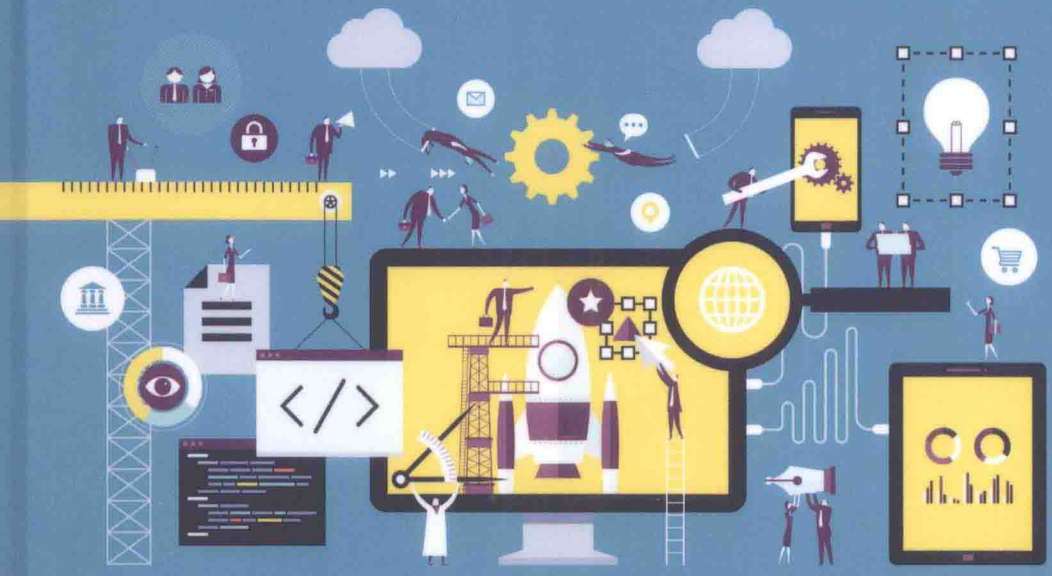

ROCKY HECKMAN

DESIGNING PLATFORM INDEPENDENT MOBILE APPS AND SERVICES

IEEE PRESS

IEEE computer society

WILEY

DESIGNING PLATFORM INDEPENDENT MOBILE APPS AND SERVICES

ROCKY HECKMAN

IEEE computer society

IEEE PRESS

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Names: Heckman, Rocky, author.
Title: Designing platform independent mobile apps and services / Rocky Heckman.
Description: Hoboken, New Jersey : John Wiley & Sons, Inc., [2016] | Includes index.
Identifiers: LCCN 2016009419| ISBN 9781119060147 (cloth) | ISBN 9781119060185 (epub) | ISBN 9781119060154 (Adobe PDF)
Subjects: LCSH: Mobile computing. | Cell phones–Programming. | Mobile apps.
Classification: LCC QA76.59 .H43 2016 | DDC 005.25–dc23 LC record available at https://lccn.loc.gov/2016009419

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Thank you to all my friends who finally convinced me to write a book. Most of all, thank you to my wife Stefanie, and my two beautiful girls Elyssia and Keiralli for not only pushing me to finish, but putting up with all the time I spent doing it. I love you all very much.
Look Dad, I did it!

LIST OF FIGURES

LIST OF TABLES

PREFACE

I WAS WONDERING how to start this preface, and it occurred to me that writing one is backwards. After all, this is the text before the book, but you write it after the book is finished. I suppose that is analogous to how we have written software for a long time. We figure out what it is we are trying to do usually though creating systems that do what we think we want them to do; then we go back and write the documentation about what the system actually does.

This book, in large part, is aimed at helping to make that a more harmonious effort. Technology is moving so fast now that often we find ourselves trying to create mobile apps and services in a very reactionary manner. We tend to be on the back foot and playing catch up most of the time. If we could just take the time to sharpen the proverbial axe, we'd be able to get more accomplished, faster, with a lot less hair pulling.

I suppose over the past couple decades of doing this, I've seen that pattern time and time again. But there are also some good habits and patterns that I've seen along the way that in some respect were ahead of their time. Service-oriented architecture, for example, was a great idea for connecting the myriad of systems we've had inside organizations with simple, easy-to-use interfaces. In fact, this tried-and-true pattern, or collection of patterns if you will, is more relevant today in our commodity cloud computing, mobile app world than ever before.

This book is designed to provide some high level architectural guidance on how to design modern mobile apps and services so that they can adapt as technology inevitably changes. Of course, we start off with a brief history of our mobile computing explosion, and take a look at attempts to create cross platform apps and technology stacks.

Then I want to introduce what hopefully has become an obvious application stack. While we have been fairly fixated on a *N*-tier stack, where *N* usually equaled three, to truly futureproof our architectures, we really need two more clearly defined layers to provide us an abstraction boundary which insulates our code from changes in external client technology, as well as the rapidly changing data storage technologies we use today.

Once we have our layers sorted out, we'll have a look at various patterns of application development and how they apply to this layering system to create performant and resilient services for making powerful mobile applications.

I hope that you find this guidance useful. Perhaps it will make you think of things in ways you hadn't before, or validate thinking you've already implemented. In any case, I hope it prevents you from having to operate in a reactionary manner to the rapid changes of our modern computing world and lets you get on the front foot

so you can focus on creating great apps and services instead of retooling everything because a new phone hit the market.

Target Audience

This book is for anyone who is responsible for the design, architecture, and development of modern mobile apps and the services that support them. I've written this book with futureproofing in mind. Ideally, the architectures and patterns in this book will provide you with an approach that will futureproof your designs.

By following this guidance, you should be able to create mobile app services that you can adapt, modify, update, change, or integrate without disrupting your mobile apps, or your teams. You should be able to deploy new services, change existing services, and add new client apps all without disturbing any of the running systems.

Most of all, you should be able to adapt services and apps based on this guidance to any new mobile platform that comes along. This will greatly increase your code reuse, make your teams much more efficient, and make your organization adaptable to the ever-changing mobile app landscape.

ACKNOWLEDGMENTS

These kinds of things don't happen without a lot of people in the background pushing, pulling, helping, and sometimes simply nodding and smiling. I would like to thank Chris Bright for encouraging me and allowing me the time to put this together. I'd also like to thank Andrew Coates and Dave Glover for letting me harass them with ideas, and "what if" questions all the time.

Most important of all, I need to thank my wife Stefanie Heckman, and my two girls Elyssia and Keiralli. They not only encouraged me to finish, but were patient with me, and gave me the time to keep typing away. I think, in the end, their love and enthusiasm are what really got this book over the line. So if you like it, don't forget to thank them too.

TABLE OF CONTENTS

CHAPTER 1

THE MOBILE LANDSCAPE

1.1 INTRODUCTION

When the idea of reaching people first struck home in the dark ages, we wanted to find ways for people to use and pay for our services. We had to find a way to let people know these services existed. In early days there were town criers, then during the industrial revolution when we could reproduce and distribute text to a largely literate audience, we had broadsheets. Then came the catalogue where mercantile companies would list their wares for sale. Once we heard the first radio waves, one of the first things they did was to sell advertising on the radio. This graduated to television advertising. Then along came the Internet. Everyone had to get themselves a website and would put their website address in their print, radio, and TV ads. Along came Facebook and everyone created a Facebook page for their companies.

Now, everyone wants to have an app for their customers to download. These apps go with customers wherever they are and provide instant interaction between consumer and supplier. We can push advertising into them, take orders through them, keep in touch with friends and relatives, and of course play games, listen to music and watch videos all in the palms of our hands. These experiences require devices, operating systems, and apps, all of which require software companies, architects, and developers to produce them. Unfortunately, these devices and operating systems often change.

In today's computing world, there is one thing you can be sure of; the leading operating system (OS) platform will change. As recently as 7 years ago, Microsoft Windows Mobile was the leading smartphone platform and tablet computers, while mobile, were large and clunky and ran full versions of the Windows XP and Windows 7 OS. Then came Blackberry which took a lot of market share from Windows Mobile. But that only lasted until the iPhone came along in 2007 and we went from a feature phone dominated world to a smartphone dominated one. This set a new benchmark and became the leading mobile computing platform. In the same year, the Open Handset Alliance re-released Linux-based Android-powered smartphones. Then in 2010, Google launched its Android-based Nexus series of devices. By 2011, Android-powered smartphones made up the majority of mobile OS-powered smartphones shooting past the iPhone.

While phones were taking off, in 2010 Apple released the iPad. Tablet computing was not new and in fact Microsoft and its Original Equipment Manufacturers

Designing Platform Independent Mobile Apps and Services, First Edition. Rocky Heckman.

OEM partners had been trying to sell tablet computers since 2003. However, the iPad's sleek design brought tablet computing to the masses despite the clumsy and restrictive iOS operating system. This opened up the tablet computing market which Android was well suited for. After the iPad's initial success, by 2013 Google's Android-powered tablets had overtaken iPads as the tablet of choice. Additionally, although lagging considerably behind, Microsoft has re-created itself to make a run in the mobile and tablet computer markets as well. With Microsoft's massive install base, and very large developer ecosystem, they are likely to challenge Apple and Google in the mobile and tablet space eventually. With Windows 10 released as a free upgrade for over 1.5 billion eligible devices [1], it is likely to be the most common cross platform OS. That is, over half of Gartner's predicted 2,419,864,000 devices shipped into the market in 2014 [2]. Overnight the app ecosystem market leader could change again.

What this means for software developers, independent software vendors (ISVs), hobbyist app developers, and online service providers is that every few years they will have to retarget their efforts for a new platform, new development languages, new development tools, new skills, and new ways of thinking. This is not an attractive proposition for anyone. However, due to the success of the iPhone and iPad, software developers were willing to re-skill and even purchase proprietary hardware just to be able to develop applications for the new platforms. Then when Android devices surpassed the Apple devices, these same developers painfully went through the whole process again. Developers were forced to maintain three or more separate and complete codebases. This is the problem that Platform-Independent Delivery of Mobile Apps and Services solves.

If you are not planning a platform-independent strategy, you will likely be an ex-company in 3–5 years. Due to the rapid change of the consumer and enterprise mobile computing landscape, software developers must be able to adapt to new platforms, devices, and services before their competition. While cross-platform goes a long way toward this goal, it is still cumbersome and tends to lag behind a more platform-independent strategy. While it is not practical to get completely away from device-specific app code, the more you can move off of the device and put into a reusable back-end service, the less code you have to write and maintain when a new OS version or a new platform comes along. In this book we will examine strategies to do this, and provide future proof foundations to support changes in the computing landscape down the road.

Disclaimer: This book was written in late 2014. Everything in it was accurate at that time. If you are reading this in 2025, expect that a few things have changed. Just keep this in mind as we go through this so I don't have to keep writing "At the time of this writing...."

1.2 PREVIOUS ATTEMPTS AT CROSS-PLATFORM

1.2.1 Java

"Write once, run everywhere" was a slogan developed by Sun Microsystems which promised cross-platform capability for Java applications supported by Duke, Java's Mascot shown in Figure 1.1. This gained significant traction in the mid to late 1990s.

Figure 1.1 Duke

In the beginning of this era, the promise seemed legitimate. You could write the Java code once, package up your Java byte code in .jar files and run them on any system that had the Java Virtual Machine (JVM) interpreter. It worked so well that there are even C to Java compilers so your C applications can run with the same cross-platform reach that Java had.

The promise was that you could write code like this:

```
class CrossPlatformApp {
    public static void main(String[] args) {
        System.out.println("I am omnipresent!");
          // Display the string.
    }
}
```

And it would run on every computer and device that ran Java without compiling multiple versions for each target device. All you had to do was make sure that the target device had the correct version of the JVM installed on it.

This worked fine until various vendors started creating their own versions of the JVM to run on their platforms. By 2014, more than 70 different JVMs [3] had been created that could run Java applications, for the most part. The catch was that they were each slightly different.

If we take the Sun JVM to be the standard, some of these other JVMs were better, and most were worse, at interpreting Java byte code. Some of them such as the IBM J9 (http://en.wikipedia.org/wiki/IBM_J9), the Azul Zing JVM (http://en.wikipedia.org/wiki/Azul_Systems), and the Microsoft JVM (http://en.wikipedia.org/wiki/Microsoft_Java_Virtual_Machine) were better