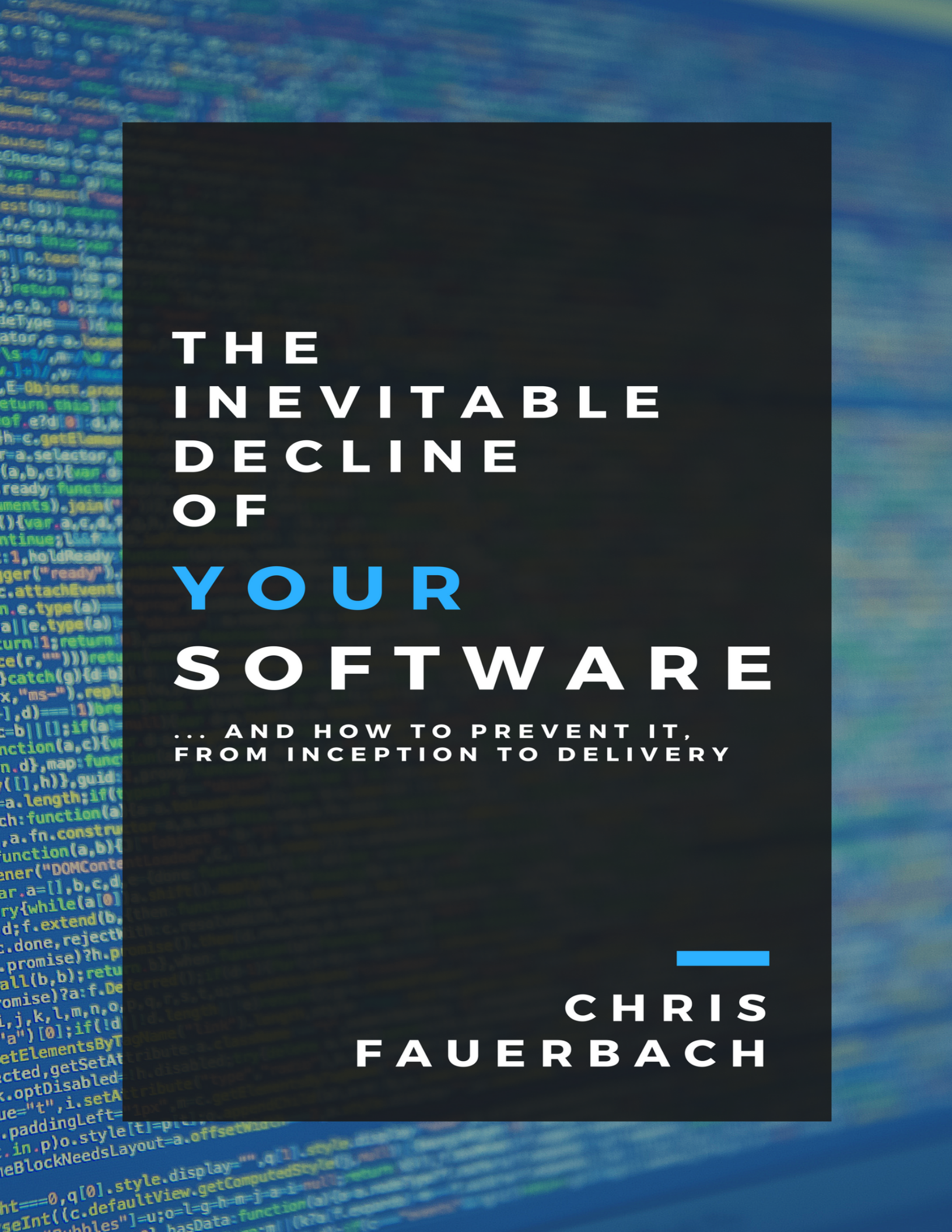
THE
INEVITABLE
DECLINE
OF
YOUR
SOFTWARE
... AND HOW TO PREVENT IT,
FROM INCEPTION TO DELIVERY
CHRIS
FAUERBACH

Inevitable Decline of Your Software

... and how to prevent it, from inception to delivery

Chris Fauerbach

This book is for sale at http://leanpub.com/inevitabledeclineofsoftware

This version was published on 2019-10-27

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

To Leigh Ann, thank you for dealing with my squirrel brain.

Contents

Chapter 0 - Thanks!

Thank you, from the bottom of my heart, for previewing this book. I've been developing software for a long time. Yes, long enough to make me feel old. When I go to work, I'm no longer the youngest, I'm one of the oldest. I'm not quite sure when that happened, but for those of you who haven't transitioned to 'one of the old folks', it'll happen before you know it!

I digress, but that's to say, I've been doing this a long time. As we continue in the relatively new field of software development, we are continually finding new practices and tools to help make our applications better.

Paradigms have shifted. Tooling has changed, yet one thing remains the same. As the age of our software bundles increases, we stop following best practices. We stop writing unit tests. We stop commenting on our code, and we never improve our build pipeline.

One of the most hilarious practices we have is deploying code to a production environment, that can only be built on a specific developer's workstation. Sure, I've done it too, but that was before we had these CICD (continuous integration, continuous delivery) methods. That's never acceptable.

Oh, does your team do it? Did I offend you? Good! Make it better.

Let's work through this together, learning about how to build a new pipeline for delivery.

How to keep code organized, and avoid the spaghetti monster as it ages.

Who's this for?

Software Developers/Engineers/Architects

Quality Assurance, QA, QE

Software Engineering Managers, Directors

Product Managers/Owners

Benefits

The goal of this book is to help your software live a better life. Instead of going to a convalescent home due to your code's inability to speak anything but Java 1.2 RMI, I want to help you create long-living software that will get so old, it can play in the Seniors Tour of the PGA, kicking some serious butt!

Chapter 1 - It’s doomed

Ever read a book with a doomsday prediction? Maybe a post-apocalyptic scenario where survivors try to fend off the zombies. They tend to start one of two ways. Things are good, or things are already bad.

The opening scene is a beautiful landscape, zoomed far out to show you a city, maybe a hillside town. You get comfortable, start daydreaming about what it would be like to live there, and then BAM! You see the mushroom cloud, or you see the unlucky lady running down the street with obvious signs of contamination.

This is where the second type of movie starts. 95% of humanity is dead, and the rest are thieves or murders. Except for our heroine. She’s alive and well, just wanting to get home to see her golden retriever, Biff, one more time. Over the next two hours, you’ll see countless battles or tight scenarios on her epic journey to find out whether Biff is alive or not.

Now let’s relate that to the software.

The New Job

As developers, when we go into a new role, we tend to inherit a codebase from someone who is long gone. As often as I have been in that scenario, I have rarely found a codebase that is still organized well. I don’t expect perfection, but it’s always frustrating to find some code that was either never *designed *or that had obviously *broken *the initial intention of clean code.

Green Pasture

If you got lucky and started a job with perfectly clean code, or you’re starting a new project from scratch, congratulations! You are viewing the landscape and the perfect city in the distance. No nuclear bombs have gone off yet, there are no mushroom clouds on the horizon. Don’t worry, this book is still for you. It will happen. Entropy increases over time. Chaos is always the result of short deadlines and fast bug fixes. I wish it weren’t so, but every piece of software declines and will be ’rewritten’ as time goes on.

Post-Apocalyptic

If you’re here, it may be too late. Ok, no it’s not. It’s always a matter of time, money and people. Here’s the pivotal point though, do you fix it or rebuild it. I’ve always leaned to rebuild it. If the code base is older with dependencies on legacy software, then maybe it is time to rewrite. Either way, the tools and ideas promoted in this book will help take your decaying software, and turn it into manure on its way to being fertile soil.

This Book

This book is meant to scare you into being very intentional when you're writing software. You get paid pretty well to write code for someone else. You're a product developer, and you want to charge more for your product for a long time. Open-source software is typically built by a group, and if you want your pull request to be accepted, you must write "clean code" that matches standards.

Audience

The primary audience of this book is going to be a regular software developer. Whether you're junior or senior in your career, there will be plenty of information here to improve your coding practices.

Another audience for this book is product managers, product owners and managers of software engineers. This book will cover practices that you should require your teams to follow. If you aren't already, and they're clamoring for things like automated builds, code reviews, security scans etc, you really need to give them the time to make sure these items are covered.

Enjoy!

Chapter 2 - Why does it happen?

Are all developers lazy?

Ever hear the famous quote from Bill Gates: ”I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it.”?

That quote has become a central tenet for hiring software developers. The justification is clear, a lazy developer will also find the easy way to do things. They'll automate tasks that they have to do as part of their job. Automating menial tasks is an excellent use of technology.

This viewpoint or opinion of a good developer being lazy has some teeth. Many engineers fall into this mindset. This is appropriate for some scenarios: automate tasks, reduce repetition, simplify workflows, etc. I've always taken issue with the statement because it has become so ingrained in our mantra that people brag about how lazy they are. How backward is that?

Even the good ones are lazy

I'll be the first to admit that there are times I fall into the lazy trap. When we build code the third time, we do it right. Wait, the third time? Building software three times is another mantra of development. The first time we experiment. The second time we organize, the third time we do it right. Throughout this book, I'll be talking about software quality. The base assumption is that we're beyond experimentation, or we're working on a product that has been shipped to customers.

At this stage, our software patterns and code are usually clean. A rating organization would give us an 'A' for following good design patterns, keeping methods short and having nicely organized and documented code. If you're laughing now, then you're with me. This quality of code is rare. Almost a unicorn. My point is that this is typically the only time the code looks as good as it will. It doesn't get better than this. It's like a house. We've cleaned the bathrooms, the windows, the kitchen and even vacuumed out behind the dog crates. You could eat off the floor, as they say.

10 minutes later, it's all a mess again. The dogs have scratched the floors, the kids use the bathroom and the lazy husband didn't clean up after himself in the kitchen. To get it back to pristine, it will take a large effort. The same goes for code. As the quality of code declines, it's really hard to get it back to a nice clean state. You'll frequently rewrite an application under the guise of new patterns or libraries, but it's really because the code hasn't been maintained well.

What happened?

No matter your environment, whether it's a startup building a product or within an enterprise, as we maintain our software, and it lives its life cycle, we start to cut corners. Your customers are using

your product but they find bugs. The application crashes. It doesn't behave how it was originally designed.

You are now entering the maintenance phase of your application. This is the critical time where we all mess up the code. This is the pivotal phase that can determine the longevity of your code. Treat it well, and it can last, but if you start cutting corners, then it will become more brittle, more fragile and will have a much shorter life span.

Unfortunately, this phase is almost always doomed for failure. No one likes maintaining code. We typically like building something new, from the ground up, but unfortunately, that rarely is an opportunity. Since it's rare to build new code, a lot of the jobs out there are to maintain and add features to another developer's code. Even if we did write from scratch perfectly, we'll also get into a maintenance phase if users actually rely on your application.

When we're in maintenance, we cut corners. We break our design patterns 'just this once'. We'll add global variables since it's the easiest way to pass data (seriously, do you do this?! It's a cardinal sin!). The issue I tend to fall into has to do with documentation and unit testing. I'll add comments when I first write code but will admit that keeping comments and unit tests up to date fall by the wayside.

Doom and Gloom

"Jeez, Chris, this sounds bad. Why even bother trying to fix it."

Which do you think is easier, fixing something that's horribly broken, or keeping something unbroken? I'll relate it back to the house. If you've ever looked into a housekeeper/cleaner, you'll see their price schedule. The first cleaning is $150, but every cleaning after that is $100. It costs more to fix than to maintain.

If you're a 'lazy', it goes the same way. As you patch your software, you'll cut corners. The nice abstraction layers you've built now seem to be in the way of getting code releases completed. The 'textbook worthy' implementation of the "Gang of Four" design patterns start to fall apart. If you're under pressure, we bow down to timelines and just 'make it happen' the quickest way possible. When there are production issues we need to patch ASAP. When a single customer provides 80% of your company revenue, and they're unhappy, you make fixes as quickly as you can.

imagine this style of maintenance for one or two years. In the end, the code is spaghetti. Dependencies are hard to follow. It's near impossible to find code that results in bugs. The original developer has left the team. The first primary maintainer has also left. You're coming on board to maintain this mess. You want to take this book, back in time, and smack some people silly.

Let's try to cut this off before it happens again.

Chapter 3 - Process Overview

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Lifecycle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* High-Level Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* Interface Definition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Development

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* Source Control

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* Code Quality

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* Code Scanning

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* Development Process

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* Automated Testing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

* Continual Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Maintenance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 4 - Lifecycle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Screw it, Ship it!

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Waterfall

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Agile

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Individuals and Interactions over processes and tools

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Working Software over comprehensive documentation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

*

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 5 - Technical Design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Declared Goals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Defined Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Common Design Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 6 - Cookbook - Web App at AWS

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Project - elecTrick

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Ground Rules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 7 - Interface Definition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Changing Interfaces

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Breaking Changes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Patterns for Interactions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Retrieving Specific Data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 8 - Source Control

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

The Bad that gets Ugly

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

The Good

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Source Control

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Branching

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

What Did We Learn?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 9 - Code Quality

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Code Standards

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Don’t Reinvent the Wheel

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Python Guide Explained

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 10 - Code Scanning

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 11 - Development Process

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 12 - Automated Testing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.

Chapter 13 - Continual Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/inevitabledeclineofsoftware.