineering is to specify the requirements in a visual way, such as in a flowchart. Usually, after the product owners draw the flowchart on a whiteboard, they start typing up in a requirements management tool what that flowchart describes. What if you could stop converting the flowchart into a text-based form of representation and keep the requirements specified and managed in that flowchart? Furthermore, as I mentioned in the earlier topic, what if we used a flowchart to automatically generate all your acceptance-level manual and automated test scripts to be hooked to your pipeline orchestration engine? That essentially meant for us at Awayz that we were able to have all acceptance-level tests automatically generated and ready for developers to use on day one of the sprint. Pretty amazing . . . and eye-opening for us."

Adam stepped in and said, "Please remember to take full advantage of the hashtag, #DOSPrompt, to keep the conversation going . . ."

Wahid interrupted: "Just one last very quick topic!"

The PowerPoint slide changed to:

10. Feedback Loops

"These are critical to successful continuous testing," said Wahid. "You need dashboards in real time. They must be automatic, and the entire team must have access.

"There's no point in doing anything I talked about without getting feedback on how you're moving the needle toward accelerating your applications to the hands of your users with the utmost quality. That's how you know what the most important problem or bottleneck to tackle is. Feedback loops across the entire SDLC, not just production, should be used as a compass to help you navigate your continuous testing transformation. Thank you!" Wahid stepped aside and handed over the clicker to Adam.

"So there you have it, folks," said Adam as he started his closure. It was 12:30. "I hope you'll come back for our next session in three weeks." There was applause, and about a dozen people moved forward to ask some additional questions. It was something that had always annoyed Leigh in her experience giving presentations: some of the best questions are asked at the end, face-to-face. Some people are just not able to speak out in a crowd, even when their questions are excellent. Such a waste. She stood and caught Adam's eye for a second. The moment the connection was made, she spoke as loud as she could, "What about security testing?" The audience stopped their various conversations and looked at her, and, almost comically, all turned to look toward the podium. This was good. They wanted to hear more.

11. Security Testing

Adam looked at Wahid as if asking "Should we do this? Do you have anything?" Saving him from possible embarrassment, Wahid stepped forward and took the clicker back.

"I'm glad you asked that when you did, Leigh," he said. "It's perfect because it is so often seen as an afterthought. I even have a slide ready. If you guys have a few minutes, I'll pull it up."

The PowerPoint was still on the screen, and Wahid quickly found and projected the security slide.

"Developers aren't trained to build quality into the application code from the beginning," he started, "and that's why we're talking about continuous testing: to ensure that we help them throughout their development efforts by providing them guardrails, or immediate feedback, on whether their code is working and performing." Wahid paused and looked around the room for a reaction. Most people had stayed to hear this out.

"So, why would we think they know how to build secure application code? One that is free of vulnerabilities that can be exploited? It's not realistic to expect they know how to do that. And that is the main realization for us at Awayz. We started by implementing tools without providing education to developers and testers. In the end, we found out that they were using the tools just fine, but the security of the code hadn't improved."

"You need a specialized team to do that!" A woman in the third row spoke out loudly.

"Sounds like a fellow traveler," he joked. "That's what we thought too. But what we realized is that the specialized security testing team would just be the bottleneck, similar to the other specialized disciplines, such as functional test automation, performance testing, and others. So, what we did is we created an enablement team to put in place the security policies and best practices and implement the security guardrails in the application pipeline. The enablement team was not meant to do the work, but instead enable and educate the developers so they could to the work themselves. That way, as developers check in their code they will know whether it has vulnerabilities or not."

"However, that was just the first step. Bottom line, developers needed to know why a certain piece of code contained a vulnerability, what it actually meant, and how it could be exploited. That was the only way for the developer to know how to fix it in the right way and, more importantly, learn from it, and in the future write secure code from the beginning without having to rely on the guardrails. Education was the focus in this second step. Enablement was key."

Adam was tapping his watch in an overly theatrical manner while looking at Wahid.

"I guess I've overstayed the welcome," Wahid joked, "but I hope this gave you all a sampling of the types of disciplines that continuous testing needs."

Adam stepped in, "And I hope you will all come back for the next DOS Prompt. It looks like we still have a lot to talk about."

The audience started to move out, some in an obvious hurry.

Leigh made eye contact with Adam again and made the "I'll call you later" gesture with her hand.

She looked at all these talented, driven people and wondered how many of her own would still be at Renway once the two airlines merged. She wondered if she would still be there. There was only one person she knew she could talk to right now.

* * *

"Can I come in?" Leigh asked.

"Of course," said Alicia, "let me just save this . . ." She clicked her mouse and then closed up her laptop. Not out of security concerns, but as a gesture of respect. When you were with Alicia, you always got her full attention. "But then again," thought Leigh, "there *might* be some security concerns . . ."

She looked at Leigh. "So, what can I do for you?"

Leigh thought for a moment. "Our new initiative to introduce this new approach to testing has become an even greater imperative, given current developments," she said cryptically. She paused, looking for any reaction from Alicia. A raise of an eyebrow? A slight smile? A twitch of an eyelid? Nothing. Leigh pressed on. "I have been working with teams from Test, Development, and Ops to identify the problems and bottlenecks that are holding us back from building continuous testing into the entire lifecycle and trying to show them how testing and quality have to be built in from the very start. Your presence at the meetings was very helpful by the way."

"So, what do you envision next?" Alicia asked.

"Well," Leigh replied, "you can't have action without a plan, and you can't have a good plan without facts. So, we're building a wiki in order to have an ongoing string of definitions that support continuous testing. It's early days. It's still got to have more bones, more delineations. How we do quality in development, in testing pre-, mid- and post-deployment, and ongoing production testing, that sort of thing."

"And you could have told me all of that in an email," said Alicia, "as much as I enjoy the visit."

Leigh pressed further. "I . . . um . . . discovered something the other day that I think might have a significant impact on our efforts, making them a lot more urgent, as it were . . ." Again, she paused for a reaction. Again, nothing. Alicia was a cool customer. "It has to do with being . . . um . . . you know, first in flight?"

"Yes, I'm familiar with that phrase," she said, a little sardonically, "but I don't think it would work as a tagline for Renway. I think the state of Ohio uses it. A Wright brothers thing, I believe."

Leigh still couldn't tell. Was Alicia dancing around the Kitty Hawk thing, or was Leigh just projecting?

"Actually, it's North Carolina," Leigh replied, "and Ohio uses 'Birthplace of Aviation,' but there's a lot of debate going on over who gets the glory of the Wright brothers. It's all still up in the air, so to speak."

Leigh realized she was tap dancing here, but she didn't want to let go of the Kitty Hawk reference. Alicia was so hard to read.

"Anyway . . . I found out . . . I found out . . ." Leigh hesitated.

Alicia looked at her expectantly. Time seemed to slow right down.

She folded! She couldn't do it. It was confidential information. She had been in an executive meeting room, and she had looked under a covered flipchart page. Surely that would constitute espionage? Grounds for dismissal? Escorted off the property right then and there? Alicia hadn't given her one flicker of recognition or any hint that the secret would be safe with her if spilled.

"I found out that dotPlane is flying to Russia for the World Cup. I mean, not their own planes. In fact, they're planning on selling many of them off next year, but for the World Cup they've hooked up with DelToro in Spain and that low-budget charter out of Manchester, SwiftUK.

"They just jumped on this, and if you look at their reservations site, look at their app—they've got it all set up and working beautifully. They are even doing changes and cancellations on the app. They know the 80/20 balance is reversing, and that most people will be doing this all on the app pretty soon.

"This World Cup thing isn't just a one-off. This is their demonstration of the future. In dog terms, they are coming up and peeing on our tree."

Finally, Alicia raised an eyebrow, but it was probably just for the overly graphic metaphor. "I confess, I wasn't aware they had moved quite so fast," she said. "So, how exactly does this tie back to your 'first in flight' message?"

Leigh struggled to find an answer. "Well, you know, dotPlane is hooking up with new partners—a new alliance. They're breaking new ground in becoming a transatlantic. They're getting there first." She hoped that wasn't a feeble answer.

"Well," Alicia began, "I'll let you in on a little secret. I was planning to announce it to the senior team—you, Arthur, Giles, and so on—sometime next month, but here it is."

Leigh braced herself. She was right! No more guilty secret!

"I have been given a second position," said Alicia, "anointed, as it were, to CDO, Chief Digital Officer, along with retaining my current CIO position. Hooray! Two hats! I think the goal is to blend them eventually. What this means for us as a team is there is going to be a greater focus on revenue generation. It's not just going to be about building code just to keep the lights on.

"Basically, I have been asked to take the ecommerce side of the house. The executives and the Board think this is the next logical step forward. That gives us the opportunity to take what we have today to the next level. Participate more fully in the global marketplace—that was their explanation for it. It's a whole different way of thinking. Having technological answers for business needs, as opposed to letting technology needs limit the business scope. Much more entrepreneurial."

"So that means . . . ?" asked Leigh.

"You, we, will all have a lot more work to do to make sure this hits the market in perfect shape very soon. If we are going to do this, and at the same time beat dotPlane at its own game, we might have to think about doubling up our effort and shortening the timelines. This means getting everyone on board and in place a lot faster. Who knows what else might be coming down the pipeline in the near future?"

Leigh stood up and thanked her for her time. Her head was spinning. And now she had to go back to the airport to meet Alex.

CHAPTER

6

Accelerate into the Curves

"We're in Orville & Wilbur's near Gate E, by the window," said Alex's text message. The restaurants always seemed to be better on the inside of the airport, once through security. Probably because of the captive audience, Leigh mused. Once you're through, you have time to kill and mouths to feed.

She found Alex and Sylvia quickly, seated at one of the tables overlooking the apron and the runway beyond. In the distance was the skyline of Boston. Alex had a beer in front of him. Sylvia had mineral water. As she shook hands, Sylvia said, "Don't let the mineral water fool you. I'd be at least one drink in by now if I was off the clock, but I'm not. Gotta go back upstairs for a while after this." She pointed backward toward the security area. Aurora Airlines had their main building across the street. "Upstairs" referred to her 10th floor corner office.

"I'm interested in what you're doing," she said with her distinctive, no-nonsense southern drawl. "I got the scuttlebutt already. Our community is not that huge. Good for you for trying to teach an old dog new tricks! Pushing continuous testing into a company that moves planes around the sky is not easy. Believe me. I've fought many a battle of the same type. Don't forget, I'm thirty years in this business. I started out as a programmer, and to this day I'm just trying to keep up with what's changing and then spending the rest of my time trying to get everyone else to understand what needs to be done versus what they think they *want* done."

"Well, I'm very honored that you wanted to see me," said Leigh.

J. Scheaffer et al., *The Kitty Hawk Venture,* https://doi.org/10.1007/978-1-4842-3661-1_6

"Don't be," said Sylvia. "This is not a charity thing. You're good at what you do. I saw you speak in Barcelona last year on transportation and intelligent cities. You're a bright spark, Ms. Freemark, and I'm doing myself a favor by getting to know you. Whether you end up as an ally or an enemy, it's better I know you."

"Understood," said Leigh.

"Basically, what you *are* doing, trying to build continuous testing into your airline, is a great thing. It's no secret that some—not all—but some other airlines have started down that path. So, in the interest of a little *quid pro quo*, I'll share with you a little of what I know about the business."

"*Quid pro quo?*" echoed Leigh. "What do I have that you would want?"

"I didn't get to my position by not knowing my battleground," said Sylvia. "That's *The Art of War*, you know? Sun Tzu? Just as relevant today as it was three thousand years ago, when his IT department was a guy with a brush and a bottle of ink."

Leigh was aware that Sylvia not only spoke Mandarin fluently, but actually could write some too. She likely had such brushes in her office.

"What I would like to know," continued Sylvia, "is what you were doing with four members of AbLuft's exceedingly well-tailored executive team yesterday. And don't tell me one of them's your brother, hun. That's not gonna fly."

Leigh paused to think. If she told Sylvia the reason for the meeting, it could easily get back to Bob Cartwright, Helen Humphries, and the rest of the Renway brass. Gossip and bad news always travel incredibly fast. She had been acting outside her position at the company and, in essence, sharing confidential information, even if it was hypothetical and off the record. That could mean immediate dismissal. At the same time, this off-the-record chat was about industry innovation. To tell Sylvia about it meant giving the idea away to a competitor twenty times Renway's size. But if she made something up, Sylvia—and Renway—would find out sooner or later. And the lie itself would pale in comparison to the lack of respect it showed to one of the industry's most influential executives. The four people at AbLuft would have no reason to cover for her either. The cat was coming out of the bag. Or, more precisely, she no longer had a bag to keep the cat in.

Once again, everything around her seemed to be slowing to an agonizing crawl.

"Time to accelerate into the curve," she thought.

"I was scouting for an airline to connect with for the World Cup," Leigh admitted. "dotPlane is already doing this, and I know they are onto something much bigger than soccer. They see a future for the airlines that I think most executives cannot grasp. An airline business doesn't have to have planes any more than Uber or Airbnb has to have anything of anything. It's becoming a

service business, and our section, IT, is going to drive that further. I wanted to strike a first alliance that I could bring to Helen Humphries, so she could feel it was her idea, so that she could take it to Bob Cartwright and make it feel like it was his idea."

"Very astute," remarked Sylvia. "We set up our World Cup connections and secured our landing spots about five years ago, just so you know. But that doesn't take away from the foresight in what you're doing. I would be the first to agree that just because it took us five years this time, that does not mean it will take five years next time. dotPlane is proof of that. In fact, any company that does expect it to take five years, decade in and decade out, cannot expect to survive."

A waiter stopped by to clear the glasses.

"What'll you have?" Sylvia asked. "A beer? And Alex, how about you? I'll have a Grey Goose straight, no ice."

They both looked at her in surprise.

She shrugged. "I'm having fun," she said. "Besides, vodka doesn't stay on your breath. My brother is an attorney—a barrister or solicitor or something, actually—in England of all places. They drink a lot during business hours over there. His firm has an anti-vodka regulation for the exact same purpose. Their reasoning is, if you're going to drink on your lunch hour, choose something that people can smell on you afterward. 'That way our clients will take heart knowing you're drunk and not just stupid.'"

Leigh and Alex both laughed.

"I'll have to bring that up at our next strategy meeting," he said.

The drinks arrived quickly.

"OK," said Sylvia, "to the future of the airline industry, and our place in it."

They clinked glasses.

"So, to business," Sylvia said.

"First off, let's look at the customer. Whether you're talking Joe Average consumer or a B2B industry client—they have a really low tolerance for poor experiences. They turn off applications that simply don't work, and they don't come back. There's a need for immediate gratification, and they have elevated expectations, which means on our end the need for efficient and effective testing has dramatically increased. We can no longer tolerate any outage.

"You have to look at the role of testing on a wide scale. We're in the catbird seat. We have to provide testing services to all of the departments, mainly because all these departments don't know what the other departments are doing. When I went around introducing the concept of DevOps just last year,

I was surprised at the basic questions I got. So many people out there are still working with very little knowledge or experience *or comfort* beyond waterfall, or just the basic aspects of agile.

"There was a time," she continued, "in testing when 80 percent was good enough. By the time customers identified the bugs, they would have already been fixed in the background. It was kind of a *laissez-faire* approach."

She smiled. "Now I know you already know that, and *you* know it will take a lot of conversations and meetings to help your people understand that this increased speed needs to be part of the application-development persona.

"First," she said, "let's look at one of the features of continuous testing that is sometimes overlooked. I call it 'planning to get hit by a bus.' It starts in development. When you code for a feature, there should be release notes in that code so anyone can read it. So, if you were run over by a bus one day, it wouldn't affect the SDLC, because there is sufficient clarity and organization that another competent engineer could take over. It's a standard component of project management. Create a plan that is good enough for someone else to take over."

"But doesn't that put you in jeopardy of becoming redundant?" asked Leigh.

"Sure, it does," replied Sylvia, "that's the point. My very first boss, back when I was a programmer in manufacturing, talked about roles and responsibilities. He said to me, 'The sign of a good leader is to work yourself out of a job.' At first, I thought he was crazy, but over time, I realized the wisdom in that statement. You're not so much working yourself out of *all jobs*; you just have to work yourself out of that one and move into a different one. You create an environment that is self-sustainable so they don't need you there anymore. I consider that my job now."

"So, how does that affect Leigh?" asked Alex.

"It's what you both should pay close attention to," she replied. "There should no longer be a need for a QA function. You must evolve from 'QA' to quality engineering as a Center of Enablement. A place that enables different application teams by building best practices and technology into a platform that is easily consumable by everyone. A place where you build into the software the knowledge of how to test, deploy, monitor, and even heal itself so that application teams can leverage that within their scrums without having to depend on external staff.

"The Center of Enablement helps them get started, but the scrum team is the owner of their own destiny. That approach enables modern practices and technology to be scaled across the organization in a more self-sustainable way, causing a 'pull effect,' meaning teams will look for the CoE's guidance, as opposed to the 'push effect' from the days before agile, where the Center

of Excellence tried to impose on everyone their own practices, tools, methodologies, and staff—a true bottleneck that can't exist in today's world."

"So, how have you seen continuous testing regarding continuous delivery and DevOps?" asked Leigh.

"Continuous testing is a critical component of enabling DevOps," Sylvia replied. "As you're setting up your continuous integration process, you need to think about how you will test it to ensure that, if the code is to be integrated into the source code repository, it will do what it was supposed to do and will not cause adverse effects in other parts of the application. That's why your objective should be to embed everything into continuous integration, including infrastructure, test data generation and management, functional testing, performance testing, security testing, and so on. For that to happen, you have to treat everything as code so you can control it through CI in a standard and scalable way. The whole concept of continuous integration and continuous builds—you can use for anything. Like a sprint dedicated solely to technical debt.

"You see, part of the problem is your people. By and large they are order takers. But that's not the way we are going to do it in the future. Infrastructure needs to be embedded within that delivery lifecycle. We can't wait days or weeks to get environments built up, only to fight about who gets access to them. We need ephemeral infrastructure created on demand, and when we are done with it, it goes away.

"Test environments should be able to be defined as code as part of the application state that is stored in a versioned repository like GitHub. The orchestration engine drives automated provisioning of those environments when they are needed, whether in CI, CD, or *ad hoc* testing contexts.

"So, instead of me putting in a request in a nice lengthy form to spin up 50 virtual machines, which will be fulfilled by another group—no—that request should be automatically fulfilled based on the details I put into that request. And, better yet, the information I provide in the request should be parsed into code, which would then trigger the automated provisioning of the 50 virtual machines with the configuration that I have specified, without a human being having to intervene. Fully automated. But someone needs to build that platform, and that's the role of a Center of Enablement. They must set this up once so that multiple teams can just leverage this capability on demand. And with speed. No delays. No bottlenecks. No more 'I'll get to it when I have time.'"

"It definitely highlights an 'operations' mentality," said Alex.

"Does it?" Leigh replied. "I think it highlights a 'traditional QA tester' mentality, not an Ops one. The Ops mentality is more about preventing or controlling change that has the potential to create operational instability."

Sylvia continued. "There are times I will hold a workshop and it just becomes a venting session. I will have a developer shouting that he's underwater and that he needs to have continuous testing embedded in his team. But my point would be, as you move toward full continuous testing, you shouldn't even need to have a tester there. What you need is to set up a DevOps organization to hire additional more well-rounded engineers including application developers that can wear many hats—like maybe a matrixed organization. These are the conversations you need to be having. But remember, with any matrixed organization there will be the looming threat of job insecurity. Like you said, people aren't going to look kindly at a revolution that appears to make them unnecessary."

"So, the rank-and-file mentality about job protection continues to stand in the way of change," said Alex.

"Always," said Sylvia. "It never ends. When cars were introduced in the 1910s there were people who would heckle drivers, especially when they ran out of gas or blew out their tires. 'Get a horse!' they'd shout. There were movements to ban the automobile completely out of fear that bank robbers could outrun mounted police. It's always the way. People only know what they know, and they don't like the uncertainty of change. Developers and testers are just the same."

Leigh thought of Arthur and Owen, and the conversations they'd had, and the ones yet to come.

Sylvia added, "It's not just the frontline folks, you understand. It's management, too. I've been trying to push DevOps for three years, trying to introduce it to my VPs. But they didn't know how to do it. They were constantly debating, 'Who's going to run it? How will it get run?' They were treating it as an organizational issue."

She continued, "And then you get shift left, getting developers to embed testing into the code and CI. Frankly, with developers, the way they approach coding tests, at least until they are trained properly in how to test, tends to be quite poor, especially if they lack a means of mature evaluation of quality in development-management practices. App development teams must have much more accountability. Shift left is all about the developers' taking on more responsibility and being accountable for the code. If you spend an extra day building excellent-quality code, you can save weeks on the backend. But if it winds up moving into pre-production, then the problems only get worse."

They paused for a moment and looked around. Just outside the entrance to the bar, a small Segway-shaped robot whizzed past, pulling a small trailer. Leigh thought it looked terribly familiar. Sure enough, she could make out the inverted rocket fork with the flaming tines. There was obviously a Veneto Burrito franchise in the food court, and it was making a direct-to-seat delivery. "Incredible," she thought.

"Cute little thing, isn't it?" asked Sylvia.

"You don't know the half of it," said Leigh, and showed Sylvia and Alex the Veneto Burrito app on her phone. Across the top of the screen it said, "Now Serving! Vegan and Halal!"

"This was just an idea three days ago," she said. "I've been talking to the CEO. He's my former boss."

"Just think about the divergence of customer loyalty for a moment," Sylvia said. "What you're facing—what we're all facing—is a commoditization of the airline business. This is Amazon disrupting the system once again. Once Amazon starts selling seats, they're gonna leverage airlines the way they do with FedEx and UPS. The customer isn't gonna care who delivers the package, as long as it gets there. It throws loyalty out the window. The name on the plane no longer matters."

"So, is it all over, then?" asked Alex.

"Only if you want it to be. Only if you play old school. The thing that Amazon cannot deliver well is an experience. They are basically a two-day and same-day shipping service enabled by a logistics backend that differentiates them on quality-of-service metrics. And then if you go to the Uber or AirBnB level, you're just dealing with brokers.

"But not so long ago there was a luxury car company—I can't remember which one it was right now, but they took a different tack against the customer's roving eye. They did everything they could to stop customers from wanting to look at other competing brands. They would deliver a test-drive model to your house, they would invite you to fancy soirées at the dealership. They supplied all kinds of nice trinkets—umbrellas, highway safety kits, the works. They took the opposite tack to avoid getting lost in a swamped marketplace. They worked to become part of the customer's life.

"So now, think about the average customer today. Their home is wired with unlimited internet. Everyone, including the dog, is face down on their phone or tablet. And now, the big players are introducing variations on Siri—a listening post in the living room that can do everything from turning on the A/C to finding and playing a Willie Nelson song for you.

"So, what happens when a certain airline, Renway, for example, buddies up to the family, maybe delivering tunes or information or ideas through their home information device—not ham-fisted vacation-deal ads, but the types of media or engagement that connects directly with that individual family member? That's where it starts. Curating a presence that stops the customer from wanting to look at other airlines. You're it. You're their buddy. You don't try to out-Amazon Amazon in terms of volume and price, but you do it through personalization of the customer experience.

"But even that is really just chatbots out to own the experience of travel booking. The real threat here is fundamentally more disruptive than this. Brokering, like Expedia or Amazon might do, is not really disruptive. What would be disruptive is a marketplace where flight operators bid against customer demand for flights. Think eBay for flights. And that, my friend, needs substantial software and culture evolution."

"They have really all become 800-pound gorillas, haven't they?" said Leigh.

"Oh, they're more than that, hun," said Sylvia. "They're planets. And we orbit them now. And you want to talk continuous testing? If you realize something is coming, like a home-based relationship app that is different for every customer, it's best to be first out of the gate. To do that you must have your stuff together to do what Amazon does so well. Do you think they wait for environments or spend weeks testing after development?"

Alex echoed, "Without a continuous testing framework implemented, you can't do this."

"Correct," said Sylvia, "and if you don't do this, you cannot compete. Cloud becomes an enormous success factor here. You can't do any of this with on-prem systems. You cannot work inside legacy. We're talking transaction speeds of milliseconds. Like what Uber did to taxis, or Uber trucking is doing to FedEx."

Leigh spoke up. "An Amazon Prime loyalty program. It doesn't have to be airline specific. It's Amazon that gets the glory."

"An airplane seat or camping equipment—it's all the same," said Sylvia. "But—and there's a big 'but' here—any commoditizer of airline services still needs to know where the pilots are, where the planes are, and the logistics behind it all. Amazon or whoever will still need this data. And they will turn to the airlines or organizations that can deliver that immediately. Companies like Amazon—and they are not the only ones, by the way—have turned commerce into a by-the-second art form."

"Cool! Fire trucks!"

The excited chattering of a group of six- and seven-year-old kids broke the conversation's momentum. They were pressing themselves up against the floor-to-ceiling glass of the departure-lounge windows overlooking the gates and the apron. Leigh, Sylvia, and Alex turned in their chairs to see what was going on.

There was indeed a forest of red flashing strobes congregating around a plane that was sitting in the middle of the apron with the tug that had just pushed it back from the gate still connected. Three of the monstrous fire trucks were joined by airport security cars and Homeland Security SUVs to form a dizzying nightclub scenario of flashing lights. No smoke or flames were visible,

no slides had been deployed, and no foam was being sprayed. The plane just stood there, mute and immobile. Again, Leigh thought of the sea. The time she had helped save a whale that had beached itself on Duxbury Beach. It just lay there, expecting to die, or expecting to be saved or something. No struggle, no desperate attempts to turn itself around. It just stayed there silent and patient and still.

The plane outside did the same, and Leigh knew the passengers inside were doing likewise. Sitting still, metering their patience, and balancing it against that creeping fear that something wasn't right. They watched it do nothing for a minute more. The fire trucks, too, stayed at a distance. No one seemed to be getting out or moving around.

Alex casually fished a pair of field glasses from his briefcase and scanned the plane. The tail was difficult to read from this angle, as the plane was still facing the gates.

He pressed an app on his phone and put his eye up to the camera for a quick retinal scan.

"Alert level 2," he said. "Nothing huge. Just a problem between the plane and control. The plane's computers cannot confirm the correct sequencing of ignition for engine 2. It could be a physical failure, or it could be just a bug. Alert 2 means the emergency crews deploy, but they stand back.

"Who is it?" asked Sylvia.

"MidWest," he said.

Leigh felt a hole open in the pit of her stomach. "Not MidWest!" she protested silently to herself. She realized such a sentiment was selfish and a little heartless given all the passengers on board. But still. Why MidWest? Such a public failure!

"So, if it's a software issue, why all the emergency trucks?" she asked.

"Because that's not confirmed yet. It could be any number of problems up to and including terrorism. Or maybe someone forgot to turn a switch. The point is, it's currently a disabled plane, full of people and fuel, sitting in a live traffic lane. They won't bring it back to the gate until they know what the problem is, and even if they freeze all the traffic around it, there is always a risk of it getting clipped."

Leigh pictured the news footage to come. Mechanical problems and delays happen all the time. It might not make the six o'clock news tonight, but it might show up as B-roll as soon as some Wall Street analyst looks to pick apart the Renway–MidWest merger on Bloomberg or MSNBC.

"That's an interesting app you have there," said Sylvia, turning away from the flashing lights, "and not everyone carries field glasses in their briefcase. Remind me again, where do you work?"

Alex put down his phone. "I used to work for Fugue Air out of Dublin. Five years as head of development. Then, I got headhunted into Aspecta1 AG, a Swiss company, and I work for them here in Boston. Air traffic control and navigation software. It's an industry that has some real parallels to what Leigh is facing, especially regarding legacy systems. Aspecta1 is turning air traffic control, logistics, and navigation into an SaaS economy, where the testing and service is delivered by subscription."

"And is that what you do?" asked Sylvia, genuinely interested.

"I'm in a sort of skunkworks shop," he said. "We're building an Ethereum-based blockchain to make flight-log data safe from alteration and forgery. We're being underwritten by a large French insurance company that is interested in using the same code base to protect their flight-delay insurance products through smart contracts."

"Whew!" said Sylvia, "that went right over my head. I mean, I've heard of blockchain and all, but I am amazed at just how fast these companies—these people—come up with new and better ways to do things."

"Anyway," she continued. "The World Cup. We see these things happen all the time. We see a huge spike in bookings to a particular location. We monitor that type of thing closely to ensure we have the bandwidth. You have to think about a spike in call-center volumes. Even our crew apps. All of these have to be fine-tuned to accommodate a huge surge in volume."

She checked her phone. "I really do have to get back to the asylum," she said, "but if I can give you a takeaway, it would be this—two things, actually: 1) work on a clear definition of what continuous testing is—where it happens, how it happens, why it happens, and who makes it happen; and 2) get up to speed on change management, both to applications and to employees. You're going to need to help usher some of your people over their own wall—they're going to defend their status quo because spreading out testing along the line is going to shake up everybody's job description and make a whole lot of people nervous about their future. As much as you have to focus on the technology side of things, they're gonna have to realize the jobs they had aren't going to stay that way for much longer. And that's not a threat. That's just the news."

"Anyway," she said, standing up, "it's been a real slice. Leigh, thank you for coming out to see me. I'll be watching your progress with interest. Alex, it's been a pleasure also. I'm sure we'll meet again."

As she departed, Leigh and Alex both turned back to look out the window, where the MidWest plane was still sitting there, motionless.

* * *

She wanted to get home to Casey and Copernicus. She cued up a text message that she used so often it was saved as a keyboard shortcut: "H1." "Will be

home in an hour—heart emoji, dog emoji." She then pointed the Porsche toward Thornbury. First, she needed to see her dad.

She parked in the driveway of her childhood home and went inside. She had texted ahead—keyboard shortcut D1: "Can I drop by for a few minutes?"—and Warren was waiting. "I have cold beer and hot coffee. Which would you like, Bru?"

She opted for coffee. "You were right about the people," she said. "They really seem scared. They don't seem to want to move ahead. This business is changing, and we can't afford to stay behind, which is where we already are."

Warren nodded. "Change is seldom welcomed when it is foisted upon people, Bru," he said. "It's an intrusion, and that triggers fear. Think about Mr. Spock and Captain Kirk," he said.

"Um . . . you lost me there, Dad," she replied.

"Well, you could use any one of a thousand examples," he said, "from what's on TV now, right back to the ancient Greeks. Drama and comedy are reflections of human existence," he said, "and much of that has to do with our two sides, logic and emotion. In *Star Trek*, Captain Kirk was emotion and Spock was logic. The same for Picard and Data. Or what's the one you watch now? *Big Bang*? Look at Sheldon and er . . ."

"Penny."

"Penny. Again, logic and emotion, trying to balance out, trying to understand each other." His hands mimicked the balancing of a scale. Leigh stared at him intently over her coffee cup, trying to make the connection.

"What all of these TV shows are demonstrating, Bru, is the two sides of a person. Logic and emotion. But the key fact of the matter is, emotion always wins, and the strongest emotion of all is fear. So, that means everyone interprets everything they experience with fear first. When they're faced with change, people seldom ever think, 'What will I get from this?' They're more likely to say, 'What am I going to lose?' It's only people like you, who are driving a change, who see the benefits. To everyone else, it's just a scary fog.

"Everyone listens to the same radio station, Bru, WIIFM—what's in it for me—all worries all the time. No matter what kind of change you're talking about, if you want to lead people, if you want them to follow you, you have to get their emotional side in love with you as their leader. And to get that, to get people to hear you, you have to deliver enough fact—enough logic—to rise up and meet the level of fear they feel and neutralize it.

"When I was trying to get the brass at Mainland Technologies to listen to me about changing up an inspection process or retooling the actual manufacturing, the first thing out of their mouths—every—single—time—was 'How much is this going to cost us?' It was never 'How much will this make us?' or 'How will

this improve us?' or 'Good thinking, Warren.' No, it was always the fear factor. 'How will this hurt us?'"

Leigh sat quietly, taking this all in. She had never felt the fear he was describing. She had always just done things. It was hard to understand why people would not want to just go for it. Accelerate into the curves.

Warren smiled. He was reading his daughter's face.

"OK," he said, "let's take a scenario we both know very well. A plane full of people. Look at the pre-flight safety demonstration. Do you know why the flight attendants go through that performance on every flight, other than the fact that it's the law? Do you know why they do all the gestures? With the fake seat belt and the oxygen mask? Because it lets people know what to do. Not think or feel but *know*. Their worries about safety—their fears—are matched and neutralized by a tangible knowledge of what to do. How to hold the oxygen mask, how to pull the little tube. When they tell the person sitting by an exit door how to operate the door, it's not just for that person's comfort; it's for the comfort of everyone else around them who now know there's someone close by who can help them."

He looked directly at her. "You gotta bring the facts up to meet their fear, Bru. They need to know it's going to be ok, but they need data to prove it."

Leigh smiled. "So, maybe I should stand at the front of the room and do a pre–continuous testing deployment safety demonstration, including all the emergency exits—two at the front and two in the rear of the boardroom."

He smiled. "Actually, that would be kind of fun," he said, "and then you could hand out safety cards with those pictures of cartoon people going down the slides with their shoes off and their life jackets on. Nothing gives people something to hold on to emotionally better than something physical they can actually hold on to."

"There might be a bit much info for a seat-back instruction sheet," said Leigh, pausing for a moment. Her eyes lit up. "But the other thing all planes have is a flight plan, so that everyone knows where they're going and how they're going to get there. My god, Dad," Leigh exclaimed, "you've just given me a fantastic idea."

CHAPTER

7

Where Am I?

"I saw your droid at the airport," Leigh typed into her phone.

Almost immediately, Steve Fibonacci wrote back.

"His name is Dennis. Hey! I will be on a delivery at the Perlman Building, two blocks up from you. Let's have coffee. 10:30? There's a place on street level right there. You have the time?"

Leigh checked her watch. 9:05. "Yes. Perfect."

Alicia appeared in her doorway. She crooked her index finger in a "come here" gesture and then used the same finger to point repeatedly down the hall. Apparently, there was an impromptu meeting about to start. Leigh picked up her laptop and headed down the hall. Alicia hadn't waited for her.

In the dungeon meeting room there stood someone who rarely visited the windowless depths: Helen Humphries. She was there, along with two well-tailored people who looked like they were straight from Central Casting, Legal and Corporate Dramas Division. Arthur, Owen, and Giles were already there. There was no pizza.

"There has been substantial public blowback over the cutover fiasco," she said, without waiting for introductions or preliminaries. "People and the media are questioning the safety of our airline, and there is already some sort of social media campaign up and running."

There were murmurs of incredulity from the group, but no one wanted to interrupt her. She continued: "I know that most of you would seek to dismiss the cutover as a routine procedure gone temporarily wrong, no big deal, to be

J. Scheaffer et al., *The Kitty Hawk Venture,* https://doi.org/10.1007/978-1-4842-3661-1_7

fixed in the fullness of time, that sort of thing. Except it is a big deal. People are making a lot out of this, especially our competitors."

She pointedly looked at every person in turn. "My role demands that I understand every area of operations in this company. It appears I need to fill in some gaps in my knowledge with regard to software development, as I had expected a routine upgrade to be routine. I undertook some, shall we say, independent research, and I now understand a major reason for the failure has to do with the fact that it had not been sufficiently tested. And if there had been sufficient testing earlier in the lifecycle, the fault would have been found and fixed. Am I right to assume that?"

There was silence for a few seconds, and then Arthur spoke. "We did it all correctly, Helen," he said. "We had a phased code and a phased rollout."

"But it wasn't correct, was it, Arthur?" Helen shot back. "It didn't work. And the problem with that is, when a line of code stops an airline in its tracks, that reverberates all the way through the company. It causes fear. People stop buying seats on our planes. When they do that, we have to cut flights. And when that snowballs, the next thing we have to cut is jobs, first inside the planes and then outside. There is nothing about an airline that can be laughed off or buried. Your software is at the heart of everything we do. Literally. When you stop, we all stop."

Arthur opened his mouth to speak. She raised her hand to stop him.

"Mr. Fergus," she stared directly at him, "you are a highly experienced software specialist. I recognize that. But I will not hear you say 'This is how we have always done it.' Do not use those words, sir. When I shared the report of the cutover fiasco with my external consultants, do you know what their reaction was? One said we were stuck in the past. The other called you and your team just plain dumb. That latter comment is not an epithet I take pride in."

She turned to one of her well-dressed henchmen.

"Mr. Severen?"

Severen stepped forward and touched the screen of his tablet. He adjusted his glasses and paused to read for a few seconds, then he looked up over his glasses to address the group.

"My name is Howard Severen. I am from the firm of McManus Steffens LLP. Our investigation to date shows that in the course of the cutover failure, IT Operations called the Apps Development team to the war room to do code fixes on-the-fly directly in Production, because . . ." he paused to read his notes directly off the tablet screen again, "quote, 'the Operations teams don't know how the app code works; they just run deployment scripts and do minor adjustments on the apps, mostly around configs and not app code logic,' unquote."

He looked quickly around the room and then returned to his notes. "In addition, QA, who should have caught the problem before the release, quote, 'always demonstrate they've tested everything and then suggest it was a problem related to environments, configurations, or just was not specified in the application requirements, meaning they wouldn't know they missed something in the code,' unquote."

Leigh felt her face flush with shock, anger, and embarrassment. Who the hell was this guy? Where did he get this information? These quotes? Who had he talked to?

Helen Humphries broke in. "I am not going to bore you with the entire report quite yet," she said. "Suffice to say, we seem to have uncovered some rotten plankwork in our little ship."

Leigh felt a touch on her arm. It was Jessica, who silently handed Leigh her own tablet. On the screen was the website of McManus Steffens LLP. Consultants. Wonderful. The home page showed the entire team all standing side-by-side on a wide staircase somewhere, all in blue power suits, looking like that type of law firm that vigorously practices high-profile corporate litigation from gleaming high-rise offices. They were all very good looking. The one in the center held an aggressive posture that looked like he had forgotten to remove the hanger from his suit. Leigh scrolled to the bottom. Areas of practice included business consulting, forensic accounting, and mergers and acquisitions. Hmmm.

Severen was about to speak again.

"Scenario," he said, cryptically. "Someone erroneously removes a reference to a database in an API. No one catches it because they do manual deployment. Arthur's development team packages it up, throws it to Owen, who follows the deployment instructions from Arthur's team but does not have the power to open the code. Owen's team's job is to deploy what Arthur sent them. Arthur likes everything to be manual. Traditional. But maybe Arthur gave Owen a package with some serious issues. This only happens when things are manually done."

Leigh felt her stomach drop. Such finger-pointing was truly unwarranted, wasn't it? These guys were playing hardball. She could tell Arthur was livid.

"Who's your mole, Helen?" he spluttered. "Who gave you this? Was it Giles?"

Helen Humphries merely looked straight at him. She did not respond. Giles watched him without a hint of expression.

"Giles, the British rock star," he continued. "Senior Director of Mobile Applications Development. When everyone else gets their budgets cut, his is increased. He gets a direct line to the CMO because everything his people do is connected to revenue. Mr. Agile. Mr. Bleeding Edge. Well, I'm sorry, but

bleeding edge won't work around here, because we have a lot of mainframes and legacy applications. Our APIs are not broken down into microservices. We're not there, and we can't be."

Helen Humphries was unflappable. She looked at Leigh.

"Ms. Freemark, what do *you* think?"

Leigh tried to collect her thoughts. She felt she had just witnessed Arthur's resignation speech. "I think Arthur and Owen are best qualified to answer these types of questions. They're the ones in the war room trying to roll back the changes."

"Unacceptable," replied Helen. "You're the Senior Director of Quality Assurance. Do you think this rollout has the level of quality anyone would expect?"

Leigh felt like she was on a rodeo bull. If she could get through these next eight seconds without being interrupted, she could solidify everything she had been working on through this long week, and possibly launch Renway into a new era. Maybe even save stupid Arthur's career at the same time. Leigh replied carefully. "We tested all the requirements provided to us with appropriate test coverage. It worked as per the requirements, and we can prove it—"

Arthur interjected. "That's not even—"

No! thought Leigh to herself. Not now! Shut up, Arthur!

And she actually said it. "Shut up, Arthur!" But she heard a strange echo to her words. Alicia had said the same words at exactly the same time. Her eyes were wide and aimed directly at Leigh, encouraging her to keep going.

So she did. "That was what our team understood the task to be, but that's not what we are building around here, Helen." My god! She thought. I just called her *Helen*. "We're not here to check boxes on a checklist. We are actually in the midst of building a completely new IT organization to not only prevent these fiascos from happening, but also so IT will become an integral part of the entire Renway business.

"For a start, we are making sure quality will be measured differently. Not by just passing tests and checking items off a spreadsheet. We will have to ensure that what is being delivered to our customers is achieving the intended business value, and we will do that by introducing continuous testing all along the development lifecycle. That's what we've been working on since Sunday. It's big and it's new. It will take us a while to get fully rolling, but that's how we see our future."

"Except for the fact we can't do that," Arthur interrupted.

My god, thought Leigh, this guy is a stubborn mule!

"We can't do that, not for years!" he continued. "We're not set up for that. We run monolithic legacy systems. Our staff's average tenure is fifteen years, and no one really understands the new technology stack required to achieve what you just said."

Alicia chimed in. "That is why our new IT organization is one we will have to build together. And it will have the customer at the center of everything we do. We must have a way to measure quality appropriately. And that also requires a new way of thinking. It's beyond the 'does it work' kind of thinking; it's about 'is the user perceiving the value we originally intended?' If it works as we wanted, but the user does not perceive any value in it, then it's a failure. We need to quickly go back at it with the lessons we're learning right now. From my perspective, that's how dotPlane was successful. That's how Uber and Google and Netflix do it. They listen to their customers and adapt quickly to their needs."

Helen Humphries stood up, indicating the meeting was over. She looked directly at Arthur for a few seconds and then looked pointedly at Alicia, as if sending her a silent edict. "I have a board meeting in one week. Please make sure I have something new to tell them."

She left the room, followed closely by her consultants. Everyone was still in shocked silence. Arthur left next, and as he did so, Giles sprang up and followed him quickly.

Leigh leaned over to Jessica. "Can you ping Dinesh? I think we should go grab a coffee up the street."

* * *

Leigh and Jessica emerged into the bright fall sunshine. The front doors of the Century Building opened to a forecourt of sorts, ringed by a low concrete wall studded every twelve feet with decorative knobs. The wall offered a place for its workers to sit, talk, and eat their lunches, while preventing skateboarders from terrorizing them. They saw Dinesh waiting for them. He was sitting next to someone they couldn't immediately make out, although the posture seemed awfully familiar. It was Giles.

"Giles!" Leigh exclaimed. "You haven't gone and beaten Arthur to a pulp, have you?"

"Not yet," Giles replied. "No need, actually. He's sinking his own ship without my help."

"He was a little harsh back there," Jessica added.

"He's scared. He's old school," Giles replied. "It's difficult for anyone to see their way of life become meaningless."

"Do you think he can be saved?" asked Dinesh.

"I really don't care," replied Giles.

Leigh, Jessica, and Dinesh walked on, enjoying the feeling of being outside, especially after the oppressive feel of the basement meeting room. As they walked the two blocks to the Perlman Building's coffee shop, Leigh told them about Steve Fibonacci and their history, right up to Monday's meeting and the caveman cartoon.

The coffee shop was on the corner of the building and had a small patio with tables. The autumn air was not too much of a threat, and most of the tables were occupied. Leigh recognized Steve right away. He had claimed a table in the corner and arranged chairs for each of them. He waved. Dinesh tapped Leigh on the arm and pointed to the cars parked along the street. Directly adjacent to the coffee shop was a gleaming Tesla S, dark red. Dinesh pointed at the vanity license plate: "Burrito."

Steve got up to greet Leigh with a respectful hug, then shook hands with Jessica and Dinesh. "Go grab your coffees," he said, "and we'll get started. They've already been paid for, just grab what you want."

* * *

Steve leaned his elbows on the table and steepled his fingers. "First of all, Leigh, thank you for inspiring me to put vegan dishes on the menu. We're getting great feedback already. I remember the day I first worked with vegetarian meals at FoodFlight. I asked one of our suppliers' sales reps, 'If vegetarians eat vegetables, what do humanitarians eat?' He didn't get it. Or at least, he didn't laugh. I guess he just wanted to make the sale."

"And did he?" asked Dinesh.

"No," said Steve, "but he might have if only he'd laughed. You always gotta have some fun, you know?"

He sipped his coffee. They all noticed how he focused on the taste and the aroma of the coffee. He was obviously passionate about his work. A foodie.

"To business," he said. "Leigh has told me that you are trying to introduce a culture of continuous testing into your software community, and you're at the point of wondering where you are in this journey or if you have even really started yet. After all, you're not even two weeks past your little reservations website scare. These are early days yet. So, I want to share with you what we did when we designed our IT organization structure about three years ago. I'm going to give you six action items to ponder.

"First, improve the relationship between each tester and each developer. There shouldn't just be teams on either side of the wall anymore. You should have tester/developer couples—like a marriage—who can communicate and give each other feedback on an ongoing basis. A loop. This gives both of them more timely knowledge of the source code being tested and its relationship

to requirements or the user story, and it allows the tester to give feedback to the developer on the type of testing that needs to be automated and which tests are affecting that source code for that feature that's being released or being merged into a master branch for validation.

"The trick is to keep the teams small because the more people who get involved, the more fuzzy and inconsistent the process becomes. But at the same time, each of these teams must remember that they form part of a larger community, so there must still be inter-team collaboration. So, make reports easy to access and share them online. Collaborative chat platforms are way better than email for this."

Dinesh smiled. "Yes, we're already trying to move away from email."

"Good," said Steve, "the easier it is for real information to be shared in a smooth flow, the better the teams will work. And there's a much greater sense of collective enthusiasm that way.

"Second, after relationships comes automation. Put automation as a priority. What does that mean? It does not mean removing all manual testing tasks. That would not be a good idea. Some things just can't be automated anyway—things like usability testing. But it helps to adopt an 'automated first' mindset and focus on areas that will be run repeatedly. I think of it like setting up the plumbing first so that you can let the water flow later without any worry or concerns about leaks.

"Think about your entire SDLC and how you would like it to look. From when you define a requirement to starting to code, check code in, build, deploy, test, release, feedback, and so on. As you map that out, envision how you can automate all those activities. Make sure you automate quality gates between each automated handoff. For example, when code is built, how do you automatically check if the build was successful? And who should be notified in case it wasn't so they can fix it immediately? Not tomorrow, but now? See? It's not just about application testing. It's about embedding quality through the entire application pipeline so that problems, including application defects, don't move to the next stage in the lifecycle. This is also a time saver and basically makes things more interesting because the dull, repetitive work gets taken care of through automation.

"Third, get small. This is a big jump away from the waterfall approach. Back then, you had blocks of work being designed, coded, tested, and deployed that were very large. What you must do now is break these down into smaller increments that are easier to test after coding and design. This makes it easier to automate the tests that you need to run, and it also makes things much more easily deployable.

"Fourth, keep track of everything. Use metrics. It's very easy to lose track of things when you're automating. Like setting up triggers for pull requests.

Or alerting the developer of the code that's been affected by a manual test or an automated test. And make sure you can easily set pass–fail criteria. Continuous testing is all about immediately identifying if things are working or not, so make sure you can set that up easily."

He picked up a fork from the table and looked at both sides of it. Then, he pulled out a plastic "spork" from the pocket of his shirt. It was a spoon, with three foreshortened tines built into it.

"Fifth, you gotta have the right continuous-testing tools. This is crucial. Good tools help get stuff done and get it done right. The wrong tools not only gum up the works, but they also drive people away from focusing on quality or even wanting to do the work at all. So, take some time to find the tools that will help you to develop, test, and analyze continuously. Here's what we did."

He reached for a sugar packet and laid it on the table.

"One. We tried out a bunch of different tools before buying. You know, it's like sampling food from a food supplier. Try them out on a localized test or project. See what people think. Find out what they can and cannot do well. And always consider open source. It's always evolving, and it has a powerful community behind it."

He placed a second packet next to the first.

"Then, we chose the tools that integrated best with the ones we were currently using."

"So, you're not using a single tool that incorporates all aspects of continuous testing?" asked Jessica.

"Nope," Steve replied, "that's the old way of thinking about tools. Now it's about picking the right tools for the job at hand. And we always tried to pick the best-of-breed tools that work together in a way that incorporates easily into the working environment. I think it's essential to choose tools that have community-based support, where everyone shares their challenges, solutions, and interesting use cases.

Leigh had been hand-writing notes on a series of paper napkins to look less like a court stenographer and to try and stay part of the conversation. She paused to rub her wrist. Writers' cramp was setting in. Dinesh noticed and grabbed a handful of napkins to take over.

"It's nice to see at least some things being done old school, despite our conversation here," Steve said as he observed the handover. "Never underestimate the productive power of writing by hand," he added. "There are many studies that show it can actually be more conducive to creative thought than typing or tapping. It's to do with multisensory input. I'll send you some stuff if you like. It's really useful when you're brainstorming in the war room. Rule number one: write things out.

"To that point," he added, "develop a system to display your results." He grabbed the napkin to fill out point number six himself. "Item number six. You have to be able to do deep dives into those results, because that's how you will know if your code is working and where the gaps are."

He rearranged the two sugar packets again, simply to draw attention to them.

"Result priority number one," he said, "is to define your KPIs and acceptance criteria and make them quantifiable. There is no point in setting a KPI that can't be tracked. It's like adding a license metric to an application that can't be tracked; it just makes no sense. These indicators should form an accurate reflection of the product's purpose and Renway's business goals. They could include test coverage, number of pass–fail builds, average response time, that sort of thing."

He pointed at the other sugar packet.

"Then, create dashboards to track the KPIs, including a baseline and subsequent changes. This will help your people assess things at a glance and see more quickly where the problems are. You might want to make these as public as is appropriate so that more engineers can see what's going on and can help fix things more quickly."

He picked up a straw.

"This also becomes part of your pipeline orchestration. You're bringing together developers, testers, and ops people inside continuous integration. To start seeing the value of continuous testing, people need to have something that will give them the ability to amplify the work they are doing. This is where you will see a real distinction between plain 'test automation' and continuous testing. You can create all the test automation you want, but when you have a CI pipeline built, you can quickly have all the automated test scripts in the CI pipeline and see the advantages right away. And remember, if you're still manually writing code to create your test-automation scripts, you're not going to achieve real continuous testing, since you'll have to stop and write that code to either create new scripts or fix existing ones. That's not really continuous; it's just plain good ol' test automation."

"Thank you," said Leigh, a little formally. "I feel like we just sat in on a university class without paying for it."

"There's nothing wrong with that," said Steve. "I used to do that all the time. The next time you look at all those fancy fonts available in Microsoft Word, thank Steve Jobs. He dropped in on a calligraphy course and became fascinated by it. And he put it into the Mac. And later Windows did the same."

"Yes," said Dinesh, "but that would have happened anyway, eventually."

"Most things will happen anyway, sometime," replied Steve, "but not right away. The IBM–Big Blue world was happy enough with dot-matrix printers, thank

you very much. You had to have someone with the vision and the guts to change things up, to ask why, and to build in a new approach. That's why Jobs is still quoted endlessly in business. He looked at things differently and made computing a different thing. And that's what you are doing, right now. That's what we are here talking about. Continuous testing is a new thing. You are introducing the flexibility of fonts into a world that has only known Courier."

"So, that's why you bought a Tesla?" asked Dinesh.

"Yes," said Steve. "Go and read on its history, especially how its software was coded. You'll see the same type of approach we've been talking about."

He looked at the three of them. "Not everything is going to work right the first time. That's why my robot got lost in your basement office the other day. Couldn't get a signal."

Jessica and Dinesh looked at each other, eyebrows raised. They hadn't heard that story yet.

"Get started with one small project, get that successful, and build other small projects," Steve added. "Go for small wins and build upon them. When you fail, you fail small. And you learn from it. You have to get used to the fact that, moving forward, failing will be part of your day-to-day. And that's a good thing. That's how you learn and do it better in the next go around. As Yoda says, 'The greatest teacher, failure is.'" He delivered Yoda's voice with stunning accuracy.

He checked his watch, the universal signal that time was up. "Time to go have lunch at the competition," he said. "The Foodery. I can't tell you how much I despise restaurants that brand themselves with an '-ery' suffix. Eatery. Grindery. Creamery. It puts me into insulin shock. But, hey, that's just me. Not everyone loves Italo-Mexican as a word yet either. But they soon will," he added with a wink. "Keep in touch, guys. I'd love to hear how things turn out for you. Not just the airline, but for you three in particular. You're bright sparks. That's important."

* * *

The three of them walked in silence. Steve had given them a great deal to process.

Jessica spoke up first. "I think he gave us a good set of steps there," she said. "We need to put them somewhere. Like a manifesto, just like the original Agile Manifesto."

"Like a wiki?" Leigh asked. "I've already started one. I should have shared it with you earlier, but I didn't think it was ready for you to see yet. Isn't that just so sad and ironic? Here we are talking about continuous testing, and I couldn't even put the wiki online for you to edit along with me."

"Old habits. Hard to break," answered Dinesh. "There's a good lesson there."

"OK," said Jessica, "I feel like we need to get some more answers. Find out the things Steve didn't say. How are we going to do that? Everyone's already meetinged to death, and I think we need to go beyond the company. Get some more impartial feedback from people on the outside. Anonymously, so it doesn't tie back to Renway. Quora maybe? Or Reddit? Maybe some vendors?"

"Yes," added Dinesh, "let's just fire out some questions. Set a timeline of 5:00 today. Just sample people who are online right now and see what we get."

"OK," said Leigh. "So, before we go back inside, let's take just ten minutes more out here and figure out the questions to ask."

"Yeah," said Jessica, "and let's fire them out there right now. Before anything else."

They had arrived back at the Capital Building and sat down on the very same wall where they had met just over an hour ago. Giles, of course, was no longer there.

* * *

Seventeen floors above, Alicia watched the three of them walk up to the building's apron and sit down on the wall. Her office allowed magnificent views of Collingwood and of Boston in the distance, and she often used her field glasses to look at interesting things going on, as well as aircraft approaching or departing Logan. This was especially enjoyable when she found herself stuck on the endless conference calls required by her position. Through the glasses, she watched the three of them huddle together, talking with obvious energy, gesturing with their hands, and typing into their phones.

She looked briefly back at her monitor. Open on the screen was the email from Lester Greene from Legal sent last Tuesday. "Eyes only. Green light." She sighed deeply and looked for a few minutes at Leigh, Jessica, and Dinesh as they continued their meeting.

Twenty miles away, well beyond the power of Alicia's field glasses, a private Learjet owned by MidWest Airlines landed nimbly on runway XYZ and taxied to the executive side of Logan Airport—the side with the private arrivals and departures lounges. Two limousines waited close by to whisk its occupants to Collingwood.

* * *

"Come on over, Bru," the text read. It was her dad. "Casey and the dog are already here. I'm going to cook on the fire pit instead of the BBQ. Old school!"

Leigh winced. She'd completely forgotten about this.

"Will be a little late," she texted back. "Have a meeting with Jessica and Dinesh at 5."

She had told Warren a lot about both Jessica and Dinesh a few weeks back, about their young energy and their expansive vision. He had laughed at that. "Listen to you, talking about the young folk and their new ways. You're aging yourself, Bru," he had said.

"So, do it here. Bring them by. There's plenty to eat," he replied.

"Perfect," thought Leigh, and immediately texted both of them to see if it was possible.

Half an hour later, Casey's SUV was joined in the driveway by Leigh's Porsche, Jessica's Jeep, and Dinesh's BMW motorcycle.

They all exchanged greetings, and Copernicus made sure everyone knew they were loved. Leigh, Jessica, and Dinesh talked to Casey, and Warren took care of the drinks.

"OK," Warren said, "you guys get to work so you can be finished in time for the food. Casey is in charge of kitchen prep, I'm going to get the fire underway, and Copernicus will be on squirrel watch."

"Sounds like a plan," said Casey.

Leigh, Jessica, and Dinesh retired to the dining room, devices at the ready. Leigh opened up the wiki, which she had now shared with the others, and prepared to add to it.

"So, we posted questions to Quora, Reddit, and LinkedIn. The first question was 'What barriers to adoption of continuous testing have you experienced?'"

Leigh typed: "Barriers to adoption of continuous testing."

Jessica and Dinesh scrolled through the responses.

Jessica read out, "Open source might not be allowed in your company. You might need a strategy around getting buy-in from leadership to change that."

Dinesh added, "Culture is the biggest hurdle. Culture. Not just people, exactly, but the culture of the place. It's the number one item everyone talks about. Your people. Do you have the resources? Will you have to swap people? Train them? Bring in coaches?"

Jessica continued, "The next barrier is time. There is no space for this to happen. Also, leadership. You need top-down leadership support. That means business and IT leadership support, together. The impact to the company culture can't be understated, but the benefits are worth it. Without the right types of leadership it will take years to make a difference."

"OK," said Leigh, and typed as she spoke: "Culture, time, leadership, and—oh yes—don't forget legacy applications."

"Next question," Dinesh said as he read from his screen, "Should the traditional Center of Excellence feel threatened? What is their role in continuous testing?"

Jessica read out from her screen: "In DevOps, people get fearful that their work will become redundant. They will question their roles. You have to talk through their strategy and role. Do some pilots. Whet peoples' appetites to hone their development skills. Let them know they can learn as they go along. Some people cannot deal with uncertainty and will leave, so explain things and give them assistance early and often."

Dinesh was next. "The biggest barrier is a person's mindset. Think about the Fisher curve. There's a range of emotions from complacency all the way through to seeing your way through the jungle. Humans are creatures of habit, and they need to see things to believe them."

He laughed briefly. "Ha! Who does this sound like? 'People go into denial. I've been doing this for 15 years! Why change? But once they see others do it successfully, that's when the change starts. Cultural barriers start at the individual level.'"

"So, a lot of focus on people and change management. Anything else?"

"I have a LinkedIn quote," said Jessica. "'Senior leadership driving accountability to the organization will make a substantial difference. An individual evangelist can only connect to 20 to 30 percent of the population. For the rest of the organization, it's up to leadership to set expectations and hold the course.'"

"Our next question was . . ." Leigh paused to read her notes, "What does a continuous testing maturity model look like? Does it include open source?"

"I have a reply here," said Dinesh. "First, get the right mindset, then build up test automation at the unit and API levels for functional, performance, and security testing. Keep test automation at the UI level to a minimum, as scripts are very brittle as UI changes. Then, add a feedback loop that includes performance- and user-monitoring tools in pre-production environments as part of your pipeline, just like you would do in production. That gives you full visibility into not only what is failing, but also why and where to fix it. Full telemetry. This accelerates defect resolution. Get to a point where all these areas are firing on all cylinders. All these capabilities must be fully integrated, collaborative, and not siloed in any way."

They looked at Leigh busily typing away.

"Wouldn't you like to just copy and paste this stuff?" Jessica asked.

"No," said Leigh, "thanks, but I want to capture the essence of this for the wiki and keep it as brief as possible.

"OK," said Jessica, "so here's some more on open source. This person writes, 'Definitely OS. It's key. Companies need to be able to embrace and accept open source, because that's where developers are going. Companies must be able to accept and understand this movement and shift from a *no OS* mindset to one that says *OS first*. With open source we can experiment and fail faster without having to justify expenses.'"

"One more," said Dinesh. "For a continuous testing maturity model, think of the destination state like an automated pipeline. All the tests are running into that pipeline, all automated. The low-hanging fruits are the smoke tests. Run those all the time while you remove dependencies with capabilities like service virtualization that will enable you to run the bigger integration tests, fully automated. Next, use a CI tool to run all automated tests, ideally at every code build. As you get more mature, you incorporate automated performance and security tests into the pipeline, along with the automated functional tests. Try to avoid running integrated functional tests as the main tests, as they have too many dependencies you probably don't control."

Leigh looked outside. Food was being cooked, and it was obvious that they were spending too long on work on a Friday night. Leigh agreed with the sentiment, but as she had explained patiently many times to many people, hers was not a job where you could just slide down the tail of your brontosaurus and leave the quarry on the dot of 5:00 like Fred Flintstone. Especially where quality was involved. She gestured at them, "Five minutes more."

"So, what's that segment on test data management?" she said, pointing at Jessica's screen. "Let's do that."

Jessica read out, "When you are trying to test a piece of functionality, you need some sort of data to make sure it's right. Customer info, accounts info, purchase records, just to test that the functionality is right. Many times, that data is scarce. People may want to pull data from the production environment, but they may have to mask it to make sure the API data is visible for developers and testers, who may not be entitled to see them, or they may not be able to generate that data. You need to get that sufficient amount of data to visualize all possibilities, to cover end points to do both the positive tests—that your functionality works—and the negative tests—certain data should not be supported by the business logic. To get all that right is very challenging. It's one of the things people don't think about. They accelerate the development and then they hit a red stop light because they lack data. Currently, they solve that problem by doing it manually on spreadsheets like Excel, but there's never enough data. Or they can pull from production."

"So, we'll include test data management as part of the maturity model." Dinesh closed his laptop triumphantly, hoping this would signal the end of the meeting. He was hungry and needed another drink. Jessica followed suit. Leigh started to close her apps but paused to check her email one last time.

"Guys," she said, "wait. I got a reply from Sylvia Finch. She's Managing Director of Quality Engineering at Aurora. I sent her an email today with the same questions. I never thought she would get back to me." She decided not to elaborate with the details of their meeting yesterday. Not yet, anyway. She read through the email. "Oh, this is good stuff. This I will copy and paste. I'll join you in just a sec."

Jessica and Dinesh stayed anyway and read over Leigh's shoulder.

"Leigh," it began, "glad to offer my two cents' worth."

Thank goodness Sylvia was not one for small talk, Leigh thought to herself. Nothing about "Nice to see you yesterday." No tracks to cover. She had simply inserted her replies to each question.

"For organizations that want to get started in continuous testing, what do you think the first step is to adoption?" the first question read.

"No one wants to get started in continuous testing if they see it adds four to six weeks to the timeline. They just want to do better. Continuous testing is a critical element to enable that. People will not convert if they do not recognize how it will help them with their mission. So, introduce continuous testing by saying, 'I want to help you successfully deliver all your demands, without delaying your schedules.'"

"What barriers to adoption of Continuous Testing have you experienced?"

"Regarding barriers, I have suggestions. Everyone across IT and the business has their performance measurements. Everyone is focused on meeting their own performance objectives. An app development organization is about delivery. It must build credibility and respect with business units that depend on technology. The business just wants it all to work. App development teams are under stress to deliver. Understanding the business needs and requirements as to how an app is to behave is essential to those app development teams. We are getting better with the concept of agile and integrated pods. The business is part of the pod. Provide continuous feedback. Show them we are moving in that direction."

"It's all about 'what's in it for me?' WIIFM. QA must be extremely humble. Our number one objective is to make our app team successful. We are the enablers. We are the supporting cast. It's a symbiotic relationship, and there's great honor in that."

"What does a continuous testing maturity model look like?"

"It looks different for everyone. Continuous testing should not be in the spotlight, but it's a critical enabler to velocity, quality, and reliability. The maturity model must ensure code coverage. At Aurora we had lots of issues with stability. Reservations is a highly specialized function. We spent a lot of time and energy testing the booking functions. More than once, we released a new booking feature, and the release resulted in an outage because we broke

something, and the website wasn't completing any booking anymore—that's when we realized that testing continuously was critical."

"OK," said Leigh, "save and close."

They walked out to the back yard to finally join the party and the food prepared so expertly by Warren and Casey. Copernicus put into action his own plan of seeking to convince each human in turn that he had not been fed for weeks.

CHAPTER

8

The Value of Metrics

It was a sunny Monday morning. The weekend had been uneventful. No crises, no calls to any airport hangars—just beautiful and all too short. Leigh parked her Porsche in her space and walked across the lot to the Century Building's rear doors, which provided access to both the above-ground parking lot and the smokers' picnic tables.

She was surprised to see Arthur sitting alone at one of the tables.

"I didn't know you were a smoker," she said to him.

"I'm not," he replied. "I am just savoring my last day here."

"What makes you think this will be your last day?" asked Leigh, not entirely surprised.

"I received an email request from Alicia. A meeting at 9:30. Twenty-five minutes from now. She sent it to me at 4:00 p.m. yesterday afternoon. Sunday."

"That could mean anything," said Leigh. She hoped she sounded supportive.

"Come on," replied Arthur, "the writing's on the wall for me. Especially after that run-in with Helen Humphries last week. These people have it in for me. You included. You all seem to think that taking on some new flavor-of-the-month design process is going to solve Renway's problems. But it goes a lot deeper than that. I have spent years building a system that gives people the authority and independence they need to do their jobs correctly. And now

J. Scheaffer et al., *The Kitty Hawk Venture*, https://doi.org/10.1007/978-1-4842-3661-1_8

all of a sudden that's not good enough. You want everyone to be engineers. Testing everywhere and anywhere. You want to take on these new upstart airlines that have been in business all of two weeks. Why? Because they have a cool website and a nifty app? That's not what puts planes in the sky, Leigh. It's craftsmanship. Pride in a product and a respect for tradition."

He looked up into the clear blue autumn sky. A westbound aircraft was laying contrails at 30,000 feet or so. Coming in from Europe somewhere, maybe heading to Houston or even LA.

"Frankly, I don't think Renway will survive much longer anyway, at least not by itself," he added. "Don't you feel that, Leigh?"

"I think we should always be on the lookout for that next opportunity," said Leigh.

"You mean for the airline or for yourself personally?" Arthur asked.

Ouch! Leigh thought to herself. *That was uncalled for.* She shook it off as the amplified bitterness of a cranky old man.

"Sehr gut," he said. He got up from the picnic table and walked inside.

Leigh shook her head to dislodge the clumps of shock and confusion that had wedged themselves in her brain. *Why is he speaking German to me? Fergus isn't a German name, is it? Maybe his wife is German. Maybe he's learning German. That could be it.*

But she left feeling a little out of sorts, as if that *wasn't* it.

* * *

Walking to her office, she stopped at the basement kitchenette to get a coffee from the coffee pod machine. Derek Zajdner, her boss, was there, washing out his coffee mug in the sink.

"Good morning!" she said, rather shocked. "It's good to see you back! But we weren't expecting you until next week!" In hushed tones, she added, "That's when the party you're not supposed to know about is!"

"I couldn't wait," Derek replied. "Four months in a hospital room can make you stir crazy. And fearful. It's amazing what your mind comes up with when it has nothing else to do. It's nice to be back in the dungeon."

A hiking accident that resulted in torn quadriceps in both legs had kept him out of circulation all summer, exchanging his screens and keyboards for hundreds of hours of physical rehabilitation exercises. For a while, he did not even know if he would ever walk again.

"Let me carry that for you," Leigh said, pointing at his full coffee mug and looking directly at his new walking cane.

"No, that's OK," Derek replied. "Thank you, though. I have to learn to do these things, as clumsy as it makes me feel."

They started to walk slowly down the hall toward his office. He leaned heavily on his cane and kept his attention on not spilling his coffee.

"I have been in touch with Alicia quite frequently over the summer, as you know," he said. "I appreciate the extra work you and your team have been doing in my absence, and it looks like the continuous testing initiative is already starting to make waves. Would you agree? Is it catching on, in your opinion?"

"I'm not sure 'catching on' is the phrase I would use to describe it," Leigh replied. "It's still quite a tough sell. People just can't let go of what they're comfortable with. But the fact of the matter is, this must be done. It's what the business demands. It's what our customers are demanding, even if they don't use those terms. There's a lot of greener grass appearing on the other side of the fence."

"Yes, an apt analogy," Derek replied. "So, OK, let's look at where you—we—are right now. You have introduced the concept of continuous testing. So, what now? How will we fully define what it should look like for us? How will we track its progress and the successes? Are we going to run a pilot? A test phase? It's going to be important that we establish metrics to illustrate the progress we're making. We have three tough audiences to please: the brass, the customers, and our own team."

"Don't forget the media," added Leigh.

"Yeah, OK. Four tough audiences."

They arrived at Derek's office. The conversation paused for a while as he settled himself carefully in his chair.

"I'll tell ya," he said, finally, "just like the song says, 'you don't know what you've got till it's gone.' It's gonna be a while before standing up and sitting down become less of a project for me."

He checked his watch. "There's a summit happening in Chicago right now. It's a full-week conference that started this morning. It's focusing on a bunch of forward-looking issues in corporate tech. Mostly blockchain, AI, and DevOps. But there's a session happening this evening that I think you should go check out. It's an early evening seminar on continuous testing metrics, and it ends with a bacon crawl."

"A what?" asked Leigh.

"A bacon crawl. It's a pub crawl, but they're trying to take the focus away from beer and shots by introducing something healthier, hence, bacon. I think you should go, in person. I know it's short notice, but it's timely."

"I'd have to go home first and pick up some overnight stuff," said Leigh.

"Well, only if you need to stay overnight," said Derek. "We could get you on our 2:30 to O'Hare, and you could come back on the 11:25."

"OK, I can do that," said Leigh. She instinctively looked at her watch.

"Great," said Derek. "I'm sure we'll have lots to talk about tomorrow.

* * *

Five hours later, Renway Air Flight 742 to Chicago O'Hare pushed back from its gate at Logan International Airport. The flight attendants started in on their departure routines, closing overhead bins, inspecting seatbelts, pointing out exits, and delivering the safety lecture. The ground crew guided the plane toward the taxiway. In seat 17-B, Eric Edelman, a sales representative for a farm equipment manufacturer based in Moline, Illinois, quietly unbuckled his seat belt and slid across to the unoccupied seat 17-A, the treasured window seat. His seat neighbor on the aisle, in 17-C, nodded in appreciation. They would both have a little extra space for the flight. The intended occupant of 17-A, one Leigh Freemark, had checked in but never boarded.

From a food court inside the Logan terminal, Leigh watched the Renway plane as it started its slow trek toward the taxiway to get in line for takeoff. She looked down at the dotPlane app on her phone. The little chat bubble said, "How are you enjoying your complimentary lunch, Leigh? Boarding commences in 15 minutes. You have the window seat!"

She selected a happy face emoji in response.

* * *

Chicago is such a beautiful city, Leigh thought as her airport limo dropped her at the massive front entrance of McCormick Place. She knew she could spend days just walking amid the stunning facades, gazing at the architecture that gave the entire town a sense of grace, strength, and classic geometry. Her dotPlane flight had gotten her in literally fifteen minutes after the Renway flight. Her limo driver was still there, waiting patiently inside the arrivals area, as limo drivers do, holding up a card with her surname on it.

She checked in at the convention's welcome area and found her way to Conference Hall A, where a marquee announced the event inside: ***Understanding the Metrics of Continuous Testing***. It was 5:15, and she saw that the panel was already talking. She had missed the introductions and housekeeping, but she could see she was not the only late arrival. She took her customary seat in the back row, at the end. In place of the regular PowerPoint screen, a simple line of text, like surtitles at an opera performance, summarized the session's objective:

"Q1. What are the Key Measurements for Success in Continuous Testing?"

Below the surtitle sat three people on stylish barstool-type seats.

Leigh looked around. She noticed that on the back of the seat in front of her was a card with a QR code, the conference logo, and the words "Play-by-Play." Every seat had one. Intrigued, she pointed her phone at the code. An augmented reality app started to download itself onto her phone, pausing only to ask permission to continue. When it was done, the text on her phone screen read, "Point your phone camera toward the stage." She did so and saw on her screen an enhanced view of the stage in front of her. As each expert spoke, a bubble popped up above their heads, providing the expert's name, company, and other details. Hashtagged comments appeared even higher above their heads, and even the backdrop changed color, subtly and very pleasantly, incorporating visuals and images that the real-world screen did not offer. Leigh could only wonder what types of data such an app could pull in, if it was being used all over the convention center for polls, food orders, networking, and subscription requests. What a goldmine of real-time information.

According to the AR app, the expert sitting on the left side of the stage was Andrew Waterbridge, whom Leigh knew of through Twitter and several well-written blogs.

"The whole point of metrics in general is to identify business value," he said. "Regardless of the walks and talks you do, if you can track the business value, and you can show you are meeting the targets—like customer adoption, engagement, revenue, whatever the business value—those are your overarching benchmarks.

"But down to basics of continuous testing; you're looking to track how many problems you have. How many defects in pre-production and how many in production? You need both. But leakage is more important. So, too, is the speed at which you can deploy. These, when combined, show if you are releasing higher-quality apps faster. You will expect to see the number of defects drop, obviously, as people write better and better code.

"Another method is, you take your code coverage and your required coverage, what we call functionality coverage, and you combine them to track the overall functionality coverage.

"Secondly, there is the sentiment analysis of your end users. You have to understand the sentiment of your end users and add them to a continuous feedback loop that goes back to developers and the business. So that, plus business value plus overall functional coverage—if you can track these top three—you're well on your way to validating that your continuous testing is right. In fact, continuous testing should probably really be called continuous validation, because you are constantly validating that your app or service will provide or is providing the business value expected."

The female speaker sitting on the next stool spoke up. Leigh's app identified her as Neha Mehta, VP of IT Operations for a national retail payment clearance company.

"At the moment, few people do metrics very well," she said. "First, you have to understand what quality is for your organization. A baseline, if you will. If you don't know where you are starting from, you can't tell if you are improving. Quality in a DevOps environment is way different than it is for waterfall. Also, the way we log defects is different. So, what is a fair indication of quality? Ideally, that is viewed from a production perspective. You will need a KPI that establishes production quality. For example, 'Do I know the ratio of released to production defects?' This means the ratio of each, and the difference between them. Both need to be defined because it is not obvious. That's the foundation. That's the North Star that points to value.

"Second," she continued, "like Andrew just said, there's code coverage. Being able to know what kind of coverage there is across unit tests and functional tests, and performance and security. It should no longer be thought of as functional and non-functional testing."

She looked at the audience. "Who thinks security testing isn't as important as functional? Who thinks performance isn't a huge part of a customer's experience?" No hands went up. "So, let's stop devaluing performance and security testing by calling them non-functional.

"Then, there's overarching time to production. The point of continuous testing is to reduce timelines to release higher-quality releases that provide business value. More traditional groups will have pass–fail rates, but with DevOps and agile, some of those metrics are less important. We care about increased releases while decreasing the rate of production defects and condensing the time to market through automation."

The host, Erica Dell, spoke up, alternately looking between the audience, the guests, and her tablet.

"We have a question from the audience," she said. "Traditional teams focus on more traditional metrics, so what are the challenges in convincing people that pass–failure rate in QA is less important? How can you change people's minds to look beyond pass–fail?"

Andrew answered. "Don't forget performance and security testing. But you're right. It's tough. In waterfall, you can't get away from it. But as you start agile transformation and put people on teams, you must get an aligned definition of *done*. You should start tracking stats at a story level. There's no point in pass–fail. I care about my burn rate. Why do I need to log a defect when I can just have a conversation with a developer who is sitting near me?"

Erica responded. "You still have to log the defect. Defects have to be tracked, and automatically is preferred. That way, we have stats on it so we can use the analysis to help with improvements. To not log a bug and just have developers fix it might hide a bigger problem: you might not write a good test to validate the bug was fixed. So, I wouldn't advocate not logging bugs."

Neha chimed in. "There are other types of metrics, too. For enterprise organizations that have converted their Centers of Excellences into Centers of Enablement, you want to know how they're being effective in scaling the adoption of those key capabilities. For example, if you had a Performance Testing Center of Excellence, and now they've become responsible for enabling agile teams to achieve proficiency and run tests on their own, you will want to measure how many agile teams are actually achieving proficiency. And that could be measured by the number of agile teams that have started doing performance testing in their own sprints, for example. Or even how many performance-related issues have been found within the sprint before and after the team was enabled by the CoE. If that number is trending up, that's a good indication of adoption. Then, you start rolling those metrics up to the enterprise level to get visibility into adoption across all agile teams in the organization. It's a similar concept for the old Service Virtualization Centers of Excellence or Test Data Management."

The text graphic being projected onto the proscenium advanced to the next talking point: "How to measure continuous testing success."

Neha started the chat. "With success measurement there are overarching timelines being brought in. For example, a high-performing team delivers working code to production every fifteen to thirty days. As you start to build metrics around testing, you realize that testing in the lower test environments is done quickly because the tests are smaller. But when you do regression testing from an end-to-end perspective in the staging environment, it takes 25 to 30 percent of the overall time. If you can eliminate the tail of regression and performance in the pipeline you are able to reduce time to market by 50 percent.

"Today there is still a conversation around testing holding up production. Continuous testing can eliminate wait times with longer-running tests. The dialog changes. The bottom line is, testing can keep up with development, and you can ship code at any point in the sprint, not just at the end of it. That's the evolution into true continuous delivery, enabled by continuous testing. If it's ready, it means it's tested. For real. So, push it to the users!"

Leigh's phone vibrated. She looked at it. The app showed a simple phrase: "Key takeaway: Testing can keep up with development, and you can ship code at any point of the sprint, not just at the end. Do you agree? Yes/No/Not sure." She held the app up. Through the AR screen, responses to the poll were appearing in real time. Not as a standard Excel bar chart, but as three different-colored Roman columns, growing from the stage to the right of the guests.

Neha continued. "Currently, teams can't see the blockers because the testing bottleneck is getting in their way. I call this the Fog of Testing. Once the fog lifts, testing becomes part of the project. For example, what if we went to a microservices architecture? Are we doing branching? Can we use feature

toggles and blue-green deployments? You don't even get to that conversation because of the fog. This is hard for people to grasp until they hone in on that problem. Then, they can start to see that the behaviors from a DevOps perspective need to be changed as well."

Erica, the host, spoke up again. "Our next metrics question is one of quality versus quantity."

Andrew took the floor. "This is one of those issues that touches several points. Time to market, value-stream analysis, and the timelines and costs of manual test execution, for example. But here's one more illustration: let's look at regression testing. We had multiple teams doing regression testing in the UI once per month. And that took several days to be completed. As we started shifting left, we were able to automate our testing earlier and focused it on non-UI components. So, when we got to the UI, we didn't expect any functional or logic problems and were able to focus more on the user journey across the business processes enabled by the UI. That approach also enabled us to achieve continuous testing across the SDLC, starting from when the first line of code was written, until the story had reached its definition of *done*, which in our case included being released to the user.

"Plus, the average regression suite only handles 20 percent code coverage. So, let's automate, let's reduce the number of regression tests that need to run, let's ensure we are getting much closer to 100 percent coverage. Yes, people have been talking about automation for twenty-plus years, and regression is a place that many have focused on. But the reason this has failed is because everyone—well, almost everyone—has been focused on UI-based automation. And we all know that when the UI changes, the automation breaks, and it's not one test, it's twenty, then one hundred tests, and there is no time to go back and fix all those automated scripts.

"Our regression needs to get lower in the stack. We need API-based regression tests. We don't want to get rid of UI automated tests, but we can't rely solely on them."

Erica asked, "This obviously offers substantial efficiencies through the system, which the metrics can show, but what about the cultural acceptance of these types of changes throughout the organization? Should that be tracked and discussed too?"

"Certainly," said Neha. "There is a significant cultural aspect that many organizations do not do enough justice. We have to talk about how change occurs both from the top down and the bottom up. As you move into continuous testing, how does your company handle change? I know of one large financial services company where people are encouraged to embrace change. It's not just talked about. It's rewarded. But there are many others, the majority, actually, where this is still just lip service. Leadership must continue to recognize people and create mechanisms to socialize. Create space for

people to interact. You know we talk about continuous testing as being all about the numbers and the sprints, but it is equally important to establish a culture of change, where people who have been used to doing things a certain way and are now facing change and uncertainty—it's essential to put plans in place to handle this and to establish metrics to track it and manage it in real time. Socially, you're rewriting a cultural code using real human beings, and that needs its own version of continuous testing."

This time the surtitles above the speakers' heads changed to a graphic—very simple and very tasteful. It consisted of three circles in a row. The first contained a left-pointing arrow, under which was the phrase, "Shift testing left to developers." The middle one had an icon of a group of people. Its caption read, "Democratize testing tools." The third contained three arrows pointing at each other like a recycling symbol. Its caption read, "Testers evolve to engineers."

* * *

Back at the Renway offices, on the 17th floor, a different set of three circles was being projected on a different wall. The wall was in Alicia's office. The audience consisted of Alicia, Derek, and Arthur. These circles each contained a photograph, each taken at Logan Airport's departure areas. The first showed Leigh talking with the four representatives of AbLuft. Below the photo was the AbLuft logo. The second showed Leigh talking with Sylvia Finch. Below her circle was the Aurora Airways logo. The third circle showed Leigh in line at a boarding gate, taken just a few hours ago. Below this circle was the dotPlane logo.

Arthur's face appeared quietly impartial. Alicia and Derek sat with eyebrows raised and cheeks puffed out, silently asking each other, "What now?"

* * *

In Conference Hall A of McCormick Place, the surtitle read: "Most Valuable Metrics."

Neha was explaining that every leader could and should have their own view of what constitutes the most valuable continuous testing metrics. "What is your ultimate goal?" she asked. "For most companies, the reason for doing this is to increase speed to market while getting improved quality. So, that forces you to define what quality is.

"One of the credit and debit card groups we deal with, they had multiple releases per month, and one in ten would be an incident. As they implemented continuous testing, the number of releases increased, while the incident count stayed the same, which is in essence a decrease. With continuous delivery, they doubled the number of releases per month and started to record a true decrease in incidents.

"I have seen other companies that have no dashboard for product quality. Everything looks different, and their quality people have to go around and find out what's happening, basically policing and reporting the entire process. They don't yet truly understand the problem they're trying to solve. Such specifics must be consistent across the company in order to ask the question—is the work we are doing the right work?"

Erica asked, "So, if we were to draw up a preliminary list of the ideal metrics to track?"

As the two experts answered, the takeaways appeared on her conference app:

- Lead times
- Releases
- Production incidents
- Business value

As she was reading this, two large monitors descended from the ceiling. Simultaneously, a player appeared on the app. Erica addressed the group.

"As part of our special session on continuous testing metrics, we are proud to bring in the expertise of Dr. Joseph Perry, Senior Director of Mobile Applications for Taze. Taze is a green power supplier that focuses on solar-, tidal-, and wind-generation technologies. Their head office is in Hobart, Tasmania, and they serve markets in Australia, New Zealand, and Malaysia. They are seventeen hours ahead of us, timewise, which means Dr. Perry is enjoying his second coffee of the morning at 10:45 Tuesday. Good morning, Dr. Perry."

"Good morning," he said.

Leigh was relieved to see that Perry was speaking directly into the camera and not downward, as most people did when talking through Skype or Google Hangouts. They always looked at the screen rather than at the pinhole camera, which was inevitably at the top of the screen. Dr. Perry evidently had a camera crew.

"Dr. Perry," Erica continued, "can you share with us what you call the 'Efficiency Imperative' of continuous testing metrics?"

"Well," Perry began, "organizations have been trying to adopt agile for some years now, and they are getting to the point where the law of diminishing returns kicks in. Think about the transition that happens with agile. Whenever you're doing a release, there's a certain amount of risk, and a certain amount of cost associated with mitigating that risk. I call that 'paying the release tax.' As you move from waterfall to agile, agile adoption drives you from an eighteen-month cadence to three. This requires test efficiency, ideally gained from automation and shift left. You're testing faster, smarter, and cheaper.

This represents all the testing you have to do to make sure it's ready to go, and so you're paying that tax more often. The only way to keep the cost of this contained is to get more efficient about how you do the testing. If I get twice as efficient, I can afford to pay it twice as often.

"Well it's not just how you do testing. It's also the architecture of the application. So, some applications can only get so much faster. That's fine, but it can't achieve, say, faster than three-month releases. But it's still much better than two-year releases. So, it's not just testing, but also app design.

"However, it hits a wall. As you go toward shorter and shorter releases, in terms of weeks, from six months down to one week, you get a huge inflexion toward the marginal cost of getting shorter cadences. To get there, it can't just be about getting more automation, more shift left; it's about thinking very differently about how you test.

"If you put this into the perspective of continuous delivery, deployment is enabled by architecture, containers, blue-green deployments, feature toggles—a bunch of stuff that is not to do with testing but enables you to test in a very different way, with things like in-production testing as well as pre-release testing. We're dealing with very small payloads, where you can actually figure out the risk surface. What tests do I need to run, and what tests can I live without running? These are payloads small enough that you can actually figure out what the risk surface is. You are testing surgically rather than doing an entire regression. It's a totally different tax regime. That's what's different about CD. But how do you get there? How do you move toward testing differently? How do you get smart about thinking how to automate, figuring out which test you need to run, in which test configurations you need to do all that integration testing smartly, in the face of that massively exploding complexity? We need a whole new way of thinking.

"Think about the current development world. We've got QA, development, operations; we've got CI, CD; people are doing in-production testing; you've got a whole bunch of that stuff going on in the organization. You also have the testing that used to be happening in testing COEs or QA organizations. These are starting to shift left because of agile adoption, and you're getting this very distributed approach to the testing—in CI and in CD. Does it add up to something that's coherent as the right way to test that application for that release, given what's in it? How do you know that that's efficient and effective?

"So, what this calls for is a competency around continuous testing that isn't something you do between CI and CD but rather is something that is a competency in which CI and CD live, enabled by a set of tools that assist these different ways of testing and different ways of thinking about testing. This is where the future is going, and I think the idea of coherent test planning and coordination is particularly interesting in regard to that exploding complexity

problem. That's why you need tools that help with planning, test plans, and test automation, that then get implemented across the CI and CD implementation tool chains."

"What advice would you have for viewers considering continuous testing strategies?" asked Erica.

"Look at where your biggest problem is," replied Perry, "just like lean principles. Where are the constraints, where is the waste? None of that changes here."

Erica thanked Dr. Perry, and his face disappeared, replaced by the conference logos. She then thanked her guests and reminded the audience that the video of this meeting, including audience tweets and polls, would be available online within minutes.

Leigh carefully typed some entries into her wiki.

It was 6:00 p.m. Chicago time, 7:00 Boston time. She decided to pass on the bacon crawl and give herself thirty minutes of fun. She downloaded the app to a lakeside bike rental service and reserved one of the bikes in the stand next to the hotel. She texted her limo driver to meet her at Cloud Gate, the famous reflective "bean" sculpture in Millennium Park, in an hour.

CHAPTER

9

The Security Perspective

"Espionage is an extremely serious allegation, Arthur," said Helen Humphries, who had joined the meeting in Alicia's office. "Where is Ms. Freemark at the moment?"

"I sent her off on a bit of a wild goose chase to Chicago for the afternoon," said Derek. "I wanted her completely out of the loop for the moment."

Helen looked at the image still being projected on the wall: the three photos of Leigh. "And you have no idea what was said?"

"No," replied Derek, "I was running this from my hospital bed. I sent a guy out to follow her, but he couldn't get any audio."

"So, all you have," continued Helen, "is an admittedly free-spirited individual who appears to be cavorting with other airlines for reasons we cannot fully determine. Agreed, it does not look good at first glance. But we must be sure. How did these suspicions come about in the first place?"

Arthur spoke up. "Leigh has been making something of a nuisance out of herself recently," he said. "Ever since the cutover she has taken it upon herself to shake up the whole IT department and start switching jobs and roles around. She wants continuous testing to be implemented across the whole company, and frankly I don't think she knows what she's up against. I have been developing code since she was in diapers, and there is a rigor to both development and testing that I don't think she is familiar with. She never

J. Scheaffer et al., *The Kitty Hawk Venture*, https://doi.org/10.1007/978-1-4842-3661-1_9

seems to be in the office. She's always tooling around in her car, and I think she's sharing information with our competitors who want to spy on us."

"What information and what competitors?" asked Helen.

"And why?" added Alicia.

"We have a theory," said Derek. "We think she knows about Kitty Hawk," he started. "We think she's spooked about our acquisition of MidWest, and she wants to lower the value of Renway as an airline to get the shareholders to vote against the purchase. What better way than to share details of the deal with other airlines, or maybe reveal our current IT culture that brought about the cutover problem, so they can make a counter offer or devalue Renway or do some other sort of move that will cancel the deal?"

"And also," added Arthur, "by sowing chaos inside the IT department, getting people to work against each other instead of in the way we've always been doing it, the airline looks weak and disorganized. Maybe she's selling information, maybe it's sabotage. I don't know."

"What makes you think she's not just scouting around for another job?" asked Helen. "That's not against the law. Maybe Aurora or dotPlane are interested in her. People do move on, occasionally," she said.

"Then why did she just fly dotPlane to Chicago? I got her a seat on our own flight. If she wanted to check out the competition, why didn't she come to me, or to Alicia, to do it through regular channels?" asked Derek.

"I agree that's not within normal operational procedure. It might even be a violation. But we're not in competition with AbLuft," said Helen.

"Not directly," said Arthur, "but people forge new routes and alliances all the time. If she was doing this legitimately for Renway, then why the secrecy? We could have sent someone else to make the connection."

"Perhaps that's why she did it," said Alicia, dryly.

"What about Sylvia Finch at Aurora?" Derek interjected. "She's high level. Why would she want to talk to someone at Leigh's level unless there was dirt to be found?"

Helen Humphries stood up. "I will have somebody take a closer look at this. I will reach out to Sylvia myself. Arthur, you and I have sparred many times over the years, and frankly none of our interactions have given me great pleasure. However, if it is found that Ms. Freemark has indeed assisted in our competitors' espionage attempts or is attempting to sabotage the acquisition, you can be assured the appropriate recourses will be undertaken. Good day to you."

* * *

The interior of the departure lounge at O'Hare looked a lot like Logan, or any one of the thousands of other airports around the world: lined with seats filled with people wanting to be somewhere else. Garish lighting. A strong tinge of security-related tension in the air, giving a sensation like tinfoil on dental fillings. Departure lounges for the masses are never comfortable places.

Leigh slumped down on a chair, exhausted. It had been a long commute for a one-hour presentation that she had missed the first quarter of. She had liked the app, though. She pictured the room full of developer-testers that were behind it. No. Giles called them engineers. She pictured the engineers, all likely much younger than her, a forest of colorful clothes, multitasking mavens, with the occasional hipster beard and man-bun thrown in.

Whirrr.

Without moving her head, she lifted her vision from her phone screen to just above it, and then followed as a Veneto Burrito Foray robot scooted by, pulling its little trailer full of hot food. Her eyes tracked it from left to right and then returned to her phone screen. She pulled up the Veneto Burrito app and placed an order.

Minutes later, the robot pulled up in front of her. She flashed her app at its eye, and the trailer lid popped open. "Gotta hand it to them," she thought, "it's great food."

Ten minutes later, she was done and was feeling much better. A text message appeared on her screen. It was Steve Fibonacci.

"Skype?" it said.

"Sure," she replied.

Within seconds a Skype video call appeared on the screen. Leigh fiddled with her earbuds and answered.

"So," Steve began, without any formalities, "two cannibals are eating a clown. One turns to the other and says, 'Does this taste funny to you?'" He then collapsed in helpless laughter. "That has to be one of the funniest jokes I have ever heard. Right up there with the ship carrying a load of yo-yos that sank 150 times." He wiped tears from his eyes. "How are you, Leigh?"

"Great, thank you," said Leigh, suppressing a smile. "Still monitoring your clientele's orders, I see."

"Only the special ones get to me."

"And you waited until you knew I was finished. Nice touch."

"Nothing should interfere with the enjoyment of great food," he replied. "I had planned to get in touch with you anyway. You're heading back to Boston? Or are you moving further west?"

"Boston," she replied.

"I have someone I think you should meet," he said. "He's a security specialist who has some important things to say about continuous testing. He also has some aircraft industry in his background. And he likes ribs."

"He likes ribs?" Leigh echoed, unsure of the connection.

"Yeah. That would be his fee. Barbecued ribs. And a nice whiskey. Kentucky Bourbon or Islay. In exchange, he'll give you an earful on building security into continuous testing."

"And he's an aircraft enthusiast?"

"Yup."

Leigh smiled. She talked to Steve Fibonacci for a few minutes more until she heard her boarding call.

Just before joining the line for Renway Air flight 745 to Boston, she texted her dad.

"Ribs, beer, aircraft stories? I'll bring all three if you supply back yard and BBQ."

"Deal," came the reply.

* * *

Once again, Warren's driveway played host to Casey's SUV, Leigh's Porsche, Jessica's Jeep, and Dinesh's motorcycle. Additions this time included a Mini Cooper with Union Jack flag, belonging to Angela Kim from Renway InfoSec, and a sleek-looking 1973 Volkswagen Karmann-Ghia, orange, with the signature Veneto Burrito rocket-fork on the hood and the slogan "Point.Click.Burrito" laid out along the side panels in a darker orange. This two-seater brought the guest of the evening, Peter Cheswick, and his driver, Steve Fibonacci.

Once again, Casey took over kitchen duties. Warren oversaw drinks and barbecue lighting. Copernicus allowed every guest to inspect his new stuffed octopus toy, a gift from the CEO of Veneto Burrito.

"Here," Steve said to Casey, "from our kitchens. Perfection in Italo-Mexican cuisine and an ideal complement to barbecued ribs. Guacamole-infused cannoli. The recipe has been blessed by the Sisters of Perpetual Hope Convent and Orphanage in New Mexico. We call it the Holy-Moly-Noli."

"How uplifting," said Casey with a smile.

"OK," they heard Leigh call from the back yard, "perhaps business before pleasure? I would like to introduce you to Peter Cheswick. His title is Senior Software Engineer for Veneto Burrito, and his specialty is security. Just as a bit of background, for those who don't know, Steve Fibonacci, who is currently in the kitchen tossing salad, is the CEO of Veneto Burrito and my former

boss from a previous life. It was only when we had the meeting last Monday that we reconnected, and he's been quite a mentor to me in both lives. I have not met Peter before, but as we move toward creating a continuous testing culture at Renway, it's obviously imperative that we talk about security. So, welcome, Peter. Perhaps we can start by asking you for your own definition of continuous testing?"

"Thanks, everyone," said Peter. "I have to say, Warren's back yard is one of the nicest conference spaces I have ever spoken in. No PowerPoint needed! I think, to answer the question of a definition of continuous testing, you have to think about DevOps' having three components that lead one to the other. The first is identifying and confirming the value chain that you have as you go from inception to customer. You have to include testing in all aspects of that value chain. So, that's step one.

"Step two is to start putting feedback loops in there so you understand whether or not you're meeting that grade. It also enables you to fix issues as they arise and not wait for a user to complain. We need to make sure that when we set our acceptance criteria, it is more than just functional; it needs to have performance and security requirements as well. This will go a long way to helping developers and testers alike ensure high quality.

"Then, step three, you get information about where the team is struggling, and that allows you, from a continuous testing point of view, to talk with the developers about their coding. I'm going to move that into the kind of assurance space, the CI form of CI/CD, where I'm doing continuous integration and sending the tests that have been written by the team to do a basic sniff test, seeing whether the code sucks, and moving into the CD portion. Then, you take that CI portion and feed that back. You're always looking in that data for opportunities to get better. This is the part of DevOps that is all about continuous improvement. You take the data and ask 'Where am I weak? Where am I falling down? Where am I not meeting the grade?' This becomes an opportunity to bring training, education, more tools, those kinds of things into the process, such that in the next round you suck less and then in the next round you're getting better and better. That's continuous improvement.

"So, bottom line, continuous testing is that continuous improvement journey, such that you don't end up with a disaster in production. So, if something gets missed and it goes through the CD pipeline and goes into production and breaks in production, continuous testing says, 'I found something, I need to add a test farther down the stream.' The question becomes how far left can I go to find that? How far left can I put that test in there to prevent that problem moving forward? So, that's how I think about continuous testing."

Leigh spoke up. "OK, let's dig a little deeper and discuss the role of security within continuous testing."

"Well," Peter said, "do we have any developers here?"

Leigh and Angela said "No" in unison.

"Well," he continued, "when we talk about opportunities to train, remember, developers were never trained on how to make their software secure. It's not part of the college curriculum, and even if it is, it's not a requirement. From a security perspective, you have to assume you're working with an untrained workforce."

Leigh and Angela looked at each other and smiled, both picturing how Arthur would respond to being called part of an untrained workforce.

Peter continued, "This becomes a wasteful practice. Developers unknowingly create security vulnerabilities. But as you start to introduce continuous testing, you'll start to see the places where they introduce these vulnerabilities. My security team's job is to look at this data and find out what training they can bring to bear on making the developers better, such that they start to understand and become aware of the security of their code and not create those vulnerabilities in the first place.

"Developers need to be clear on what the expectations are for what we will release, what we won't release, and we need to give them the training and the tooling necessary to measure that, and then do overtime helping them get better by introducing more and better education that you can then measure downstream. So, as you go through that CI-CD pipeline and you're running the static analysis, dynamic analysis, software composition analysis, you start to see the mistakes that they made, and you use that opportunity to make them better. But the security testing needs to happen at every step. Developers need to run security scans so they find problems when they happen. Sound familiar? That's right! It's exactly the same thing for functional testing.

"So, bottom line here, that's security's job. To be that partner and the trusted advisor and expert to uptrain the development team so that eventually they no longer write those vulnerabilities in the first place, and so they're faster."

Angela spoke up. "So, what kind of person does the security specialist need to be in this instance? I've never heard the term 'trusted advisor' thrown our way before."

Peter grinned. "Sounds so warm and fuzzy, doesn't it? It's really part of this breaking down of walls thing, which so much of continuous testing is, both in terms of the software and also the people. I mean, in security we have the exact same challenges that you would have in QA: not enough authority to get everything done but all the responsibility to get it all done. You've heard of Centers of Enablement? That's a big change over Centers of Excellence. Instead of the security team doing all the testing, we are advising Dev and Test how to test and develop their code.

"I have to qualify my comments by saying at Veneto Burrito I have been able to build this culture from the ground up. It's been helpful having Steve at the

helm, because he fully understands the need for IT to be at the table for everything that involves the life of the company. But in my previous job, I can say that I experienced the typical role scenario. People just saw that it was my job to do the security, my job to do the quality. I wasn't getting anything from the other side. They would just ship it over to me, and then it became my problem. That's exactly the struggle. Trying to convince the masses is the wrong way to go about it.

Leigh spoke up. "Interesting. So, what would be the better way?"

"The way I've seen these programs work, in a way that really makes progress and gets traction, is eventually you figure out that you have to go higher," he said.

"To senior management?" she asked.

"Yes," Peter replied, "higher up the ladder. To a level that is higher than the wall. Look at the relationship dynamics at play here. We have this wall in between us. I don't know you, you don't know me, you're just a function. So, the attitude is an 'us versus them' thing. Development is the problem, or QA is the problem. Or the security team is the problem. They highlight security vulnerabilities that aren't really problematic because they don't understand the app. One group is goaled to get the code out quick, and the other group is goaled to make it sufficient quality—or, in my case, secure—which means slowing the delivery. These goals push against one another.

"As you start to get over that wall and start to have names associated with the people in both camps, you can start to align the goals, to get everyone pushing in the same direction. Now, once I have that, we come to an agreement around taking responsibility for the quality of the code. And eventually the wall comes down. And David Hasselhoff sings."

Dinesh and Jessica looked at each other. Casey looked over at them as well.

"He sang when the Berlin Wall came down," Peter said. "It's all over YouTube."

"So," Dinesh interjected, "the focus shouldn't be the troops on the front lines?"

"Not directly, no," said Peter. "What happens is you go to those lowest levels, and they'll just say, 'That's not what I get paid to do, my goals say ship the code fast, and that's what happens.' But you need to go up to those high levels and say, 'Look, this is all about mutual accountability, where if I wrote it, and if it's broken, I should be the one that fixes it.' You have to get to that point of agreement where you have transparency and reporting at that level to say, 'This is what we're going to do now; I am accountable.' When it rests at the higher levels, if things start to look bad, I'm going to look below me and say, 'Hey, what's this next level of management doing to tug in the right direction here? Why aren't we fixing these problems?' Which cascades all the way down to the lowest levels of the organization all by itself. If you just start

at the lowest levels you may get traction across a handful of teams. To get real adoption across the company, you need executive sponsorship."

Leigh thought back to the Hangar meeting and Helen Humphries' threat about certain people being "no longer around."

Peter continued, "The effort needs to be put high up, and not down low. That's really the problem. In most companies it's still upside down. Sometimes you need the low down to show success in order to get buy-in from the executive, but it shouldn't be just that way."

Food was being served now. They took their seats around the patio table.

Steve Fibonacci, ever the genial emcee, offered to make the toast.

"When Leigh came to work for me at FoodFlight, I saw her talent right away. She was a gifted programmer, but she was always talking about the sea. Sharks, in particular. I couldn't see just how a wanderlust for the ocean could lead her into the airline business until I saw the parallel. Or maybe I just created it in my own mind, I don't know. The point is, sharks have to keep moving forward. If they stop, they drown. It's as simple as that. I think you can see that Leigh has that same intuition around the company she works for. So, all that time she spent sitting on a research vessel in the Atlantic Ocean was worth it.

"And now tonight I have had the opportunity to meet her dad, our host, Warren, who seems to have hydraulic oil in his blood. It's people like him who set the bar for quality assurance very high, working with sharp metal and big machines to keep our planes safe literally from the ground up. So, thank you to Warren for hosting us, thank you to Peter for sharing your expertise, and thank you to all of you for just being such interesting people!"

Everyone raised their glasses. They then settled in to enjoy the food. Peter immediately engaged Warren in conversation, asking him about his work in engine manufacturing. Warren happily obliged.

As the meal moved into the dessert phase, Peter stood up. "I have to say as much as I don't want to bring shop talk to the table, I can't help seeing such a perfect parallel between what we're talking about in security and what Warren had to go through in manufacturing. In fact, I use this as an analogy whenever I teach. Now I have Warren to correct me if I say anything incorrect."

"Damn right," said Warren with a smile, and rolled up his sleeves as if ready to take on the QA work of the speech to come.

Peter smiled. "Basically, you can take the tack that when it comes to assembling an aircraft engine, I am going to take all the parts that come in and I'm going to put all of the thousands of hours of labor and hundreds of thousands of dollars' worth of parts, I'm going to throw that together, and then I'm going to put it in a test jig, and I'm going to turn it on. I'm just going to say, 'OK, here we go.' Which is what that quality thing often looks like, right? I'll just

turn the engine on and see what happens. And half the time the damn thing blows up. And the reason it blows up is because some of the sub-parts or sub-assemblies inside were not perfect.

"And if you look at and diagnose where that could have been found, it's usually on the loading dock. If I had looked at the fins from the engine—for the jet engine—I would have found that some of them had little cracks in them. If I hadn't put those parts in in the first place, if I had done that, I would have saved all of the work of putting that thing together to have it blow up or have to tear it half apart to go back to that point to fix those things to put it back together.

"So, even if it doesn't blow up, what ends up happening is you knit the sweater to un-knit the sweater to re-knit the sweater because you dropped a stitch somewhere and you weren't paying attention because you weren't measuring for quality. So, back to continuous testing, by having quality measured in there earlier, I don't have to undo and redo a bunch of work, and that's really where all the cost savings associated with fixing something that you could have found earlier is possible. You get rid of that stuff by saying, 'If I can measure and hear the problems, this is the place I should measure, because it's an easy and simple change to make, and then it doesn't cascade downstream to affect everyone else's work.' Does that make sense?"

Warren spoke up. "I wish I'd had you as a boss."

Peter smiled again. "So, let's talk about roles." He looked at Angela. "In a typical organization, a large organization, you have two key roles that are responsible for security. You're going to have your Chief Information Security Officer, CISO, who sits up at the top and owns everything, owns physical security, owns point security, owns the security of all of the infrastructure, machines, the people, the entry cards, and all those things.

"And then working for that person in a larger organization is going to be an AppSec—someone responsible for the application security. This person deals directly with how we secure the applications, how we monitor them in production, how we make sure that they go through the right gates of governance prior to getting to production, what we do when there is an incident with an application."

Dinesh raised his hand. "So, what is the power relationship—the power structure between security and the rest of the teams?"

Peter replied, "Well, like I said before, we pay these people to work against each other. If you look at DevOps in general, what we do is we say to the developer, 'Ship the code fast,' and then we say to the operations people, 'Make it run smooth.' One person's pushing the stuff out the door, the other one is blocking the doors and saying 'It can't go out yet, it's not ready.' Which is, in essence, the entire problem, the conflicts inside this airline. Any change becomes a disruption to the smoothness, so these two things work against one

another. And because the teams are not co-responsible for them, any resulting dumpster fire isn't the developer's problem. So, they keep making dumpster fires, and they're either unaware or they don't care. And its operations' job to go on their own to figure out how to work around it and make it better."

Angela added, "So . . ."

"So, the answer to all this stuff is DevOps and DevSecOps have to see the entire process as a people problem. You know, like *Soylent Green*. 'It's people!'"

Jessica and Dinesh looked at each other quizzically, and then quickly looked down at their phones to Google *Soylent Green.* Steve leaned across to both of them and whispered, "It's spelled s-o-y-l-e-n-t." He winked at them both.

Peter continued. "So, if you think about this as an 'us versus them' problem, it's not necessarily security against testing. It's security and testing against development. Security and testing are sitting on the wrong side of the wall from the developers. In fact, there might even be some kind of camaraderie between security and testing because they're both facing the exact same problem. They can't get the developers to do the thing in a way that makes their function look good."

Leigh looked at Dinesh, Jessica, and Angela with a pained-looking smile. "Arthur!" she mouthed silently.

Peter continued. "Development does it the way they always have, and then it's quality and security that have to go figure the problem out. And it's my job to secure, but I can't make the changes because I didn't write the damn code.

"I can put tests in place to show them that they're doing it wrong, but I can never get them to do anything about it. So, I would see this not as a conflict between security and quality, but as more of a partnership of trying to figure this out together. Again, we're given responsibility without authority."

Leigh looked at Angela. Again, they seemed to share the same thought: "Oh my god, we have the same problem! Maybe we can fight this together!"

By this time, Copernicus was fast asleep. Casey and Warren were playing darts in the living room.

Leigh spoke up. "So, as we start bringing in continuous testing and we start knocking down the walls, breaking down the silos, everyone now is part of a big team where you have much shorter testing cycles. They'll eventually find that 'shift left' actually makes them faster. Some of those habits can be trained out of developers. You know, like, 'I write it this way because that's the way I do it.'

"Exactly," said Peter. "If you just taught me how to write it differently, then it wouldn't be a security problem in the first place. So, a lot of the things that you get as a bonus from this training is the developers go faster because they're writing it securely in the first place."

"OK," said Dinesh. "So, with these types of improvements to the process, how do you see people viewing their career security if their jobs are changing so substantially?"

"First, it's not an either/or situation," Peter said. "That's what people don't seem to understand. They think 'If you introduce this new thing into my world, it's going to change my job or maybe even make it redundant, so please don't introduce it.' But the point they always forget is, there's no guarantee their job will stay if no change is made. In fact, the likelihood is even greater that their job will vanish or shift without the change. When it comes to change management, they should never think 'It's between this change or no change.' In actual fact, it's about 'this change or this other change.'"

"It's like finding two insects on your picnic table and deciding which one to get rid of first," said Steve. "Always go with the lesser of two weevils." He broke up in helpless laughter for a few seconds.

Peter blinked and regained his train of thought. "So, to go back to what I said before. Security is currently a big ugly bottleneck. We're blocking the door because your product is not secure here. There will be a cascade of phone calls that go up the power chain to find the person that maybe owns the P & L for that product. That person says, 'You know what? I have to make my bonus too. And I only get that if you guys ship this product, because this feature is critical for making my revenue numbers. So, we're releasing anyway.'

"So, if you're serious about changing the status quo, and you look at what should happen to the roles in the lifecycle, security becomes more like partners. They come in and up-train the development teams on how to do it better and faster. So, there's a lot of work that needs to get put into education. That in turn takes security to the next level. They can introduce Red Team and Blue Team—you know, pretend good guys versus pretend bad guys—to help with threat modeling, and in the end the security team becomes more of a partnership-based training opportunity. They can go off and be strategic, to really learn how we can get the company to the next level. The attackers are always ahead of the defenders."

Steve added, "So, if we're trying to protect our company, we need to be out there looking at what's going on in the industry, how attacks are being done, looking at breaches, and understanding that breach data so that we know how people are being attacked and breached. So, we can bring that to bear on our application security programs. They need to become more strategic. And allowing shift left and giving people the right tools and training allows them to be more strategic."

Peter checked the time on his phone. "Anyway," he said, "you've probably heard enough from me. I hope it's been useful to some extent."

The group around the table agreed enthusiastically.

"So, let me just conclude with this thirty-second story. One of the last things I did at my previous employer before I joined Veneto Burrito was to get the security team to evolve their thinking so that they saw themselves as an internal part of IT. Prior to that, security was on a completely different floor from tech. It was a sealed department with walls and locked doors—only security people were allowed in. They had gated themselves, working in a vacuum, because it was thought that such measures were needed in order to focus on cybersecurity and the like. But then one year ago, I convinced the CISO to help me tear down the walls and get security to partner with tech. I told them they needed to engage with us. What we achieved in that last year was to help tech take a lot of their practices and figure out how to automate their security scans and embed security engineers into the pods. It seemed to have worked well."

The small group gave Peter a round of applause, and Copernicus, newly awakened upon sensing the end of the boring human talk, offered him a chew of his new octopus.

Leigh, feeling relaxed after what felt like a successful evening, absent-mindedly checked her email, purely out of habit. What she saw made her stomach drop. It was an email from Dieter Schmidt, one of the team from AbLuft that she had met up with last Wednesday. It read:

"Dear Ms. Freemark,

We are pleased to confirm that we are moving ahead with your offer. Details will be arriving tomorrow.

– Dieter."

The email was cc'd to Alicia and Derek.

CHAPTER

10

Future Proof

They look like sharks, Leigh Freemark thought to herself for the second time in two weeks. She sat at the far end of the table in the executive boardroom on the 17th floor. Facing her around the table were CEO Bob Cartwright at the far end, Helen Humphries to his right, Alicia to his left, then Derek and Arthur. Lester Greene was there, too, as well as two other members of Renway's legal counsel whom she did not recognize.

Honestly, she thought, *I wonder if their eyes roll back in their heads when they eat their Cornflakes in the morning*. She stared past them at the painting of a Renway Boeing 737 that occupied most of the boardroom's end wall. It was a beautiful work in oils, six feet across and six feet tall, highly realistic, and painted from the very vantage point that she had enjoyed just a week and a half ago at the hangar meeting with Helen Humphries. The artist had captured the plane's face, and its huge, smooth body, the delicate texture of the glass covers on landing gear lights and the soft reflection on the concrete floor.

She looked at Alicia and Arthur, but neither of them met her gaze.

"Ms. Freemark," Helen Humphries began, once again not waiting for any official start or opening remarks. "It probably comes as no surprise to you that certain of your behaviors in recent days have caught the attention of the executive leadership team and warrant some explanation. Perhaps you did not fully understand my directive when we last met, regarding finding a resolution to the cutover fiasco. We had not anticipated that you would take that instruction as an invitation to reinvent the wheel and throw our entire IT department into disarray."

Leigh opened her mouth to speak, but Helen stopped her with the most subtle raising of her index finger. It moved barely a quarter inch, but that was all that was needed.

J. Scheaffer et al., *The Kitty Hawk Venture*, https://doi.org/10.1007/978-1-4842-3661-1_10

"Then," she continued, "we started hearing stories about your dalliances with our competitors, including taking a dotPlane flight to Chicago. You must be aware that, given the highly competitive nature of our industry, and the requirement for absolute confidentiality, we treat such unauthorized interactions very seriously indeed."

Bob Cartwright interjected. "In addition, we understand you have obtained intelligence about the potential acquisition of MidWest. We don't know how you found out, but the fact that you did, and your subsequent interactions with these representatives, has the potential for being a very serious offense."

Again, Leigh tried to speak up. Again, she was overruled.

"Since you are apparently already aware of the Kitty Hawk project, I can let you know that in light of recent events, including the cutover fiasco and the discovery of your activities, MidWest has pulled back from the table to consider further options. Please don't think that you alone are significant enough to change the direction of this acquisition arrangement, but it became one of the final straws in their collective basket of concerns. I can assure you that our Board is not happy with this development."

"Finally," added Helen, "there is this email from AbenteuerLuft expressing acceptance of your 'offer.' We invite you now to explain precisely what this offer pertains to and how it came about." She looked at Lester Greene. "Be aware, of course, that what you say in this room will be recorded and reviewed by our internal legal department."

Leigh stayed quiet for a full ten seconds. Long enough to hear some of these powerful people fidget in their seats a little. She didn't know why she did that, especially when she had so much to say. But she did it anyway.

"Firstly," she started, "let it be said, there is no offer. The executives from AbLuft apparently misunderstood my discussions with them, or it was miscommunicated somehow. When I met with them, I spoke in hypothetical terms about how an airline like Renway could form temporary alliances to capitalize on pop-up events like the World Cup. In the past, these types of things would be arranged years in advance, but new businesses like dotPlane are doing these things almost on spec. They're leveraging a new form of commerce that is very quick and interactive and is less focused on brand loyalty than in the past. That, by the way, is why I flew dotPlane to Chicago. I used my own credit card and did not reveal anything about who I worked for. I wanted to see what the customer experience was like and whether it was better than what Renway provides."

She waited for one of them to ask, "and was it?" but not one of them did. Not one. Not even Alicia.

Helen Humphries spoke up. "Need we remind you, Ms. Freemark, that you work in IT? It is not your job to do this."

"But with respect," Leigh replied, "it is. It's everyone's job to do things like this. From an IT perspective—look at the software dotPlane is producing. What about how the customers are interacting with them on their phones? What about their own crew-scheduling apps and reservations apps? These things have to turn around on a dime, which means the app must be architected in a way that allows this, but so too the developers, the testers and operations. They must all be aligned and working together seamlessly. Screwups like the one we had two weeks ago are already starting to have much larger ramifications when the time to repair stays constant but the time for fast turnaround gets shorter. This has become everybody's job now."

There was a derisive snort from Arthur. He leaned across to whisper to Alicia in a way that everyone could hear, "There she goes with that line again. 'It's everybody's job to do this and that and the other.'"

"Ah, yes," interrupted Bob. "Continuous testing. Alicia and Derek have been informing me about this. This, too, is your proposal? To shake up the IT department's tried-and-true methodology for a flavor-of-the-month software trend? Have you also been having lunch with some aggressive software vendors behind our backs?"

"Ah, yes," Arthur echoed, "last Thursday, wasn't it?"

Everyone around the table simultaneously looked wide-eyed at Leigh.

"That was not a meeting with vendors, Arthur," Leigh exclaimed, with anger edging into her voice. "That was a get-together of programmers who are interested in learning about new developments and continuous improvement for their companies. It's called lifelong learning. You should try it some time. You know a lot's going on in the world, and if you just moved your eyes off your screen for a moment you might notice that."

Arthur remained cool, sensing victory at last. "My eyes are on the screen so that our planes stay in the air, Leigh. *You* should try *that* some time."

"OK, OK!" Bob was almost shouting. "Leigh, you can see in microcosm here just what happens when change is introduced into a system that operates on a delicate balance. It has to be done with great care and planning. This is why we were forced to come to a difficult decision."

Leigh steeled herself. *OK*, she thought, *this is it. I'm fired.*

There was silence and tension for a full ten seconds.

"We have decided," said Helen Humphries, "that overall, your assessment of continuous testing is a correct one. We, as the executives, have agreed to start implementing it across the organization. We acknowledge also that your assessments of the ways the airline industry is changing are more up-to-date than our own have been, and we realize that Renway must do more to keep pace with that."

Leigh was amazed. She couldn't believe she was hearing this. She looked across at Arthur, expecting him to be fuming once again, but incredibly, he was nodding in agreement.

"I am aware that you have already introduced the continuous testing concept to the IT team. Alicia has independently assured me that this is the correct route to take, and I will take her word over anyone's, frankly. Once this meeting is adjourned in a few minutes," Helen continued, "I would like you to brief your IT colleagues on next steps, specifically how to make this transition 'future proof,' as I believe is the term."

Leigh's mind was spinning, but again she felt the compulsion to push back. She got a flashback of her early days on the shark-tagging boat, when she was allowed to throw more chum in the water so the team could observe the feeding behavior. She got to chum the placid waters. She was doing it again now.

"With respect, once again," she started, "may I suggest you all attend that briefing? If you have determined my assessments about continuous testing are correct, then I have to ask you to also reconsider the role of IT itself. It cannot remain a silo. It's not just some department full of eggheads tucked away in the basement out of public view. Everything Renway does now is dependent on an IT system that runs correctly. Not just running of its own accord, but running in complete sync with the strategies and initiatives that you develop up here. IT needs a seat at the C-suite table, and, conversely, the executive leadership team needs to know more about what we're doing."

More silent seconds followed. Leigh's mind went to the iconic shot in *Jaws* where Roy Scheider's character finally sees the size of the great white his team has been pursuing and recoils slightly in shock, a cigarette dangling from his lips. *Leigh*, she thought to herself, *you're gonna need a bigger boat.*

"Very well," said Bob, exhaling slowly. "Tell us what more we need to know."

Leigh stood. Her legs felt a little shaky. Not out of fear, just pure adrenaline.

"To start," she said, "what I have learned from my research and my discussions," she looked at Arthur as she spoke those words, "is that to future proof, we must have a strategy, starting now. For all aspects of development. From coding to testing, including functional, performance, and security. Also environment management, from test-data generation to service virtualization. The whole enchilada."

She paused to take a breath and to make sure Bob and the rest of the executives were still with her. "We will need dedicated resources to get this done. It's not something you should give completely over to a third party. We need strategies that align with where our company is planning on going, from a business-strategy perspective, because IT will have to support that at the speed and quality that the business is being pressured by the market to

deliver. We need to work in lockstep; IT is part of the business strategy, not just supporting that strategy, as it is seen today.

"As just one example, my friend Adam, who has the same role as me over at Consolidex, his team is deploying a tool solely to look at test-data management. It's a tool that helps teams get their own data—they call it 'eating their own dog food'—and then roll it out to the testers on teams. Application pipelines need data, of course, but they need it faster. He says also that it's imperative to maintain a clear line of sight to ensure tools used for continuous testing are going in that same pattern.

"You also have to look at your relationships with third parties because they might have a solution today that can be integrated into the pipeline. As we mature and get better, we can expect to build up our own expertise, which means we might not need the same types of iron-clad relationships with these third parties. Things are changing. We want to keep our relationships with third parties, but they, too, will need to evolve. "We need to keep getting better at, for example, building our own lab, or leveraging a solution in the cloud to do cross-browser and cross–mobile device development. He told me that having dedicated teams to build out those platforms and using core metrics aligned to continuous testing, we can keep application quality as a first-class citizen.

She was feeling a little more like herself again now. Alicia spoke up.

"Leigh, can you tell us what your research predicts for the future of continuous testing?"

Leigh looked down at her phone.

"You might not be aware of this," she started, "but in support of this I have already started a wiki. I don't profess to have all the answers yet, but here's what I have put together around the future of continuous testing." She looked down at her phone. As she pretended to type up the URL for the wiki page, she discreetly sent out a text to the one person she knew who would reply quickly. "Urgent. Facing firing squad. Need intel on future proofing continuous testing. Thx!"

She then read out from the text that she actually did have in her wiki. "The biggest thing is testing in production. That's the biggest change. The whole concept of quality engineering will go away to be replaced by testing in production, doing more testing of the open source portion of software, which can become 30 to 40 percent of the software stack, and more non-functional testing.

"Some believe companies might no longer need testers, while others say you will always need both testers and testing organizations. The profiles of a tester and a developer will start to merge. This will require both sets of people to change their mindsets." As she read this, she looked directly at Arthur once again. "The amount of black-box testing will start decreasing while white-box testing will become more prominent and relevant, and testers will need to be able to do that."

She looked around the room. "And as I believe I have mentioned before, this will demand a change in the COE from being a Center of Excellence to a Center of Enablement, and this will further demand change in the makeup of the teams and the testers."

She read through her wiki again.

"Analytics. Big data will be critical for testing, especially getting refinement in code coverage for all types of tests. The market is really exploding regarding coverage of the regression suite, especially in terms of developing a scientific way to know our regression tests achieve the appropriate levels of code coverage."

As she read this, a text came in from Steve Fibonacci, exactly as she knew—or at least fervently hoped—it would. She continued to read as if the text was part of the wiki.

"Companies have to be prepared for change and be willing to go with the change. AI is going to play a larger role. Automating the automation, autogenerating testing on the fly. This will replace regression suites of tests, which will no longer be maintainable. You have to see what the users are doing at any given moment and make tests to validate that. Or figure out how to improve. It will be much more automated. Record and replay will be a thing of the past; there will also be a continued improved focus on getting the requirements correct and on new ways of analyzing them and identifying what's missing."

She looked up at the group. "Is this making any sense?"

Bob spoke. "You are detailing a number of ways that IT has to change how it produces code, but I want you to better connect your vision of the future of continuous testing with the future of continuous air travel."

Leigh paused for a moment and looked once again at the painting of the 737 that dominated the far end of the boardroom.

"Perhaps I can answer that question by drawing a comparison with the Kitty Hawk project."

Leigh could no longer tell if she was actively splashing around in chum-strewn, shark-infested waters or if she was safely up on the bridge. She pressed ahead anyway. "Traditionally, an airline merges with, or acquires, another airline to gain access to additional routes and cut the unprofitable ones, expand its

physical assets, essentially eliminate a competitor, and capitalize on pricing power by establishing limits on seats and baggage. Through this they expect to raise profits and shareholder value and avoid the bankruptcy that is plaguing the industry. Would that be a correct assessment?"

"Please continue," said Helen Humphries, without confirming the question or adding additional comment.

"Well," Leigh persevered, "these are power plays that have been employed by airlines for decades. But in the meantime, the marketplace itself has changed, and, increasingly, external, non-airline players are starting to make a greater impact on how seats and flights and routes are used. This is entirely software based, and its turnaround times are measured in seconds now, not days.

"There comes a point with organizations in all types of industries that the executive leadership team must recognize how fast the world around them is changing. Work is being done with mobile technologies and cloud-based applications, and speed is its lifeblood. I simply see the airline industry and the IT business as both being heavily burdened by tradition and legacy systems that have become like an anchor, and all the while, new start-ups that don't have anchors are moving in."

She felt her phone vibrate. She quickly looked at it, as if checking the time. Steve Fibonacci had sent her a two-second animated GIF. A highway merge lane with queue jumpers. *Perfect*, she thought.

"Have you ever been driving on a congested highway and some nut job pulls out from behind you and rides the merge lane all the way up and then squeezes back in, a few cars further up?"

"I hate that," said Derek, who up until now had been silent. "It's completely against the rules of the road."

"Yet it still happens," said Leigh. "People more nimble and willing to play by different rules are going to continue to get there first. That's what's happening right now with dotPlane. They're squeezing in and taking all our passengers across to Russia for the World Cup because we're still in the slow lane deciding whether to even do this and whether we would have the time to build out a system to support it."

"It's illegal," continued Derek.

"Well, highway line jumping might be, but the principle—the approach being taken by dotPlane—is not. These might be overused examples, but I think they are worth mentioning. Uber is the world's biggest taxi company, and they don't own cars. Airbnb is the largest rental company, and they don't own property. And Facebook is the largest media outlet, and they don't own reporters or writers. So, why would you not think the next 'largest' airline company will not own planes?

"This is what's going to trigger the next evolutional iteration in continuous testing strategy, by the way. Everything is moving toward a seamless end-to-end customer experience. Cloud, DevOps, agile, big data: the end product for all these things is a seamless customer experience. Now we have to decide how to apply that to testing. This is where the two worlds meet.

"It's one thing to validate a scenario, but it's quite another to validate the customer experience—that the user is actually perceiving value that we, the business, intended. This is where continuous testing becomes an innovation enabler. Testing must become a core part of everything we do. When you think about innovation, testing always has to go with it. As much as companies currently put testing as a second-class citizen, it has to now evolve to become the feedback mechanism that will enable the business, us, to determine whether we are achieving the business objectives we invested in.

"Perhaps even more important than ensuring the application is built right is to ensure we are building the right application—something that will solve users' problems and bring value to the customers and that is pleasing to experience. Continuous testing helps with this. By breaking down the units of features to be tested and making them more quickly available to users, it shortens the time gap between design and usage."

She looked around the room again.

"When it's used in conjunction with feature flags and canary deployments, continuous testing presents end users with a constantly updated capability and user experience. And then, based on customer feedback, the new capability can be rolled out more broadly, or it can be rolled back. This type of agility is amazing! It means we can ensure that over time we're building the right set of features and giving customers the most intuitive experience.

"What we know as 'testing' today becomes your compass and sounding board and connects business strategy with business value. Today, we have no way of measuring if what we are building and delivering is giving us any value. And guess what—when Arthur retires, no one will understand our systems. What then? An outage happens, our flights are grounded, and Arthur will be sipping margaritas somewhere in the Caribbean enjoying his retirement. The time to act and change is now."

Alicia, who had been painfully silent for the bulk of the meeting, spoke up. "I think you will find also that newer developments like the Internet of Things are bringing this reality a lot closer to us. Devices are connected in all kinds of ways. Right now, continuous testing is challenging for browsers and APIs, but when you talk about physical and mobile devices and the Internet of Things, integration hubs that allow continuous testing will be the drivers of change. And wherever it is lacking, people will start to notice."

Leigh jumped in. "One of the speakers at the event I went to up the street—he works for a travel portal, Awayz. These are the guys that the customers are going to book vacations and flights from. These are the future of airline commerce. His people are already doing levels of production testing using AI and big data. One thing he mentioned specifically is that you always have to be thinking three steps ahead in terms of things like predictive analytics. Big businesses like ours, and others like healthcare, are being forced to move toward a whole new retail model.

"What I saw in Chicago, too—there's a survival-of-the-fittest mentality now, more intense than ever before. Top-notch QA and QE folks are going to evolve quickly. Their skill levels will get higher and the skillset needed for continuous testing will be very different than that required today. It will be an overarching level of skillset—wider cognitive bandwidth to leverage the power of AI and IoT. COEs will need to figure out how they are going to support that going further."

She looked at the group. They remained motionless and focused. *These people*, she thought to herself, *have made a career out of attending meetings. They have incredible stamina for sitting down*.

"I have one more thing for you," she said as if laying one more sacrifice on the altar of an unreadable, dispassionate god. "Look at mobile. Companies must put mobile first. If you don't do continuous testing with mobile apps as a priority, there will be an immediate negative impact on customer experience, and consequently on our ability to generate revenue. It's a simple star-rating paradigm. Consumer confidence will decrease once they remember that the app did not work, no matter how good our planes and crew are. They have expectations that their app will be updated proactively, and it's a substantial risk to our reputation and company image if we can't keep pace. And then, beyond mobile first, we must achieve omnichannel excellence. That means we must have adopted DevOps practices in order to effectively handle the multiple channels. And continuous testing underpins all of that."

Helen Humphries spoke. "Thank you, Ms Freemark. You have indeed shed some light on a pressing issue here at Renway, and I know that Bob and the entire executive body are grateful for your wisdom. This is something that we will be focusing on more intently going forward. It represents a momentous change in approach for us, and it would certainly benefit from the type of vision that you seem to have in infinite supply."

Leigh relaxed a little and smiled. "Thank you, Helen," she said. "It's simply what I think is right."

There was a further silence. Leigh felt that a shoe somewhere was about to drop.

Helen continued. "However, there's also another issue to consider. As smart as you are, Ms. Freemark, which might almost be as smart as you *think* you are, and, as important as these innovations might be for Renway, our company has no tolerance for loose cannons on deck.

"Your behavior, in meeting with other airline representatives, flying on other airlines without notifying your manager, and generally playing fast and loose with the long-established protocols and culture of Renway Air, is something we cannot tolerate in any employee. Your most recent actions posed a distinct security threat to our company. There is simply too much chatter going on between you and outside agents, and this constitutes grave violations of the terms of your employment and could easily constitute grounds for immediate dismissal. Furthermore, you appear to be losing the confidence of your team, and this does not reflect well on your capabilities as a manager and leader of others. Renway is a forty-year-old company. We are proud of our traditions in the civil aviation community, and we work hard daily to maintain them."

Helen Humphries paused again, and repeated herself, putting emphasis on the words. "*Maintain them*, Ms. Freemark, not change them on a whim like fashions or pop music. You may think it's admirable to be a free-flyer and a risk taker, but we insist our employees, everyone who wears the Renway badge, keep their feet on the ground. Our business might be about moving people through the sky, but ultimately every flight's success is based on having those planes safe back on the ground. Tradition, Ms. Freemark. That's what *make*s a company."

Leigh simply stood there, feeling slightly numb. For some bizarre reason, her mind flashed back to Adam's DOS Prompt Open Space meeting, the 1970s Ford Econoline van image, and the lone voice at the back of the room shouting approval. *People living in the past*, she thought.

"Now," Helen Humphries continued, "as you see, Lester and his team are present and have been observing this discussion. They have scheduled time for you to discuss your recent actions, and they will advise me as to whether any punitive actions against you should be pursued. In the meantime, I expect you to work more closely with Derek and Arthur to fully project plan the transition into continuous testing. Thank you, Ms. Freemark; you may now leave the meeting."

Leigh could only stare in disbelief at Arthur, who met her gaze for a moment and then started to fold his newspaper and slip it into his briefcase. He showed no hint of anger, satisfaction, sadness, anything. Just blank.

She stood and nodded silently to the executives before turning to leave the boardroom. As she left she could hear Helen Humphries' voice, "Now, to other business . . ."

* * *

As she waited in the 17th floor vestibule for the elevator to take her down to the dungeon, Leigh checked her phone. There was a text from Dinesh from two minutes ago: "Coffee? Perlman Building. Right now?"

"Sure," she texted back. *And why not*, she thought to herself. Ignoring the ding of the arriving elevator, she walked to the end of the vestibule, where a floor-to-ceiling window gave her a view of a clear blue autumn sky. She swiped across to her calendar. Her day consisted of a teleconference at 2:00 and a pile of emails to return. She looked out and up at the sky and then back to her phone. She deleted the 2:00 appointment and then created a full-day block marked "Unavailable." She quickly typed out an email, hit Send, and then pressed the elevator button again. When it arrived, she did not press "B1" for the basement. She pressed "G" for Ground.

She stepped out of the elevator into the lobby. "There you go, Helen," she said quietly to herself, "here are my feet, on the ground. She walked through the big glass doors and leisurely across the parking lot to her old Porsche. She unlocked the car door, got in, and fired up the engine. As she checked behind her to back out, a movement caught her eye. A familiar figure was walking slowly toward her. The angular and mildly arthritic walk and the silver hair left her no doubt. It was Arthur." Why did he leave the meeting so quickly? "Was he coming for another twist of the knife? Was he truly not finished with sabotaging her career?"

He walked straight toward her car, and then he walked past it and unlocked the door of the car parked next to her. A Buick Roadmaster, at least 15 years old. Arthur's car. In all these years, they had never once encountered each other in the parking lot, either when arriving at work in the morning or leaving for the day. And yet here he was, seemingly unaware that she was sitting in the car next to him. She could have just let him back out and avoided any confrontation, but what the hell. She shut off her engine. "Arthur!" she shouted through her passenger window.

Arthur looked up and recognized her. A quick shock of fear passed over his face. Maybe he thought she would come over and punch him in the nose, right through the glass of the Buick's side window. He rolled his window down, and Leigh hopped into her passenger seat so at least they were talking side-by-side.

"Did they fire you?" she asked sarcastically.

"No, of course not," he said. "I have a doctor's appointment."

"Well," Leigh replied, "looks like change is finally descending upon Developers Kingdom. You have my condolences."

"I wouldn't worry about sharing condolences, Leigh," he said. "I don't actually care. Look. You were half right about retirement back there. I am two years away. When that day comes, I am going to leave all this software stuff behind, I am going to leave this building. But I'm not going to the Caribbean. I'm going to stay around here and open up a small store fixing grandfather clocks, especially German ones."

"Why German ones specifically?" Leigh asked.

"Because that's my heritage. I have had German and Swiss clocks around me all my life."

"You're German? Really?" asked Leigh. "I always took Fergus to be a Scottish name."

"It is," Arthur replied. "My father's name was Schmidt. Willem Schmidt. He died when I was about four. My mother remarried—a Scotsman—and we took his family name. But I'm a Schmidt. A Smith. A craftsman. That's what I have always been. I hired Owen and Giles, and now my days mostly consist of making sure they don't kill each other, or if they try to, that they don't damage the equipment in the process. But I'm not a manager of people, Leigh. I like to work on detailed structures."

He laughed. "Someone like you, you'll do well anywhere, Leigh," he said. "You have the instincts of a sailor or a pilot. I can see that." He waved, just a small movement of the hand, and backed out of his space.

* * *

Leigh drove the four blocks to the corner coffee shop where she, Jessica, and Dinesh had recently met up with Steve Fibonacci. She found a curbside space close by and paid the meter. *Thank goodness for meter parking apps*, she thought. *Who carries change anymore?* and then, sardonically, *I bet they do continuous testing already*.

Jessica and Dinesh were waiting for her on the coffee shop patio. There were three coffees in front of them.

"Is someone else coming?" Leigh asked as she sat down.

"No," Jessica replied, "that's for you. I just got it. It's still hot."

Dinesh looked at her car. "You drove four blocks?" he asked.

"Yup," Leigh replied, without further explanation.

"We are so sorry about the dressing down you probably got this morning," Dinesh continued. "One of the marketing admins heard about the meeting agenda from her friend over in Legal. Did they crush you? We think Arthur and Derek had it in for you from the start."

"They just clipped my wings a little," Leigh replied. "Anyway, I said what I had to say, did what I could, and it seems they're going ahead with the continuous testing rollout. They pumped me for all kinds of information first before cutting me down to size."

"Well, we have some news of our own," said Jessica. She looked at Dinesh and then back at Leigh. "We've been pretty disillusioned by the atmosphere at Helen Humphries Airways, which is what it really is. It's her family's money

behind it. It's her family on the Board," she said. "So, Dinesh and I decided pretty much at the same time, this company is not moving fast enough or willingly enough for us. We made our decisions independently, but as it turns out, well, here . . ."

She and Dinesh both handed Leigh their new business cards. They were identical. At the top left corner of each was the inverted rocket fork of Veneto Burrito.

"Steve approached us both a few days after we met him here," Jessica said. "He said he liked what he saw in us and offered us pretty much the same positions we have here, but with a heck of a lot more leeway to make things happen. It was pretty much a no-brainer for us."

Leigh smiled. "That's why you took the Veneto Burrito bags from the meeting."

"We were both hugely impressed with what we saw in the company," said Jessica. "It was a coincidence, but we both sensed this was a company that was about more than just food."

"In fact," added Dinesh, "the toughest part was going to be informing you about our departure."

"Which brings us to this," Jessica added. She handed Leigh a third business card. It was also a Veneto Burrito card, but this one read, "Leigh Freemark, Senior Director, Innovations."

As she did this, Dinesh keyed something quickly on his phone, and sure enough, a text from Steve Fibonacci appeared on Leigh's phone seconds later.

"Leigh—Am not sorry to hear about the developments at Renway. Have they fired you yet? If they did it would be their loss. I remember in my earlier years being fired from my job at a community newspaper. They told me I was a lousy copy editor. I said, 'What am I going to do now with all these mouths to feed?' Hahahahahahaha! Anyhoo, hoping you will consider putting that business card to good use. Would love to have you on the point.click.burrito team as we search for new horizons. You were always able to read the wind, and weather the storms so well, even when you were surrounded by sharks."

Leigh smiled. She put the card carefully in her pocket.

"So? What do you think?" asked Jessica, with barely contained excitement.

Leigh looked at them both. She had a twinkle in her eye.

Leigh was about to answer when her phone buzzed again. It was an email from Sylvia Finch.

"Just received your email. Let's have lunch Monday—Sylvia."

She put her phone away and finished her coffee.

Jessica and Dinesh continued to stare at her with undisguised curiosity.

She smiled again. "That's why I drove here, guys," she said. "I agree with you. I'm not going back there either. I'm going to meet Sylvia at Aurora on Friday, and, of course, I will talk to Steve Fibonacci. And once I decide, you'll be the first to know."

Somewhere down the road, Leigh could faintly hear a street performer playing a song by Lenny Kravitz; he was singing about wanting to fly away, which reminded her that her own journey in continuous testing was only just beginning.

"So, that's it? You're leaving Renway?" Jessica exclaimed, wide-eyed with admiration and excitement.

Jolted back into the conversation by Jessica's inquiry, Leigh nodded. "I'm now on long-overdue personal time. Thanks for the coffee, guys. Again, you'll be the first to know. Right now, I'm going to pick up Casey and Copernicus and take a few days' vacation."

"That sounds so great!" said Jessica. "Where are you gonna go?"

"I'm staying close to the ocean, naturally," Leigh replied, "but we're going to head down the coast for a few days in North Carolina."

"Nice," added Dinesh. "Anywhere in particular?"

Leigh smiled again. "Yes. I have a desire to see Kitty Hawk."

CHAPTER

11

Epilogue

Continuous Testing in an Age of Canvas and Wood

It has been well over a century since Orville and Wilbur Wright flew their craft, named *Flyer I*, along the dunes and beaches around Kitty Hawk, North Carolina. Although other people were also working on powered flight technologies at that time, history has anointed the Wrights with the honor of being "first in flight." Though they made many attempts, it was on December 17, 1903, that the brothers successfully flew their spruce craft in a controlled and sustained fashion. The first of these flights, at 10:35 a.m., went just 120 feet, about the length of the Boeing 737 that Leigh encounters at both the start and the end of this story. In recognition of the Wrights' success in freeing humanity from its terrestrial bonds and moving us all into a new and exciting future, Leigh drives away from Renway Air at 10:35 a.m.

The Wrights' story is one of continuous testing in an age of canvas and wood. They faced obstacles on all sides. They pulled from a long tradition of mechanical engineering, especially with bicycles and small engines. They rejected the established mindset toward conquering gravity with brute force—using more powerful engines—and instead focused on integrating concepts of control across three axes: wing-warping for lateral motion, forward elevator for pitch (up and down), and rear rudder for yaw (side to side). These were traditionally siloed activities that the brothers successfully pulled into a common area of human control to capitalize on the physics and elegance of lift and momentum.

J. Scheaffer et al., *The Kitty Hawk Venture*, https://doi.org/10.1007/978-1-4842-3661-1_11

Innovation and technical advancement deliver challenges to every person who seeks them. Some of these are physical in nature, while others are purely cultural and human. This book focuses on its own three axes: technology, company, and individual. Change crosses these axes with equal dispassion, yet the three must interact, like the Wrights' own system of pedals and wires, to gain lift and momentum.

Orville Wright lived to see his invention demonstrate aerial feats in theatres of war, carry paying passengers, and deliver atomic bombs. He also experienced the earliest moments of the supersonic age. Still, that rate of change, as dizzying as it was, was nothing compared to what we are all experiencing now.

The primary goal of this book was not to explain what continuous testing is. That's secondary. Of greater importance is the understanding of the culture of change that must exist within the very fiber of every company and must be embodied in the collective thoughts and actions of every individual, from the most senior executive to the newest hire. Without this spirit of adventure and change, the catalyst might just slip between a company's fingers, seeking bluer skies elsewhere. As the Wright brothers themselves knew, flight was an inevitable consequence of the advancement of technology. But who achieves it first? That is dependent on factors much more subtle and human than mechanical objects and physics.

APPENDIX

A

Flight Plan for Your Continuous Testing Journey

Resource Guide

Why Continuous Testing is Imperative Now

A flight plan, like any other project, must start with a justification. With every flight comes risk and expense, and it must be understood by all why this flight is needed at this time. Is this flight part of a larger venture, delivering people to a location to pursue business or leisure activities? Or is it a rescue mission? Is now the right time? Are crew and facilities in place? Is there redundancy built into the plan to allow for contingencies and change? Are we agreed that now is the time to take off?

As the pace of business continues to quicken, companies are starting to recognize that to stay competitive, the process of developing and releasing software needs to change. Release cadence has greatly accelerated. There is no occasion anymore for a six-to-eighteen-month *find and fix* turnaround in which the customer will find the delay acceptable. Things need to move faster, and they need to be ready and perfect faster.

J. Scheaffer et al., *The Kitty Hawk Venture*, https://doi.org/10.1007/978-1-4842-3661-1

A sea change is occurring in which the responsibility for testing is now being shifted leftward, closer to the beginning of the software development lifecycle and then all the way along it. This is knocking down the wall that has traditionally separated developers and testers, making quality everyone's responsibility. Testing has become more imperative because the consequences of not doing it properly have become more visible.

This is a technical change that in turn demands a cultural change. As testing evolves and spreads out across the SDLC, companies and their IT departments must find a way to change the entire culture of how software is developed and tested.

Maybe most simply, we must stop thinking about testing as an event. It is not something to be done at a specific point. Instead it must be done by everyone all the time, even before development starts. Defects and problems must be nipped in the bud, not caught/missed – as the product comes out the chute into the customer's lap.

DevOps is about breaking down those walls between Development and Operations to ensure that software, hopefully built with Agile methodology, can be deployed quickly to the users as soon as it is ready, without sacrificing quality. But even where DevOps exists, companies continue to experience a trade-off between quality and speed.

In current software engineering culture, 63% of software development delays occur in Test-QA practices across the lifecycle, and 70% of testing is still manual. But this is no longer practical. You cannot just run tests manually, and there are additional problems to consider:

- 56% of critical dependencies are unavailable
- 50% of time is spent looking for test data
- 64% of defects originate in the requirements phase

Failures, especially those that reach the public, are damaging to data, business processes, customer adoption, retention, and brands. It is not possible to turn software around with both speed and quality using outdated waterfall or even Agile processes.

Something more comprehensive, dynamic, and fluid is required to ensure software is developed and deployed with the quality built in and errors detected from the very start, rather than relying on assigning verification at the end of the lifecycle. This is why we need to embark on a continuous testing journey.

What is a Continuous Testing journey?

It is the ongoing effort of transforming the SDLC to embed quality gates in every step of the application pipeline, as it is being defined and built up, to ensure any unexpected behaviors are uncovered and fixed as soon as they are injected.

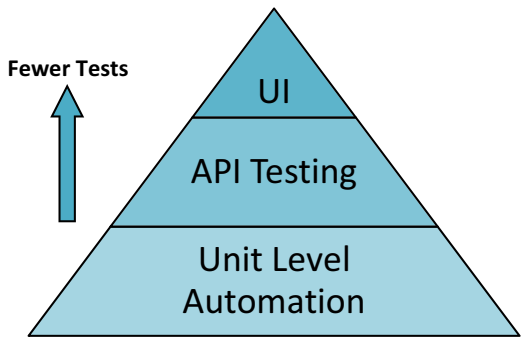
Challenges to Achieving Quality

A flight plan must incorporate all possible challenges, including weather, mechanical problems, crew and passenger problems in the air, and events that might be happening on the ground. For safe flight, redundancy is paramount. Checklists and clear repetitions of spoken commands equate to a process of continuously testing the procedure of flight: cross-checking and the physical placement of controls, are just two of many procedures designed to anticipate and avoid every challenge, and keep the process error-free.

Life has never been easy for IT. It is a department filled with craftspeople and dedicated engineers striving to develop and build a product that works according to the specs – specs that might not always match up to the expectations and limitations of its human end-users. The roles of software have changed. The products must be accessible on a wide range of platforms and must deliver a constantly shifting collection of services.

Quality suffers when we put walls between the people needed to test, develop, and create the requirements for the software. IT should no longer be seen as back office work, as a platform for the business to run on. IT is the company's business and must hold its own seat at the decision-making table.

Quality also suffers when unrealistic deadlines dominate, and capabilities are squeezed into a release without using adequate metrics to delineate quality.



Test Automation Pyramid

Testing has traditionally operated in a silo, which has resulted in an underwhelming degree of robustness. Testers and the business ask, "how good is my testing? What coverage do I have in terms of requirements coverage? What is my functional coverage, not just my code coverage? What are my acceptance criteria? Am I actually testing for them effectively? Many testers don't know.

To ensure quality at speed, testers need to aim for a large number of tests at unit level, a smaller number at API level, and an even smaller number at UI. Ultimately testing should become parallelized or concurrent (test functional, performance and security at the same time), and then shifted left to ensure it all happens continuously, from the beginning of the lifecycle.

Too often, people spend time in non-value activities, testing the wrong things or the wrong set. A team of eight people might say "we're great," but typically 40% of the work must be retested and fixed, meaning a team of eight is doing four people's worth of work, often a couple of months behind. They need to perform a Value Stream Analysis to ensure testing is being done the right way. Value stream analysis is a method of analyzing current processes and identifying a better, more effective one, using lean management ideology.

The tools used in most organizations are "legacy." They sufficed for waterfall, but they don't work well for testing to shift left. They are an impediment to testing at the speed of agile. They don't support automation in the right way for a continuous integration environment and they are tools that developers refuse to use. There needs to be a new set of tools that won't get in the way of speed and quality; tools that enable quality to be built into the process and not "just" test the application as in the past.

But it's not just testing automation that has to change, the requirements definition process must be improved as well. Requirements must be unambiguous, complete, and testable. Currently 64% of defects originate in the requirements phase.

Very often requirements are specified in a high-level fashion that is good enough for a developer and tester that have the domain subject matter expertise to make their interpretation and fill the gaps in the requirement. However, that leads to team members interpreting the same thing differently, introducing defects in the lifecycle before a single line of code is written. On the flip side, requirements can be specified in an extremely detailed fashion, dozens and dozens of pages long. In that case requirements take too long to be specified and are very hard to maintain over time. Not to mention developers and testers find it hard to grasp so much information bundled together in a giant document. A better way for specifying requirements must be used, one that works for Agile teams adopting DevOps.

Quality Assurance must also be approached differently. It must start with senior leadership, who must keep the focus on quality and not just delivery. This is because delivery is now about the total user/customer experience, not just the act of deploying or releasing software. This sets the stage for delivering the right set of requirements that help developers and testers understand what they are building. Now, with releases happening more frequently, potentially every month, week or day, proper and efficient feedback loops are critical. QA must no longer be seen as, or act as a second-class citizen simply because it has, to date, been identified as a shared services property.

To accomplish this change, cultural and organizational changes must occur. This is a human issue rather than a technical one, but it is essential to a successful continuous testing journey. With any technological or process change such as continuous testing or shift left, some people can handle the change and others cannot. The change also threatens the Centers of Excellence (CoE), as testers question how "testing" can be handed over to people who are not trained.

This is chiefly why the acronym CoE must be redefined, from "Center of Excellence" to "Center of Enablement". Its purpose must also change. Instead of highly skilled testers that do the work and are the bottleneck, they must engineer, setup and build the right processes and tools to assess quality across the SDLC. Lastly, they must teach and enable everyone in the organization to use those at scale. This needs change management support from the top down as well as from the bottom up.

As functional testing shifts left, so too must performance and security. There is currently inadequate security testing before deployment in production, and it is time to treat security as a first-class citizen, incorporating specialized security testing practices into the continuous testing process.

Once an application is in production it may turn out that it is not performing correctly under actual user traffic. More horsepower might seem like an adequate quick fix, but this is only achievable within a cloud-based infrastructure. Consequently, there is a need to ensure application performance is tested prior to production, starting all the way to the left, at the unit level by the developers, and then incrementally added to the performance testing scope as units of code start forming larger components that will realize stories, features and epics, always leveraging the smaller performance tests that have been built previously. Companies need to use the proper tools and processes to ensure this happens.

A change in culture demands top-down and bottom-up change initiatives paired with great leadership. Overall there needs to be a continuous testing culture. Executives and IT managers must talk to testers and developers and focus on building a plan together. Make them part of the solution by involving them in the initiative. Give them the chance to fail and learn from it.

What Constitutes Continuous Testing

A flight plan is filed before every flight to ensure air traffic control knows exactly when, where, and how a flight will proceed. Air traffic controllers don't just work at airports; there are hundreds of handoff stations all along flight paths who take over communication as planes pass through their airspace. These local ATCs help monitor the health of individual planes as well as other air traffic, and local weather conditions. They will guide an aircraft in crisis to the closest local airstrip that meets its needs in terms of weight bearing capacity and runway length. A flight is being continually tested by a large group of specialists from long before passengers embark until long after they deplane.

In continuous testing, everybody codes and tests. The team is individually and collectively responsible for engineering and delivering high quality software. This highlights an evolution from waterfall to scrum fall to agile to the merging of agile and DevOps to business agility, enabled by continuous testing.

Collective responsibility uses retrospectives to inspect and adapt the process and its outcomes. This redefines the definition of done. Creating a continuous testing mindset means providing developers with quick feedback to ensure their code is working so that it can move through the release pipeline fast. It means creating or finding data to feed all the functional and non-functional tests running before production and in production.

The three key items that constitute continuous testing are:

- PEOPLE: Empower developers with testing capabilities. Agile teams need to understand testing is part of their responsibility
- PROCESS: Automate everything. Including result analysis
- TECHNOLOGY: Provide testing solutions that developers will use. This means solutions offered 'as code' and hosted in the cloud

An organization must ensure it has a universally understood definition of continuous testing. It is still relatively new and like many other areas of software development, definitions and practices may vary.

Continuous testing is the practice of testing across every activity in the SDLC to uncover and fix unexpected behaviors as soon as they are injected.

A good analogy is to think of an automated software assembly line that includes testing at every step, beginning at requirements gathering and then automatically triggered upon code check-in all the way to production." It includes all aspects of testing all the way up to integration and performance testing and monitoring in production. Continuous testing should be an

effective and efficient application-centric test regime that is end-to-end in scope - across CI, CD and Production. Its goal is to be as efficient as humanly possible to getting changes into production in the shortest lead time with highest quality and top customer experience.

The 11 Disciplines of Continuous Testing

Charting a flight plan requires essential steps or disciplines to ensure you and your crew know as much as possible beforehand. Although the individual activities involved in flight number in the hundreds, there are ten essential disciplines to charting a cross-country according to Visual Flight Rules (VFR): choose your destination; choose your route; get a weather briefing; choose an altitude and cruise profile; compute airspeed, time and distance; familiarize yourself with the airport; double-check your equipment; get an updated briefing; file a flight plan; be prepared for the unexpected - www.thebalance.com

Accelerating and improving a code release process through continuous testing requires a set of disciplines to ensure that quality keeps pace. To embed quality in applications, teams need to add more modern testing practices in the form of small quality checks performed throughout the application pipeline. This would enable small sections of code to be constantly tested.

Test automation is not always the first thing to be tackled. The reason is because even after API tests and GUI tests are automated and integrated into a continuous integration agent, those tests have dependencies – test data, interfaces, and environments - that have to be satisfied before they can be executed. You must:

- Remove environment constraints and free up developers and testers to do their thing, even if manually. Virtualize any and all interfaces you don't own or control, or don't want to test
- Utilize ephemeral testing environments
- Automate test data provisioning and management to feed your tests, manual or automated, on-demand
- Automate your tests so they can run against your virtualized interfaces using the appropriate test data
- Have your pipeline orchestration engine setup in such a way that there is no manual intervention on the steps above
- Next, focus on the complementary disciplines

The 11 Disciplines of continuous testing are as follows:

1. **Virtualized Environments:** Continuous Testing means testing more frequently, which means hitting multiple environments more frequently. This can become a bottleneck if those environments are not available all the time. Some environments are accessible through APIs, others through various messaging interfaces. Those environments could be built with modern architectures, while others are monolithic legacy client/server or mainframe systems. The challenge becomes coordinating testing through multiple environment owners that may not always keep their environments up-and-running for you to test against. Virtualizing those environments allows code to be tested without having to worry about areas that are not goignt o be changed, like other systems and environments. Making those environments ephemeral and available on-demand through virtualization, removes that constraint from the development lifecycle making them always available. You control them – you can spin them up as needed.
2. **Test Data Management:** You must have the right data and the appropriate diversity of data for positive and negative scenarios. You won't find all that diversity in production, that is critical. Therefore, synthetic data generation is the way to go to achieve continuous testing with the highest levels of confidence that no personally identifiable information is at risk in the data. Also, you won't be able to bring data from production, condition it and provision it at the speed your application pipeline requires.
3. **Test Automation:** In a fully orchestrated application pipeline, today's scripts will fail due to things not related to the application code. So, no one will trust the results. You need reliable test automation to achieve continuous testing.

4. **Pipeline Orchestration:** The backbone. Everything is tied to it. It must be integrated with your automation suite. You must understand how it works, how to interpret results and how to make it scalable. This sets up continuous testing in a manner where you integrate continuous testing attributes within a pipeline. Make sure it's transparent and everyone has full visibility into what's being run through the pipeline. It's an automated workflow tool that will run all of your automated tests fully integrated with code deployment activities as it moves through the pipeline. As part of any DevOps adoption initiative, it is unrealistic to expect to get to continuous testing without the reliability and speed of a standardized and automated pipeline.
5. **API Testing:** This will enable alignment to the testing pyramid. Organizations should strive to test as much at the unit and API levels as possible and minimize reliance on UI testing. To achieve continuous testing, you have to embrace the concept of the testing pyramid properly, strengthen unit and API testing, and reduce reliance on UI testing, especially for business logic testing.
6. **Performance/Load Testing:** Even when applications are functionally good, you must radically change the way you think about performance testing. As you shift testing to the left, you're testing earlier in the lifecycle, with less application and infrastructure components, which means the tests are smaller by nature. But they're also much greater in volume. Make that capability accessible to everyone across all agile teams. That means all developers and testers alike have to be able to create a performance test and run it by themselves, without having to send any requests to anyone.

7. **Security Testing**: Developers need to be clear on what the expectations are around the state of the code in order to be allowed or denied to be released. They must receive proper training and tooling to measure how secure their code is. Education of developers must be ongoing as the application security area is always evolving. The application pipeline must be setup to automatically run static analysis, dynamic analysis, and software composition analysis, so that any security issues are caught and highlighted, then those must be used as an opportunity to make developers better. Security testing needs to happen at every step of the lifecycle.
8. **Acceptance of Test-Driven Development and Behavior-Driven Development:** Take the acceptance criteria for each user story and create the tests to ensure those were met. This keeps testing focused within the sprints. This ensures developers develop what the business expects. Over time, teams will define much more detailed acceptance criteria, which required tests to also be more comprehensive.
9. **Automated Test Generation:** The activity of manually designing and writing manual tests or automated test scripts is a bottleneck. Automatically generate the new acceptance-level tests for manual and automated testing in the beginning of the sprint, so when developers start checking in and merging the first working pieces of their code, we can start running our tests in an automated fashion. This gives the ability for developers to get immediate feedback on the quality of their code as they check it into source control.
10. **Requirements Engineering:** You must include all SDLC stakeholders into the continuous testing journey. That means finding a better way to communicate and collaborate on the requirements. The earlier you include requirements engineering into your continuous testing initiative, the smoother the continuous testing journey will be, because team members will be working with the same understanding from the beginning.
11. **Feedback Loops:** These are critical to successful continuous testing. Dashboards in real time. Must be automatic and entire team must have access. Feedback loops across the entire SDLC, not just production, should be used as a compass to help you navigate your continuous testing transformation.

Accelerate Into the Curves

At some point, your flight plan transforms into commitment, as you hit takeoff speed. Pilots refer to this as V1, at which you no longer have the option to stop, but must commit to moving to Vr speed, at which the command is given to "rotate" the aft yoke to raise the nose and accelerate into flight.

Moving toward a continuous testing culture is a critical component to enabling DevOps, but it means committing to a new area of knowledge and practice. This requires some getting used to, but it is backed by solid science and practical knowledge. Underpinning the urgency of this type of improvement initiative is the provable fact that customers, in both the retail and B2B sectors, have an extremely low tolerance for poor experiences and will influence a company's success much more quickly and visibly.

Some of these new practices may naturally run counter to the existing culture, but such is the nature of change and innovation. Teams need to be introduced to these new practices and trained or assisted in managing the transition.

For example, when coding for a feature, there should be release notes embedded in that code, so that anyone can read it. Just as in the world of project management, create a plan that is good enough for someone else to take over as needed.

Quality Assurance (QA) should evolve into Quality Engineering (QE) as a Center of Enablement. This is a place where teams build into the software the knowledge of how to test, deploy, monitor, and even heal itself so that application teams can leverage that within their scrums, without having to depend on external staff.

Everything should be treated as code so that it can be controlled through CI in a standard and scalable way. This requires teams to think about how it will be tested to ensure that, if the code is to be integrated into the source code repository, it will do what it was supposed to do and will not cause adverse effects in other parts of the application.

The people involved, developers and testers, must become part of this new culture. Organizations hiring talent should look to hire engineers who can wear many hats. These organization should also help and enable existing staff to be retrained on skills that better support this new culture. This could include a matrixed organization that allows the app development team to hire or use the skills it needs as it needs them. Change management with humans is as vital as the physical transformation to continuous testing.

From a platform standpoint, cloud becomes an enormous success factor. A continuous testing journey can be tremendously accelerated if you leverage the cloud instead of completely relying on legacy and on-premise environments.

Where Am I in my Continuous Testing Journey?

Ideally a flight plan should unfold as expected, with little deviation. Regardless, constant reassessment is needed between crew members and ATC to ensure the craft is traveling where it is supposed to and is clear of all dangers. This is an in-flight maturity model – constant verification of current status, with updates and changes applied as needed.

There comes a point early in the continuous testing journey where the leader and stakeholders must pause and ask themselves "where am I along the deployment timeline?" It is vital to first understand the imperative for continuous testing and then to adequately define it, but in preparation for launch, one must ask where the team currently stands. Two critical points to bear in mind at this point are: 1) to always put the customer at the center of the change process and its resulting continuous testing activities; and 2) to determine how to measure quality appropriately.

There are new techniques to adopt at this point and in addition, there are barriers to achievement to identify and eliminate. The techniques to adopt are as follows:

1. Improve the relationship between each tester and each developer. Keep the teams small, encourage inter-team collaboration, and make reports easy to access and share online.
2. Make automation a priority. Do not remove all manual testing tasks, but adopt an "automated first" mindset. Focus that on areas that will be run repeatedly. In essence, set up the plumbing first so that you can let the water flow later without any worry or concerns about leaks. Map out your SDLC and identify automation opportunities.
3. Get small. Break work down into smaller increments that are easier to test after coding and design. This makes it easier to automate the tests, and it also makes things much more easily deployable.
4. Keep track of everything. Use metrics and set pass-fail criteria. Continuous testing is all about immediately identifying if things are working or not, so make sure you can set that up easily.

5. Get the right continuous testing tools. Find the tools that will help you to develop, test, and analyze continuously. Pick the best of breed tools that work together, in a way that incorporates easily into the working environment. Choose tools that have community-based support where everyone shares their challenges, solutions, and interesting use cases.
6. Develop a system to display your results. Do deep dives into results, because that's how you will know if your code is working and where the gaps are. Define your KPIs and acceptance criteria and make them quantifiable. Create dashboards to track the KPIs, including a baseline and subsequent changes.

The transition to continuous testing can appear onerous but is best achieved by starting with one small project, bring it successful fruition, and then and building on other small projects. It's an iterative process where going for small wins and building upon them generates momentum and knowledge.

Failures will happen but when they do, teams and leaders should be prepared to fail small and fail forward – take the time to learn from the experience.

Senior leadership driving accountability to the organization will make a substantial difference. An individual evangelist can only connect to 20-30% of the population. For the rest of the organization, it's up to leadership to set expectations and hold the course.

The continuous testing transition will have its barriers, and these must be identified individually. They may include the inability to access and use open source. The work culture – not just people, but practices too – might not be sufficiently prepared. There will likely not be enough time to start the implementation, training, and transition towards continuous testing. This may be paired with inefficient leadership, and the existence of legacy applications.

The ultimate goal at this stage is to develop and deploy a continuous testing maturity model. That means establishing the right mindset in terms of attitude and aptitude. Build up test automation at the unit and API levels for both functional, performance and security testing, and keep test automation at the UI level to the minimum as scripts are very brittle as UI changes. Add a feedback loop that includes performance and user monitoring tools in pre-production environments as part of the pipeline. Full telemetry accelerates defect resolution.

All these capabilities must be fully integrated, collaborative, and not siloed in any way. Companies must be able to accept and understand open source, since it allows people to experiment and fail faster without having to justify expenses. They should also ensure adequate access to test data, since this is one of the most significant barriers to completion.

Metrics – Nothing's Worth Doing That Isn't Worth Measuring

Metrics are used to measure progress. In-flight metrics include airspeed, altitude, distance traveled, fuel remaining and status of all onboard equipment. Standardization of all metrics used, such as feet and miles, are vital to knowing – without misinterpretation – the status of the flight and the aircraft.

Every project requires metrics to determine progress and quality, and to identify business value. This is no different for continuous testing. In fact, continuous testing could be called Continuous Validation, because you are constantly verifying that your app or service will provide (or is providing) the business value expected. Some of the overarching benchmarks include customer adoption, engagement, and revenue. Teams should seek to understand and quantify the sentiment of end users and add them to a continuous feedback loop back to developers and the business.

You must also be able to track how many problems you have – how many defects in pre-production and production, and pair this against the speed at which you can deploy. As people write increasingly better code, there should be a ratio, one that sees the number of defects drop against a constant release cadence or better yet against an increasing release cadence.

Take code coverage and combine that to requirements coverage to better understand how much you're covering from what the business wanted as well as from what was actually built.

Make sure to understand what quality means for your organization and set a baseline. If you don't know where you are starting from, you can't tell if you are improving. Use a KPI that establishes production quality.

Learn what kind of coverage is needed across unit tests and functional tests, and performance and security. It should no longer be thought of a functional and non-functional testing.

Measure the overarching time to production. The point of continuous testing is to reduce timelines to release higher quality releases that provide business value. Focus on increased releases while decreasing rate of production defects and condensing the time to market through automation.

Change people's minds to look beyond pass-fail by developing an aligned definition of done. Defects must be tracked, in order to use the analysis to help with improvements.

You can measure continuous testing success by testing in the lower test environments which can be done quickly because the tests are smaller. Recognize that Testing can keep up with development and you can ship code at any point in the sprint, not just at the end of the sprint.

In assessing and measuring quality versus quantity, factor in the following:

- Time to market
- Value stream analysis
- Timelines and costs of manual test execution
- Regression testing

Start shifting left to developers by democratizing testing tools and evolving testers into engineers.

Every leader could and should have their own view of what constitutes the most valuable continuous testing metrics. Ideal metrics to track include:

- Lead times
- Releases
- Production incidents
- Business value

As you move from waterfall to agile, adoption requires test efficiency, ideally gained from automation and shift left. As you go towards shorter and shorter releases, in your continuous testing journey, it can't just be about getting more automation, more shift left, it's about thinking very differently about how you test.

This calls for a competency around continuous testing that supports CI and CD, enabled by a set of tools that focus on accelerating the feedback to developers across the application pipeline.

You need tools that help with planning, test plans, and test automation, that then get implemented across the CI and CD implementations tool chains.

Security – Do Developers Unknowingly Create Security Vulnerabilities?

Security is a rapidly evolving area of knowledge that must be factored into every flight plan. Procedures must be put in place, reviewed regularly, and practiced diligently for every element of human interaction, including with ground crew, airport personnel, passengers and inflight crew, as well as all the technologies and material goods on board. Security is not a side issue in aviation; it is the axis around which everything else exists.

When strategizing a continuous testing transition, it is vital to factor in security, front and center. Security specialists will quite correctly point out that security has always been treated as something of an "end-of-the-line" concern as well as running in opposition to developers. As much as developers wish to get their product out the door, it is always the security people who say, "not so fast."

Security deserves to be placed in the continuous testing stream at the very start. It is a primary concern when identifying and confirming all aspects of the value chain, from inception to customer. Feedback loops will help to confirm whether the product is making the grade and enable the team to find and fix security-related issues as they arise.

Developers often unknowingly create security vulnerabilities. Continuously testing security is not just about policing those vulnerabilities; it's also about finding out what training they can bring to bear on making the developers better able to understand and become aware of the security of their code and not create those vulnerabilities in the first place. Security must become a partner, a trusted advisor and expert, giving developers training and tooling that can be measured downstream.

Security testing needs to happen at every step, so developers need to run security scans in order to find problems when they happen. Currently security and testing are sitting on the wrong side of the wall from the developers. See this not as a conflict between security and quality but a more of a partnership of trying to figure this out together.

Clarity must be given in the requirements for what constitutes sufficient security. Security is everyone's job but we need to know deterministically when we're done. The development team must be enabled with the training to understand and the tools to measure against the requirements. Testing with these tools should be instrumented into our pipelines so that we assess ourselves continuously. As in any good DevOps process, security scanning is feedback to be used to measure current state and shift training left to upskill the team over time. In addition to more secure outcomes, there is much velocity to be gained by teaching developers to write it correctly thfirst time.

Accountability for security must also rest with a company's higher levels. From there, the questions and solutions cascade all the way down to the lowest levels of the organization all by themselves. If you just start at the lowest levels, you may get traction across a handful of teams, but to get real adoption across the company, you need executive sponsorship. So once you start seeing positive results and success in small pilot projects, elevate those to other teams and leadership. Do roadshows to raise awareness and generate enough interest for leadership to take notice and sponsor enterprise-wide, top-down adoption

Finally, the security team itself must evolve its thinking to fulfil the mission of ensuring its members correctly see themselves as an internal part of IT.

Future Proof – How To Bullet Proof Your Continuous Testing Journey

A flight plan is intended to be a future proof document, in that it outlines the progress of the upcoming voyage before it happens, including potential emergencies and contingencies. Future proofing is about ensuring a craft flies towards its destination with complete success.

Ultimately, A continuous testing plan must include future proofing to exist in an era of unabated change and speed. To future proof, you must have a strategy for all aspects of development. This means from coding to testing, including functional, performance and security. It also includes environment management, from test data generation to service virtualization.

It's imperative to maintain a clear line of sight to ensure tools used for continuous testing are going in that same pattern. The approach must align with where the company is going from a business strategy perspective, and IT needs to be part of that business strategy, not just playing a supporting role.

A company needs dedicated resources to get this done. It should not be given completely over to a third party, but nor should third parties be excluded. It is fine to use Systems Integrators and Service Partners to to accomplish goals, but the company should have its own personnel involved in the process. Build up on your own expertise, but examine and solidify your relationships with third parties, because they might have a solution today – or tomorrow – that can be integrated into the pipeline.

Identify and assemble dedicated teams to build out the platforms that support the application pipeline and the core metrics aligned to continuous testing. This means growing from an organization that has some manual testers and a few engineers, to one that has more capacity to build software.

The continuous testing journey means amount of black box testing will start decreasing while white box testing will become more prominent and relevant, and testers will need to be able to do that. This will demand a change in a COE from being a Center of Excellence to a Center of Enablement, and this will further demand change in the makeup of the teams and the testers.

Big data will be critical for testing, especially getting refinement in code coverage for all types of tests. The market is exploding regarding coverage of the regression suite. Especially in terms of developing a scientific way to know our regression tests achieve the appropriate levels of code coverage. However, just measuring code coverage only ensures that whatever was built, is correct. It doesn't do anything to validating whether what was built was the correct thing. The ultimate goal is to link code coverage to requirements coverage. Then you will ensure you are building the right thing, and building it right.

Companies must be prepared for change and be willing to go with the change. Some of the key milestones here include:

- AI playing a larger role
- Automating the automation, autogenerating testing on the fly will replace regression suites of tests, which will no longer be maintainable due to the exponential growth to achieve optimal requirements coverage.
- Real-time observation of users gaining prominence, and consequently tests will be required to validate the observations.
- Record and replay being a thing of the past
- There will also be a continued improved focus on getting the requirements correct and on new ways of analyzing them and identifying what's missing.

The executive leadership team must recognize how fast the world around them is changing. Work is being done with mobile technologies and cloud-based applications, and speed is its lifeblood. Everything is moving toward a seamless end-to-end customer experience. Cloud, DevOps, Agile, Big Data. The end product for all these things is a seamless customer experience. You must decide how to apply that to testing. This is where the two worlds meet.

The user/consumer must perceive value that the business intended. This is where continuous testing becomes an innovation enabler because it allows for continuous experimentation. If the user doesn't perceive the business value originally intended, continuous testing exposes that gap to the team so that it can be addressed hopefully in the next code deployment."

The Internet of Things means devices are connected in all kinds of ways. Right now, continuous testing is challenging for browsers and APIs, but when you talk about physical and mobile devices and the Internet of Things, integration hubs that allow continuous testing will be the drivers of change. And wherever it is lacking, people will start to notice.

You must always be thinking three steps ahead in terms of things like predictive analytics.

Top notch QA and QE folks are going to evolve quickly. Their skill levels will get higher and the skillset needed for continuous testing will be very different than today. Companies must put mobile and omnichannel first.

The most nimble companies stand to not only win market share through their adoption of a continuous testing culture; they also stand to attract the best people to help them there.

Summary: Planning Every Step of the Way and Learning from Setbacks

Transforming to continuous testing is a journey, just like Agile and DevOps was and still is. Companies and their IT teams will encounter setbacks which must be anticipated and accepted. This is why planning is essential to every step in the continuous testing journey.

Continuous testing is as much a culture as it is a technique. The entire team can learn from setbacks in a "fail-forward" approach – advanced from the walled silos of the past. All great developments and personal victories stand firm on a foundation in which knowledge binds with experience.

Although continuous testing has technology as its tools, it has human beings at its core. Once they become comfortable and confident, they will bring forth the necessary energy, vision, and sheer practical ability.

From these achievements, momentum will can be gathered, and this will provide organizations with sufficient lift to travel successfully across this new global economy.