# Beyond Requirements

## Analysis with an Agile Mindset

Kent J. McDonald

# Beyond Requirements

# The Agile Software Development Series

Alistair Cockburn and Jim Highsmith, Series Editors


Visit **informit.com/agileseries** for a complete list of available publications.

---

Agile software development centers on four values, which are identified in the Agile Alliance's Manifesto*:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

The development of Agile software requires innovation and responsiveness, based on generating and sharing knowledge within a development team and with the customer. Agile software developers draw on the strengths of customers, users, and developers to find just enough process to balance quality and agility.

The books in The Agile Software Development Series focus on sharing the experiences of such Agile developers. Individual books address individual techniques (such as Use Cases), group techniques (such as collaborative decision making), and proven solutions to different problems from a variety of organizational cultures. The result is a core of Agile best practices that will enrich your experiences and improve your work.

# Beyond Requirements

## *Analysis with an Agile Mindset*

**Kent J. McDonald**

**Illustrations by Jeff Rains**

Addison-Wesley

New York • Boston • Indianapolis • San Francisco
Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

*Library of Congress Cataloging-in-Publication Data*

McDonald, Kent J.
Beyond requirements : analysis with an agile mindset / Kent J. McDonald ; illustrations by Jeff Rains.
pages cm
Includes bibliographical references and index.
ISBN 978-0-321-83455-3 (pbk. : alk. paper)—ISBN 0-321-83455-0
1. Decision making. 2. Requirements engineering. 3. Business requirements analysis. I. Title.
T57.95.M384 2016
658.4'0354—dc23

2015022866



Illustrations by Jeff Rains



ISBN-13: 978-0-321-83455-3
ISBN-10: 0-321-83455-0
Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, September 2015

*To all of those who asked, "Is the book done yet?"*
*Yes, yes it is.*

*This page intentionally left blank*

# Contents

# Preface

## What This Book Is About

I wrote *Beyond Requirements* to paint a picture of analysis in IT projects and how it can be applied with an agile mindset to make those projects more effective. For the purposes of this book I think of analysis as the activities involved with

- Understanding **stakeholders**
- Understanding **context**
- Understanding the **need**
- Understanding the **solution**(s)
- Organizing and persisting solution information

Performing these activities with an agile mindset, which I explain in Chapter 1, best positions teams to satisfy stakeholder needs. As a result, I assume that people are approaching work with an agile mindset (which is up to each individual to adopt) and that they are using agile techniques. Most of the techniques I describe can also be used in other environments, of course, but they're most effective when combined with agile approaches.

## Who Is This Book For?

If you find yourself performing analysis on a project in order to make sure the project is delivering the right thing, this book is for you. You may identify yourself as a **business analyst** (or derivation of that title), product owner, product manager, project manager, tester, or developer.

I chose to target those performing analysis activities or possessing analysis skills rather than analysts as a role, or even analysts as a profession. While it is true that the people who are most endowed with the analysis skill set are those who generally fill an analyst role, I didn't want the advice in this book to get hung up on discussions such as, "The analyst does this, the developer does that, the tester does this other thing." I'd much rather focus on describing why and when techniques are most appropriate and leave it up to you and your team to

determine who is the best person to do various activities. In many cases, multiple people on your team will end up doing analysis in order to take advantage of strong technical and business knowledge.

The business analyst role exists primarily because in the past several organizations used a prescriptive, phase-based approach to software development. In this approach, there was a time period in the project when the main work was eliciting and documenting **requirements**. Since it made sense to structure the software development organization according to how project work was done, all the people doing work in the analysis phase were lumped together and called business analysts. But gathering and documenting requirements didn't generate much respect for the people doing it. Members of the analysis community longingly eyed the success project managers had enjoyed by proclaiming project management a profession, and they chose to do the same.

A lot of good things have come from the "professionalization" of **business analysis**, including more consideration of, training on, and attention to analysis skills. However, the benefits are somewhat diluted by the effort required to justify a separate profession for people who elicit, document, and manage requirements, and the overspecialization that may result. That effort would be better spent figuring out how analysis can be used to make projects more successful.

That doesn't change the fact that you have a business analyst title and you have spent a considerable amount of your career honing your business analysis skills. Where does that leave you? Looking at analysis as an activity more than a role, title, or profession means that you can use your in-depth knowledge of analysis techniques to help your teams solve the right problems in the right way and help out with other activities on the project whenever possible.

## To What Context Does This Book Apply?

This book focuses on the analysis that occurs on **IT projects**. An IT project is any project that results in solutions, often involving software, that support internal business processes, automate manual processes, or streamline current processes. Examples include building a system to support the session submission process for a conference, implementing a system to calculate and deliver commissions, reporting and data warehousing solutions, or implementing a solution to track student information at a nonprofit school.

I chose this focus for a few reasons. First, activities labeled as business analysis and the role of business analyst seem to be more prevalent in IT projects than in activities focused on **product** development. Second, most of the existing literature in the analysis space seems to assume a product development context,

and the context of the IT departments of an organization strikes me as underserved. Third, and probably most important, it's where most of my experience lies, so focusing on that topic gives me the opportunity to write from actual experience.

As I describe how analysis with an agile mindset works on IT projects, I won't delve too much into how to do tried-and-true analysis techniques. There are already enough resources that do a fantastic job of explaining those techniques, and it dilutes the focus of this book. Instead, I'll focus on why those techniques are helpful and when they are best used. I do introduce a few techniques from other skill sets not commonly known in analysis circles, and in those cases I provide a more detailed description of how to perform that technique. In all cases, I provide my favorite references for more information about those techniques.

The word project has acquired a certain stigma in the agile community. Those who apply that stigma feel as though the use of the word project implies some of the downsides of the way that projects are managed in a waterfall setting.

The term project often suggests the following:

- The temporary nature of projects is applied to the teams that work on them. People are brought to the work instead of the work being brought to the team.
- It takes a while to get an effort going due to the extensive chartering and planning that come with trying to predict the future 6 to 12 months out.
- Even though projects are intended to be temporary (or maybe because of that), they are rarely stopped once they get started. Sponsors and teams get enamored with projects and become more reluctant to end a project the longer it goes on.
- The project funding approach may encourage grouping multiple small **changes** together in order to justify expenditure, increasing the time before the changes are delivered to waiting stakeholders.

While these problems certainly exist, merely using the word project does not ensure that they will happen. I reasoned that most people are familiar with the idea of the project, and it would be more useful to explain that these patterns are antipatterns and it's possible for projects to work differently than to use a new term for an existing concept and deal with all of the confusion that could cause. As Deanna, one of my editors, suggested, I should just "own it" when it comes to using the word project.

## What Problem Is This Book Trying to Solve?

Analysis is often portrayed as eliciting and documenting requirements, frequently in terms that sound a lot like asking people what they want and writing it down. Deep philosophical discussions about analysis often center on the best way to capture requirements: "Should I use a use case, or should I use a **user story**?" Requirements are important, but they are a means to an end, not the end in and of themselves. As I described previously, analysis is about understanding your stakeholders and their needs, identifying the best solution for satisfying those needs in your particular context, and then building a shared understanding of that solution. Requirements play a part in that work, especially around describing the need, but they are certainly not the end product.

One fundamental problem this book is trying to solve is how to determine whether your IT project is doing the right thing and how analysis can help you do that. It's about changing the purpose of analysis from requirements gathering and capture to problem solving and building shared understanding. Along with that comes a substantial change in how your team views requirements and **designs**. They are no longer deliverables that get tossed over the wall to the people performing the next step in the process. Now both requirements and designs are tools that teams can use to build a shared understanding of the solution they seek to deliver in order to reach a desired **outcome**.

A second fundamental problem this book attempts to solve is to demonstrate how to do analysis in an agile setting. As many teams first adopt agile approaches, they struggle with finding the right balance between identifying a viable solution and describing that solution in too much detail too early. This book aims to show you how to perform analysis in an iterative fashion so that you can take advantage of the learning that occurs during development, testing, and deployment. While doing so, it also demonstrates that many analysis techniques are applicable in an agile setting with changes to when and to what extent you perform those techniques. I sought to solve this problem because many teams that adopt agile think analysis is no longer necessary, and as a result they end up creating solutions that don't solve the identified problem, or don't solve any problem at all.

## How the Book Is Organized

This book is organized into three main parts to make it a bit easier to consume. The first part, "Ideas," covers the agile mindset and some key principles that underlie the agile mindset and effective analysis. The second part, "Case Studies," features four case studies that show how to practically apply the ideas in a variety of situations. The third part, "Techniques," takes a deeper view of some techniques that are very helpful for using analysis in an agile setting.

## Part I: Ideas

The first section takes a look at some key ideas that I consider essential for effectively performing analysis in an agile setting. These include the concepts that describe an agile mindset, and some helpful concepts from outside traditional analysis thinking that supplement typical analysis techniques. Finally, I build on those ideas to place analysis techniques in context.

### *Chapter 1: Guiding Principles*

As I help teams adopt agile and tighten up their analysis approach, I find that adopting the appropriate mindset is more important than mastering a specific set of techniques. With the proper mindset and a great deal of self-discipline a team can be successful with minimal process. Without the proper mindset, teams find that they must continuously add process to aid the **collaboration** that comes naturally to those who have the right mindset.

What is the proper mindset? There are a variety of perspectives on that. The original definition of the agile mindset is encapsulated by the "Manifesto for Agile Software Development" and the corresponding principles. Others have expanded on those original ideas to describe the agile mindset, and I have done the same, placing emphasis on aspects that encourage building the right thing. I describe my perspective on the agile mindset through seven guiding principles:

- Deliver **value**
- Collaborate
- Iterate
- Simplify
- Consider context
- Decide wisely
- Reflect and adapt

### *Chapter 2: Helpful Concepts*

I use this chapter to introduce some ideas that form the conceptual basis for the following chapters. The ideas discussed include

- Needs and solutions
- Outcome and **output**
- **Discovery** and **delivery**

### *Chapter 3: Influence of Lean Startup*

This chapter explores some concepts of Lean Startup and describes how these concepts can be applied effectively to the context of IT projects. Those concepts include

- **Customer** development
- **Build-Measure-Learn**
- **Metrics**

### *Chapter 4: Decision Making*

This chapter discusses decision making in more detail, specifically a structure for decision making, the idea of **Real Options**, and the cognitive biases I find can get in the way of effective decision making.

### *Chapter 5: Deliver Value*

In this chapter I discuss some key concepts surrounding value delivery, including **Feature Injection**, **minimum viable product**, and **minimum marketable feature**.

### *Chapter 6: Analysis with an Agile Mindset*

While I'm not necessarily advocating a new "analysis process," I wanted to provide a general description of how analysis flows alongside the lifecycle of a project. This chapter positions the techniques from Chapters 11 through 15 in their usual location in the project lifecycle.

I don't spend a great deal of time talking about this flow specifically because it is not the same on every project, but going through the whole flow once helps put the techniques into the proper perspective and helps to explain why certain techniques make more sense in some contexts than in others.

## Part II: Case Studies

In this part of the book, I share four stories intended to describe analysis in a real-world setting. These stories illustrate the ways a variety of IT projects used the ideas described in Chapters 1 through 6 and the techniques described in later chapters. While I cannot cover every possible situation, I hope this mix of case studies provides fairly broad coverage of the various environments in which you may find yourself. In addition, they should furnish ideas for using the same techniques in different situations and adjusting your approach based on your current context.

### *Chapter 7: Case Study: Conference Submission System*

This is the story of developing and maintaining the submission system for the Agile2013 and Agile2014 conferences. This was a fairly straightforward project, but it provides the opportunity to position several analysis techniques in their proper context.

### *Chapter 8: Case Study: Commission System*

This case describes what happened when a health insurance company undertook a project to replace multiple commission systems. The case explores some good techniques for projects involving off-the-shelf software and the tendency to gold plate.

### *Chapter 9: Case Study: Data Warehouse*

This case tells the story of a project to incorporate a new source of data into an existing data warehouse. This story explores analysis in a business intelligence project, another environment that can benefit from an agile mindset.

### *Chapter 10: Case Study: Student Information System*

This case explores analysis in a nonprofit setting and focuses on the decisions that need to be made when a project is initially being considered.

## Part III: Techniques

In this section I describe a series of techniques that can be helpful in many different settings using my technique brief format. That format covers the following aspects of a technique:

- What it is
- An example
- When to use it
- Why use it
- How to use it
- Caveats and considerations
- Additional resources

### *Chapter 11: Understanding Stakeholders*

This chapter describes some techniques that are helpful for understanding the people you work with. The first two techniques are useful for understanding

the people whose needs you are trying to satisfy—better known as **stakeholder analysis**. The other two techniques in this chapter will help you better understand the people who are actually going to use the solution you deliver; let's call this **user analysis**. The techniques I cover include

- **Stakeholder map**
- **Commitment scale**
- **User modeling**
- **Persona**

### *Chapter 12: Understanding Context*

Understanding context means familiarizing yourself with the nature of the business and sharing that information with the rest of the team. You want to put the project in the perspective of the overall organization and determine what the project is intended to do. If the project does not support something explicitly related to the organization's **strategy** or ongoing operations, don't do it.

This chapter introduces several techniques for understanding the organization as a whole and using that information to guide decisions about your projects. The techniques described in this chapter are often called **strategy analysis** (formerly enterprise analysis) in the analyst community.

- **The Purpose-Based Alignment Model**
- **Six questions**
- **The Context Leadership Model**

### *Chapter 13: Understanding the Need*

A key and often overlooked aspect of IT projects is figuring out the real need that must be satisfied, determining if it is worth satisfying, and sharing that understanding with the entire team. If those activities were done more frequently, the story told about IT projects would undoubtedly be much brighter.

In this chapter, I introduce a set of techniques that I have found very helpful for performing those activities:

- **Decision filters**
- **Project opportunity assessment**
- **Problem statement**

### *Chapter 14: Understanding the Solution(s)*

Once we understand the need we're trying to satisfy and we've determined that it's worth satisfying, we should investigate possible solutions. The plural form is intentional. Project teams often limit themselves by focusing on one possible solution too soon instead of leaving their options open. In many cases there are multiple options.

In this chapter I identify a variety of techniques for exploring multiple solutions and describing the solutions that seem best, all in a way that is meaningful for everyone working on the project:

- **Impact mapping**
- **Story mapping**
- **Collaborative modeling**
- **Acceptance criteria**
- **Examples**

### *Chapter 15: Organizing and Persisting Solution Information*

This chapter describes techniques that help teams visualize progress and the aspect of the solution they are working on, as well as a way to persist key information about the solution for future reference. The techniques described in this chapter include

- **Discovery board**
- **Definition of ready**
- **Delivery board**
- **Definition of done**
- **System documentation**

## Part IV: Resources

In this final part of the book, I provide a couple of resource sections that summarize key definitions and reference sources collected from the rest of the book.

### *Glossary*

It's always a good practice to establish a common language for your projects. Since I am trying to be very specific about how I refer to certain concepts, and

in the interest of eating my own dog food, I decided to establish a glossary for *Beyond Requirements*. This should help me be consistent in my use of certain words, or at least give you a chance to catch me if I am inconsistent. Words in the glossary appear in bold the first time they are mentioned in the text.

### *References*

Throughout the book I reference several great sources of additional information about the topics I discuss. This section compiles all the references into a single list. Take some time to check out the references listed here; there's some great stuff.

In addition to the resources included in the book, beyondrequirements.com features additional thoughts on analysis with an agile mindset, new technique briefs, and updates to the material in the book.

# Acknowledgments

This is not the first book I have written, but it is the first I took on by myself, or at least that's what I thought the case was when I started. It turns out that while I'll be listed as the only author, this book would not have been possible without the help of several people.

There are two people who played the biggest part in how the book looks and reads. Jeff Rains created all the hand-drawn graphics in the book. It was important that the graphics reinforce the idea of having a conversation at a whiteboard. Jeff's great work allowed me to get that message across while allowing you to be able to read the graphics. Deanna Burghart provided the first line of defense that prevented me from doing horrendous things to the English language. I have worked with Deanna for several years as she edited my pieces for ProjectConnections.com. I knew when I started working on this book a . . . um . . . couple of years ago that I wanted her editorial help. She, as always, did a great job helping me sound like me.

I have been fortunate in my professional life to work and interact with brilliant people who look at things in a slightly different way and who do not hesitate to share their perspective with me. Several of those people played a part in this book, but it's important that I thank three especially. It is truly an honor and a privilege to be able to fall back on these three to discuss ideas and ways to describe them. Gojko Adzic's extensive review notes were an immense help during the editing stage and helped me see things from a different and better perspective. Todd Little reviewed most of the book during the final editing stages and, as always, provided practical and insightful advice to help me crystallize my revisions. Chris Matts, long a primary source of cutting-edge, yet eminently practical thought in the space of analysis, generously discussed several ideas for this book and was a key source of many of the more important ones. My understanding of the nuances of analysis and IT project work is due largely to being fortunate enough to know these three practitioners.

I was fortunate to receive feedback from a wide range of professionals. Special thanks go to Robert Bogetti, Sarah Edrie, James Kovacs, Chris Sterling, and Heather Hassebroek for reading and commenting on the entire draft. Their comments were very helpful in shaping and refining my initial thoughts into something that I hope is a bit more coherent. Thanks also to Diane Zajac-Woodie, Deb McCormick, Brandon Carlson, Mary Gorman, Julie Urban,

Pollyanna Pixton, Matt Heusser, Tina Joseph, and Ellen Gottesdiener, who all gave feedback on aspects of the draft.

Finally, thanks to Chris Guzikowski, acquisitions editor at Addison-Wesley, who had the patience to stick with me through the drawn-out writing process, and Jeffrey Davidson, who didn't let an opportunity go by without nagging me about finishing the book. Jeffrey, I'm not sure if Chris put you up to that or not, but I suspect he's glad you did, regardless.

# About the Author

**Kent J. McDonald** uncovers better ways of delivering value by doing it and helping others do it. His years of experience include work in business analysis, strategic planning, project management, and product development in a variety of industries, including financial services, health insurance, performance marketing, human services, nonprofit, and automotive. He is active in the business analysis and agile software development communities helping people share stories about what does and does not work.

Kent has a Bachelor of Science degree in industrial engineering from Iowa State University and an MBA from Kent State University.

Kent is also a coauthor of *Stand Back and Deliver: Accelerating Business Agility*.

*This page intentionally left blank*

# Chapter 8

# Case Study: Commission System

## Introduction

McMillan Insurance is a midsize health insurance company located in a midsize city in the middle of the United States. McMillan has grown through acquisition, and until recently one of its practices was to let each company keep its own identity when dealing with anyone outside the walls of headquarters. This included the relationships with independent agents and the resulting commission structures. This meant that Arthur, the manager of the commissions area, had to deal with a slew of different very unique commission rules down to the individual agent level, and the resulting hodgepodge of commission "systems" required to administer those different commission plans. McMillan has finished its acquisition binge and now realizes that some commonality needs to be introduced in many areas, including commissions.

Arthur was charged with making the commissions area more efficient, so his first instinct was to find a new commission system that would allow him to administer all the various commission plans in one place, while still maintaining all the unique commission structures. He sat down with a couple of more experienced members of his staff, and they started scouring the Internet for possible products. A quick search revealed several options. (Of course, this should have been obvious just from the seven different software applications McMillan had inherited from the acquired companies, only one of which was built in-house.)

It was at this point that they reached out to IT for some help figuring out what to do. Arthur was a little hesitant to do that at first because he was concerned that IT would want to build something in-house. He was pleasantly surprised when Heather, a business analyst from IT on the team, suggested that instead of immediately going out and looking for specific products they should step back

and think about what need they were trying to satisfy. Heather and Arthur sat down to discuss the current situation and what Arthur hoped to accomplish.

## The Need

As a result of their conversation, Arthur and Heather identified the following objectives:

- Reduce the time it takes to produce commission payments from one week to two days.
- Reduce the time required to set up a new commission plan from six weeks to one week (needed every time a new product is created).
- Reduce the time required to set up a new agent from one day to one hour.

They then discussed the characteristics of a desirable solution. As they were talking, Heather used the Purpose-Based Alignment Model (Chapter 12) to identify commissions as a **parity activity**, and Arthur realized that trying to have unique commission rules for every agent was, in effect, overinvesting in commissions. Data from the existing commission payments indicated that the unique rules did not have a direct impact on what the agents sold, so they were probably not worth the effort that Arthur's area spent in creating and administering them. Arthur made a note to talk to the sales managers about reducing the complexity of the commission rules.

At this point a team was formed that included Arthur and some of the more experienced members of his staff as well as Heather and a few others from IT. Arthur and Heather described the objectives they had put together and then worked with the team to create decision filters for the project, to make sure everyone was on the same page.

Here are the decision filters they came up with:

- Will this reduce the cycle time for commission payments?
- Will this help us set up a commission plan faster?
- Will this help us set up a new agent faster?

## The Possible Solution(s)

Once the team had a good understanding of what they were trying to accomplish, they decided they needed to identify options for realizing those objectives, starting with reducing the time required for commission payments. They used

**Table 8.1** *Desired Characteristics of New Commission Software*

| Characteristic | Required/ Optional |
|---|---|
| Accept inputs from multiple policy systems to determine commissions. | Required |
| Create unique commission rules for each individual agent. | Required |
| Support multiple hierarchies: some sales channels are organized based on product, others are based on geography, some are based on both product and geography. | Required |
| Allow for adjustments to occur in the calculated commission rules. | Required |
| Allow for manual determination of commission payments. | Required |
| Create unique commission rules based on free-form attributes and specific values of those attributes. | Optional |
| Support multiple commission rules unique to the individual, unique to the policy. | Optional |

impact mapping (Chapter 14) to help them identify options. Several options came up, including simplifying the commission rules and consolidating the multiple commission systems into one. The team also identified multiple options for dealing with the existing systems:

- Build something in-house.
- Revise the existing conglomerate.
- Purchase something.
- Outsource all commissions activity.
- Do nothing.

The team decided that the best route was to start with simplifying the rules for commissions in one of the acquired companies to see if there was any impact on sales. At the same time, they started the search for software to replace all of the existing commission systems. Table 8.1 lists the characteristics that served as criteria for the search.

The team included the optional characteristics as a way of seeing if any commonly used applications used complex rule logic, in case they found data to support the need for unique commission rules.

## The Deliveries of Value

The team split the work into a series of rounds. (They chose that term instead of *releases*, because not every round involved deploying software.) They weren't

**Table 8.2** *Rounds of Work*

| Round | Contents |
|---|---|
| 1 | • Simplify the commission rules for Southern Comfort Insurance (SCI).<br>• Identify a commission system to purchase. |
| 2 | • Implement a commission system in-house.<br>• Use the commission system for McMillan agents (who already had straightforward commission rules).<br>• Simplify the commission rules for Western Amalgamated Insurance (WAI). |
| 3 | • Use the new commission system for SCI.<br>• Phase out the existing commission system for SCI.<br>• Simplify the commission rules for Eastern Agrarian Insurance (EAI). |
| 4–*N* | • Roll out the commission system to the remaining units.<br>• Simplify the commission rules for the remaining units.<br>• Phase out the existing commission systems. |

sure how many rounds they would have at the beginning, but they knew they would be organized along the lines shown in Table 8.2.

The team figured that after the first couple of rounds they would simplify rules and move the units to the new commission system at the same time. They staggered the first few so that they could isolate the changes and get a sense of what impact those changes had on sales.

## Lessons Learned

The effort is still going on at the time of this writing, but the team has already learned several lessons:

**Not all problems require a technical solution.** The team found that simplifying the commission rules helped reduce the amount of time required to process commissions a great deal and confirmed their suspicions that unique rules did not have a large impact on sales agent behavior. Even so, the team decided it would be good to consolidate all the processing on a single system.

**You may not realize how good you have it on your side of the fence.** As the team started their search for a new commission system, they decided to include the five purchased systems they were already using to administer parts of their commissions. They found that as a result of simplifying commission rules, one of the systems they already had fit the bill nicely for what they were trying to do. They had to upgrade that commission system several versions, but once they

did, they found that their work mainly consisted of creating new interfaces for any data they didn't have in that system already.

**Commercial off-the-shelf (COTS) systems often contain good industry practices.** When the team picked the commission system, they found they could use that unit's commission process for all the other units as well. That process was one suggested by the developers of the existing commission system. Switching to that process for all the units provided even more improvement in overall commissions processing and eased the transition effort since the team didn't have to come up with new processes for each unit.

**Don't forget change management.** Just because the team didn't have to come up with new processes didn't make the change completely turnkey. The commissions team did not have much trouble with the change, since over half of the team was involved on the project to switch commission systems, but they had a bit of change management to do with the agents. When they found out that commission structures were changing, most of the agents complained. Loudly. The team found that the best way to help the agents adapt to the change was to give them examples of their own commissions under both the old and the new structures. Most of the agents found that their commissions would stay consistent, or even increase. The only agents whose commissions decreased were those few who had studied the old plans enough to use loopholes to maximize their revenue. These agents were among the highest compensated but were only middle of the pack in terms of actual sales.

**Don't overlook interdependencies with other efforts.** The team originally thought they would have to do a lot of work to interface with a new set of systems for each unit they brought onto the new commission system. Shortly into the project, the team caught wind that the accounting and new business systems were also undergoing projects to make things more uniform. The commissions team got together with the other two teams and synced their rollout plans so they affected the same units in the same order, though not necessarily at the same time. That meant that the commissions team did not have to build new interfaces for every additional unit; they just had to revise the ones they had already built.

*This page intentionally left blank*

# Index

## B

## C

## D

## M



## N

## O



## P

## Q



## R

## S

## T



## U



## V

## W



## Y