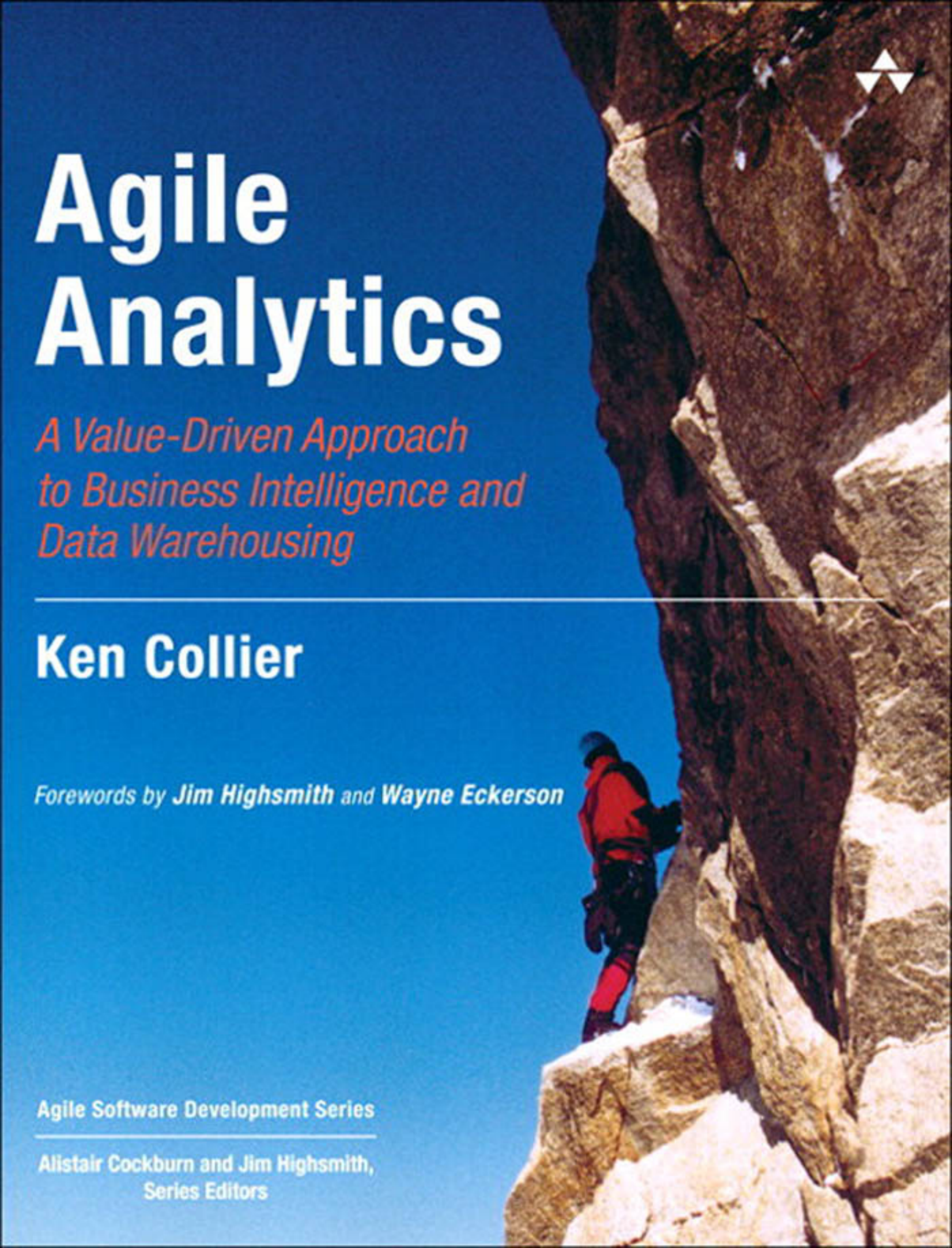

Agile Analytics

A Value-Driven Approach to Business Intelligence and Data Warehousing

Ken Collier

Forewords by ***Jim Highsmith*** *and* ***Wayne Eckerson***

Agile Software Development Series

Alistair Cockburn and Jim Highsmith, Series Editors

Praise for *Agile Analytics*

"This book does a great job of explaining why and how you would implement Agile Analytics in the real world. Ken has many lessons learned from actually implementing and refining this approach. Business Intelligence is definitely an area that can benefit from this type of discipline."

—Dale Zinkgraf, Sr. Business Intelligence Architect

"One remarkable aspect of *Agile Analytics* is the breadth of coverage—from product and backlog management to Agile project management techniques, from self-organizing teams to evolutionary design practices, from automated testing to build management and continuous integration. Even if you are not on an analytics project, Ken's treatment of this broad range of topics related to products with a substantial data-oriented flavor will be useful for and beyond the analytics community."

—Jim Highsmith, Executive Consultant, ThoughtWorks, Inc., and author of *Agile Project Management*

"Agile methods have transformed software development, and now it's time to transform the analytics space. *Agile Analytics* provides the knowledge needed to make the transformation to Agile methods in delivering your next analytics projects."

—Pramod Sadalage, coauthor of *Refactoring Databases: Evolutionary Database Design*

"This book captures the fundamental strategies for successful business intelligence/analytics projects for the coming decade. Ken Collier has raised the bar for analytics practitioners—are you up to the challenge?"

—Scott Ambler, Chief Methodologist for Agile and Lean, IBM Rational Founder, Agile Data Method

"A sweeping presentation of the fundamentals that will empower teams to deliver high-quality, high-value, working business intelligence systems far more quickly and cost effectively than traditional software development methods."

—Ralph Hughes, author of *Agile Data Warehousing*

This page intentionally left blank

Agile Analytics

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.

AGILE ANALYTICS

A VALUE-DRIVEN APPROACH TO BUSINESS INTELLIGENCE AND DATA WAREHOUSING

KEN COLLIER

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data

Collier, Ken, 1960–
Agile analytics : a value-driven approach to business intelligence and data warehousing / Ken Collier.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-321-50481-4 (pbk. : alk. paper)
1. Business intelligence—Data processing. 2. Business intelligence—Computer programs. 3. Data warehousing. 4. Agile software development. 5. Management information systems. I. Title.
HD38.7.C645 2012
658.4'72—dc23

2011019825

ISBN-13: 978-0-321-50481-4
ISBN-10: 0-321-50481-X
Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, July 2011

This book is dedicated to my wife and best friend, Beth, who never once asked, "How come it's taking you so long to finish that darn book?"

This page intentionally left blank

Contents

This page intentionally left blank

Foreword
by Jim Highsmith

I was introduced to Ken Collier through a mutual friend about seven years ago. We started meeting for coffee (a two-person Agile group in Flagstaff, Arizona) every week or so to talk about software development, a sprinkling of Agile here and there, skiing, mountain biking, and Ken's analytics projects. Early on, as Ken talked about a project that was faltering and I talked about Agile, he decided to try out Agile on his next project. As he quipped, "It couldn't be worse!"

Over the years I've heard every reason imaginable why "Agile won't work in my company because we are different." Ken never had that attitude and from the beginning kept trying to figure out not *if* Agile would work on business intelligence and data warehousing projects, but *how* it would work. Ken saw each impediment as an opportunity to figure out an Agile way to overcome it. From developing user stories that traversed the entire analytics software stack, to figuring out how to do continuous integration in that same diverse stack, Ken has always *been* Agile, just as he was learning to *do* Agile. Today, Ken champions the cause of *being* Agile and not just *doing* Agile.

Over subsequent analytics projects, one that ran for over three years, delivering releases every quarter, Ken took the fundamental Agile management and development practices and came up with innovative ways to apply them. Business intelligence and data warehousing developers have been reluctant to embrace Agile (although that is changing) in part because it wasn't clear how to apply Agile to these large, data-centric projects. However, analytics projects suffered from the same problems as more typical IT projects—they took too long, cost too much, and didn't satisfy their customers. In our current turbulent business era these kinds of results are no longer acceptable.

One remarkable aspect of *Agile Analytics* is the breadth of coverage—from product and backlog management, to Agile project management techniques, to self-organizing teams, to evolutionary design practices, to automated testing, to build management and continuous integration. Even if you are not on an analytics project, Ken's treatment of this broad range of topics related to products with a substantial data-oriented flavor will be useful for and beyond the analytics community.

In each subject area he has taken the basic Agile practices and customized them to analytics projects. For example, many BI and data warehouse teams are far behind their software development counterparts in configuration management. With execution code in Java, Ruby, and other languages, stored procedures, SQL, and tool-specific code in specialized tools, analytics teams often have poor "code" management practices. Ken spends several chapters on reviewing techniques that software developers have been using and showing how those techniques can be adapted to an analytics environment. Ken often asks analytics teams, "If your servers went down hard today, how long would it take you to rebuild?" The responses he typically receives vary from a few weeks to never! The automation of the build, integration, and test process is foreign to many analytics teams, so Ken spends a chapter each on version control and build automation, showing how to build a fast-paced continuous integration environment.

The book also devotes a chapter to explaining how to customize test-driven development (TDD) to an analytics environment. Comprehensive, automated testing—from unit to acceptance—is a critical piece of Agile development and a requirement for complete continuous integration.

The breadth of Ken's topic coverage extends to architecture. While he advocates architecture evolution (and evolutionary design is covered in Chapter 6, "Evolving Excellent Design"), he describes architectural patterns that are adaptive. In Chapter 6 he introduces an adaptable analytics architecture, one that he used on a large project in which change over time was a key part of the challenge. This architecture advocates a "data pull" in contrast to the traditional "data push" approach, much like Kanban systems.

What I like about Ken's book can be summarized by three points: (1) It applies Agile principles and practices to analytics projects; (2) it addresses technical and management practices (doing Agile) and core Agile principles (being Agile); and (3) it covers an astonishingly wide range of topics—from architecture to build management—yet it's not at all superficial. This is quite an accomplishment. Anyone participating in data-centric or business analytics projects will benefit from this superb book.

—*Jim Highsmith*
Executive Consultant
Thoughtworks, Inc.

FOREWORD
BY WAYNE ECKERSON

Several years ago, I spearheaded the development of Web sites for The Data Warehousing Institute's local chapters. I had established the program two years earlier and worked closely with many of the officers to grow the chapters and host events.

As the "business driver" of the project, I knew exactly what functionality the chapter Web sites needed. I had researched registration and collaboration systems and mapped their capabilities to my feature matrix. I was ready to wheel and deal and get a new system up and running in three months.

Unfortunately, the project went "corporate." The president assigned someone to manage the project, an IT person to collect requirements, and a marketing person to coordinate integration with our existing Web site. We established a regular time to meet and discuss solutions. In short order, the project died.

My first sense of impending doom came when I read the requirements document compiled by the IT developer after I had e-mailed her my requirements and had a short conversation. When I read the document—and I'm technically astute—I no longer recognized my project. I knew that anyone working from the document (i.e., vendor or developer) would never get close to achieving the vision for the Web sites that I felt we needed.

This experience made me realize how frustrated business people get with IT's traditional approach to software development. Because I witnessed how IT translates business requirements into IT-speak, I now had a greater understanding of why so many business intelligence (BI) projects fail.

Agile to the rescue. When I first read about Agile development techniques, I rejoiced. Someone with a tad of business (and common) sense had finally infiltrated the IT community. Everything about the methodology made perfect sense. Most important, it shifts the power in a development project from the IT team to business users for whom the solution is being built!

However, the Agile development methodology was conceived to facilitate software projects for classic transaction-processing applications.

Unfortunately, it didn't anticipate architecture- and data-laden development projects germane to business intelligence.

Fortunately, BI practitioners like Ken Collier have pioneered new territory by applying Agile methods to BI and have lived to tell about their experiences. Ken's book is a fount of practical knowledge gleaned from real project work that shows the dos and don'ts of applying Agile methods to BI.

Although the book contains a wealth of process knowledge, it's not a how-to manual; it's really more of a rich narrative that gives would-be Agile BI practitioners the look, feel, smell, and taste of what it's like to apply Agile methods in a real-world BI environment. After you finish reading the book, you will feel as if you have worked side by side with Ken on a project and learned from the master.

—*Wayne Eckerson*
Founder and President
BI Leadership Forum
Formerly Director of Research and Services, TDWI

PREFACE

WHEN DW/BI PROJECTS GO BAD

Most data warehouse developers have experienced projects that were less than successful. You may even have experienced the pain of a failed or failing project. Several years ago I worked for a midsize company that was seeking to replace its existing homegrown reporting application with a properly architected data warehouse. My role on the project was chief architect and technical lead. This project ended very badly and our solution was ultimately abandoned. At the outset the project appeared poised for success and user satisfaction. However, in spite of the best efforts of developers, project managers, and stakeholders, the project ran over budget and over schedule, and the users were less than thrilled with the outcome. Since this project largely motivated my adaptation of Agile principles and practices to data warehouse and business intelligence (DW/BI) development, I offer this brief retrospective to help provide a rationale for the Agile DW/BI principles and practices presented throughout this book. It may have some similarities to projects that you've worked on.

About the Project

This section summarizes the essential characteristics of the project, including the following:

- **Existing application.** The company's existing reporting application was internally referred to as a "data warehouse," which significantly skewed users' understanding of what a data warehouse application offers. In reality the data model was a replication of parts of one of the legacy operational databases. This replicated database did not include any data scrubbing and was wrapped in a significant amount of custom Java code to produce the reports required. Users had, at various times, requested new custom reports, and the application had become overburdened with highly specialized and seldom used reporting features. All of the reports could be classified as canned reports. The system was not optimized for analytical activities, and advanced analytical capabilities were not provided.

- **Project motivation.** Because the existing "data warehouse" was not architected according to data warehousing best practices, it had reached the practical limits of maintainability and scalability needed to continue meeting user requirements. Additionally, a new billing system was coming online, and it was evident that the existing system could not easily be adapted to accommodate the new data. Therefore, there was strong executive support for a properly designed data warehouse.
- **External drivers.** The data warehousing project was initially envisioned by a sales team from one of the leading worldwide vendors of data warehousing and business intelligence software. In providing guidance and presales support, this sales team helped the project sponsors understand the value of eliciting the help of experienced business intelligence consultants with knowledge of industry best practices. However, as happens with many sales efforts, initial estimates of project scope, cost, and schedule were overly ambitious.
- **Development team.** The development team consisted exclusively of external data warehousing contractors. Because the company's existing IT staff had other high-priority responsibilities, there were no developers with deep knowledge of the business or existing operational systems. However, the development team had open access to both business and technical experts within the company as well as technology experts from the software vendor. While initial discovery efforts were challenging, there was strong participation from all stakeholders.
- **Customer.** The primary "customer" for the new data warehouse was the company's finance department, and the project was sponsored by the chief financial officer. They had a relatively focused business goal of gaining more reliable access to revenue and profitability information. They also had a substantial volume of existing reports used in business analysis on a routine basis, offering a reasonable basis for requirements analysis.
- **Project management.** Project management (PM) responsibilities were handled by corporate IT using traditional Project Management Institute/Project Management Body of Knowledge (PMBOK) practices. The IT group was simultaneously involved in two other large development projects, both of which had direct or indirect impact on the data warehouse scope.
- **Hosted environment.** Because of limited resources and infrastructure, the company's IT leadership had recently decided to partner with an application service provider (ASP) to provide hosting services for newly developed production systems. The data warehouse

was expected to reside at the hosting facility, located on the west coast of the United States, while the company's headquarters were on the east coast. While not insurmountable, this geographic separation did have implications for the movement of large volumes of data since operational systems remained on the east coast, residing on the corporate IT infrastructure.

Project Outcome

The original project plan called for an initial data warehouse launch within three months but had an overly ambitious scope for this release cycle. Project completion was a full eight months after project start, five months late! User acceptance testing did not go well. Users were already annoyed with project delays, and when they finally saw the promised features, there was a large gap between what they expected and what was delivered. As is common with late projects, people were added to the development team during the effort to try to get back on track. As Fred Brooks says, "Adding more people to a late project only makes it later" (Brooks 1975). Ultimately, project costs far exceeded the budget, users were unsatisfied, and the project was placed on hold until further planning could be done to justify continued development.

Retrospective

So who was to blame? Everybody! Users felt that the developers had missed the mark and didn't implement all of their requirements. Developers felt that the users' expectations were not properly managed, and the project scope grew out of control. Project sponsors felt that the vendors overpromised and underdelivered. Vendors felt that internal politics and organizational issues were to blame. Finally, many of the organization's IT staff felt threatened by lack of ownership and secretly celebrated the failure.

The project degenerated into a series of meetings to review contracts and project documents to see who should be held responsible, and guess what? Everyone involved was partially to blame. In addition to the normal technical challenges of data warehouse development, the following were identified as root causes of project failure:

- The contract did not sufficiently balance scope, schedule, and resources.
- Requirements were incomplete, vague, and open-ended.
- There were conflicting interpretations of the previously approved requirements and design documents.

- Developers put in long nights and weekends in chaotic attempts to respond to user changes and new demands.
- The technical team was afraid to publicize early warning signs of impending failure and continued trying to honor unrealistic commitments.
- Developers did not fully understand the users' requirements or expectations, and they did not manage requirements changes well.
- Users had significant misconceptions about the purpose of a data warehouse since existing knowledge was based on the previous reporting application (which was not a good model of a warehouse).
- Vendors made ambitious promises that the developers could not deliver on in the time available.
- The project manager did not manage user expectations.
- IT staff withheld important information from developers.
- The ASP partner did not provide the level of connectivity and technical support the developers expected.

Hindsight truly is 20/20, and in the waning days of this project several things became apparent: A higher degree of *interaction* among developers, users, stakeholders, and internal IT experts would have ensured accurate understanding on the part of all participants. Early and frequent *working software*, no matter how simplistic, would have greatly reduced the users' misconceptions and increased the accuracy of their expectations. Greater emphasis on *user collaboration* would have helped to avoid conflicting interpretations of requirements. A project plan that focused on *adapting to changes* rather than meeting a set of "frozen" contractual requirements would have greatly improved user satisfaction with the end product. In the end, and regardless of blame, the root cause of this and many other data warehousing project failures is the disconnect in understanding and expectations between developers and users.

About This Book

About the same time I was in the throes of the painful and failing project just described, I met Jim Highsmith, one of the founding fathers of the Agile movement, author of *Adaptive Software Development*, *Agile Software Development Ecosystems*, and *Agile Project Management* and one of the two series editors for the Agile Software Development Series of which this book is a part. Jim listened to my whining about our project difficulties and gave me much food for thought about how Agile methods might be adapted to DW/BI systems development. Unfortunately, by the time I met Jim it was too late

to right that sinking ship. However, since then Jim and I have become good friends, exchanging ideas over coffee on a mostly weekly basis. Well, mostly he shares good ideas and I do my best to absorb them. Jim has become my Agile mentor, and I have devoted my professional life since we first met to ensuring that I never, ever work on another failing DW/BI project again. Now that may seem like an audacious goal, but I believe that (a) life is too short to suffer projects that are doomed to fail; (b) Agile development is the single best project risk mitigation approach we have at our disposal; and (c) Agile development is the single best means of innovating high-value, high-quality, working DW/BI systems that we have available. That's what this book is about:

- Mitigating DW/BI project risk
- Innovating high-value DW/BI solutions
- Having fun!

Since my last painful project experience I have had many wonderful opportunities to adapt Agile development methods to the unique characteristics of DW/BI systems development. Working with some very talented Agile DW/BI practitioners, I have successfully adapted, implemented, and refined a comprehensive set of project management and technical practices to create the Agile Analytics development method.

This adaptation is nontrivial as there are some very significant and unique challenges that we face that mainstream software developers do not. DW/BI developers deal with a hybrid mix of integrating commercial software and writing some custom code (ETL scripting, SQL, MDX, and application programming are common). DW/BI development teams often have a broad and disparate set of skills. DW/BI development is based on large data volumes and a complex mixture of operational, legacy, and specialty systems. The DW/BI systems development platform is often a high-end dedicated server or server cluster, making it harder to replicate for sandbox development and testing. For these reasons and more, Agile software development methods do not always easily transfer to DW/BI systems development, and I have met a few DW/BI developers who have given up trying. This book will introduce you to the key technical and project management practices that are essential to Agile DW/BI. Each practice will be thoroughly explained and demonstrated in a working example, and I will show you how you might modify each practice to best fit the uniqueness of your situation.

This book is written for three broad audiences:

- DW/BI practitioners seeking to learn more about Agile techniques and how they are applied to the familiar complexities of DW/BI development. For these readers I provide the details of Agile technical and project management techniques as they relate to business intelligence and data-centric projects.
- Agile practitioners who want to know how to apply familiar Agile practices to the complexities of DW/BI systems development. For these readers I elaborate upon the traits of business intelligence projects and systems that make them distinctly different from software development projects, and I show how to adapt Agile principles and practices to these unique characteristics.
- IT and engineering management who have responsibility for and oversight of program portfolios, including data warehousing, business intelligence, and analytics projects. This audience may possess neither deep technical expertise in business intelligence nor expertise in Agile methods. For these readers I present an introduction to an approach that promises to increase the likelihood of successful projects and delighted customers.

Although this book isn't a primer on the fundamentals of DW/BI systems, I will occasionally digress into coverage of DW/BI fundamentals for the benefit of the second audience. Readers already familiar with business intelligence should feel free to skip over these sections.

By the way, although I'm not an expert in all types of enterprise IT systems, such as enterprise resource planning (ERP) implementations, I have reason to believe that the principles and practices that make up Agile Analytics can be easily adapted to work in those environments as well. If you are an IT executive, you might consider the broader context of Agile development in your organization.

WHY AN AGILE DW/BI BOOK?

In the last couple of years the Agile software development movement has exploded. Agile success stories abound. Empirical evidence continues to increase and strongly supports Agile software development. The Agile community has grown dramatically during the past few years, and many large companies have adopted agility across their IT and engineering departments. And there has been a proliferation of books published about various aspects of Agile software development.

Unfortunately, the popularity of Agile methods has been largely lost on the data and business intelligence communities. For some strange reason the data community and software development community have always tended to grow and evolve independently of one another. Big breakthroughs that occur in one community are often lost on the other. The object-oriented boom of the 1990s is a classic example of this. The software development community has reaped the tremendous benefits of folding object orientation into its DNA, yet object-oriented database development remains peripheral to the mainstream for the data community.

Whenever I talk to groups of DW/BI practitioners and database developers, the common reaction is that Agile methods aren't applicable to data-centric systems development. Their arguments are wide and varied, and they are almost always based on myths, fallacies, and misunderstandings, such as "It is too costly to evolve and change a data model. You must complete the physical data model before you can begin developing reports and other user features."

The reality is that there is nothing special about data-centric systems that makes Agile principles irrelevant or inappropriate. The challenge is that Agile practices must be adapted, and a different tool set must be adopted for data-centric systems development. Although many of the current books on Agile concepts and techniques are directly relevant to the data community, most of them do not speak directly to the data-minded reader. Unfortunately, many current Agile books are too narrowly focused on new, greenfield software development using all the latest platforms, frameworks, and programming languages. It can be difficult for readers to extrapolate the ideas presented in these books to database development, data warehouse development, ERP implementation, legacy systems development, and so forth.

Agile author and database expert Scott Ambler has written books on Agile database development and database refactoring (a distinctly Agile practice) to engage the database community in the Agile dialogue. Similarly, I've written this book to engage the DW/BI community in the Agile movement because Agile is simply a better way to work on large, complex DW/BI systems. In 2008 Ralph Hughes's book *Agile Data Warehousing* hit the shelves (Hughes 2008). Ralph does a great job of adapting Scrum and eXtreme Programming (XP) techniques to the nuances of data warehousing, and many of those concepts are also present in this book. Additionally, this book aims to dive into many of the technical practices that are needed to develop in an Agile manner.

WHAT DO I MEAN BY AGILE ANALYTICS?

A word about terminology: I've chosen the title *Agile Analytics* more because it's catchy and manageable than because it precisely captures my focus. Face it, *Agile Data Warehousing, Business Intelligence, and Analytics* would be a mouthful. By and large the data warehousing community has come to use the term *data warehousing* to refer to back-end management and preparation of data for analysis and *business intelligence* to refer to the user-facing front-end applications that present data from the warehouse for analysis. The term *analytics* is frequently used to suggest more advanced business intelligence methods involving quantitative analysis of data (e.g., predictive modeling, statistical analysis, etc.). Moreover, the industry term *business intelligence* is sometimes an ambiguous and broadly encompassing term that includes anything to do with data-driven business processes (business performance management, customer relationship management, etc.) or decision support (scorecards, dashboards, etc.).

My use of the moniker Agile Analytics should not imply that Agile methods are applicable only to a certain class of user-facing BI application development. Agile methods are applicable and adaptable to data warehouse development as well as business intelligence and analytical application development. For many people Agile BI development tends to be easier to imagine, since it is often assumed that the data warehouse has been built and populated. Certainly a preexisting data warehouse simplifies the effort required to build BI applications. However, you should not take this to mean that the data warehouse must be completed prior to building BI applications. In fact, Agile Analytics is a user-value–driven approach in which high-valued BI capabilities drive the evolutionary development of the data warehouse components needed to support those capabilities. In this way we avoid overbuilding the warehouse to support more than its intended purpose.

In this book I focus primarily on the core of most flavors of DW/BI systems, the data warehouse. My use of the term *business intelligence* or *BI* throughout this book should be assumed to include analytic as well as reporting and querying applications. When I use the term *DW/BI system*, you should infer that I mean the core data warehouse along with any presentation applications that are served by the warehouse such as a finance dashboard, a forecasting portal, or some other BI application. However, the DW/BI acronym is somewhat clunky, and I may occasionally use BI alone. In most of these cases you should assume that I mean to include relevant DW components as well. I'll also address some of the advanced BI concepts like data mining

and data visualization. I'll leave it to the reader to extrapolate the practices to more specific BI projects such as CRM implementations. The principles still apply.

WHO SHOULD READ THIS BOOK?

An Agile DW/BI team is made up of more than just developers. It includes the customer (user) community, who provide requirements; the business stakeholder community, who are monitoring the impact of the BI system on business improvements; and the technical community, who develop, deploy, and support the DW/BI system. These communities are connected by a project manager, a business analyst (or product owner), and an executive sponsor. Each of these communities plays a crucial role in project success, and each of these communities requires a well-defined set of Agile practices to be effective in its role. This book is intended for both business and technical readers who are involved in one or more of the communities described.

Not everything in the book is meant for everyone on the list, but there is something here for everyone. I have worked with many organizations that seek Agile training, mentoring, and coaching. Occasionally I have to dispel the myth that agility applies only to developers and techies.

At one company with which I was invited to work, the executive who sponsored the training said something like, "If our engineers could just start doing Agile development, we could finish projects faster and our customers would be happier." This statement represents some unfortunate misconceptions that can be a buzzkill for Agile teams.

First, successful agility requires a change in the mind-set of all team members. Customer community members must understand that their time is required to explore and exercise newly completed features, and to provide continuous input and feedback on the same. Management community members must adapt their expectations as project risk and uncertainty unfolds, and as the team adapts to inevitable change. The technical community must learn a whole new way of working that involves lots of discipline and rigor. And the project interface community must be committed to daily project involvement and a shift in their role and contribution to project success.

Second, Agile doesn't always mean faster project completion. Even the best project teams still have a finite capacity to complete a scope of work. Agility is not a magic wand that makes teams work faster. Agile practices do steer

teams to focus on the high-value and riskiest features early. Therefore, it is possible that an Agile DW/BI system can be launched into production earlier, as soon as the most critical features are complete and accepted. However, I would caution against expecting significantly faster project cycles, especially in the beginning. On the other hand, you should expect a significant increase in quality and customer delight over traditional DW/BI development approaches.

The bottom line is that successful adoption of Agile DW/BI requires awareness, understanding, and commitment from the members of all of the aforementioned project communities. For this reason I have tried to design this book to provide something relevant for everyone.

How This Book Is Organized

This book is divided into two parts. Part I, "Agile Analytics: Management Methods," is focused on Agile project management techniques and delivery team coordination. It includes the following chapters:

- Chapter 1, "Introducing Agile Analytics," provides an overview and baseline for this DW/BI approach.
- Chapter 2, "Agile Project Management," introduces an effective collection of practices for chartering, planning, executing, and monitoring an Agile Analytics project.
- Chapter 3, "Community, Customers, and Collaboration," introduces a set of guidelines and practices for establishing a highly collaborative project community.
- Chapter 4, "User Stories for BI Systems," introduces the story-driven alternative to traditional requirements analysis and shows how use cases and user stories drive the continuous delivery of value.
- Chapter 5, "Self-Organizing Teams Boost Performance," introduces an Agile style of team management and leadership as an effective alternative to more traditional command-and-control styles.

This first part is written for everyone involved in an Agile Analytics project, from executive sponsors, to project managers, to business analysts and product owners, to technical leads and delivery team members. These chapters establish a collection of core practices that shape the way an Agile project community works together toward a successful conclusion.

Part II of the book, "Agile Analytics: Technical Methods," is focused on the technical methods that are necessary to enable continuous delivery of

business value at production-quality levels. This part includes the following chapters:

- Chapter 6, "Evolving Excellent Design," shows how the evolutionary design process works and how to ensure that it results in higher-quality data models and system components with minimal technical debt.
- Chapter 7, "Test-Driven Data Warehouse Development," introduces a collection of practices and tools for automated testing, and for taking a test-first approach to building data warehouse and business intelligence components.
- Chapter 8, "Version Control for Data Warehousing," introduces a set of techniques and tools for keeping the entire DW/BI system under version control and configuration management.
- Chapter 9, "Project Automation," shows how to combine test automation and version control practices to establish an automated continuous integration environment that maintains confidence in the quality of the evolving system.
- Chapter 10, "Final Words," takes a look at some of the remaining factors and considerations that are critical to the successful adoption of an Agile Analytics approach.

I think of this part as a collection of modern development practices that should be used on every DW/BI project, be it Agile or traditional (e.g., "waterfall"). However, these technical practices are essential when an Agile Analytics approach is taken. These methods establish the minimally sufficient set of technical practices needed to succeed in the continuous, incremental, and evolutionary delivery of a high-value DW/BI system.

Of course, these technical chapters should be read by technical team leads and delivery team members. However, I also recommend that nontechnical project team members read the introductory sections of each of these chapters. Doing so will help nontechnical members establish a shared understanding of the purpose of these practices and appreciate the value of the technical team's efforts to apply them.

How Should You Read This Book?

I like to think of Agile Analytics techniques as supporting one of the following focal points:

- **Agile DW/BI management**: the set of practices that are devoted to how you run your project, including precursors to agility, Agile project management methods, the Agile team, developer-user interface, and so on
- **Agile DW/BI technical methods**: the set of practices that are devoted to the development and delivery of a high-value, high-quality, working DW/BI system, including specific technical practices like story-driven development, test-driven development, build automation, code management, refactoring, and so on

The chapters are organized into these major sections. Each chapter is dedicated to a key practice or related set of practices, beginning with an executive-level overview of the salient points of the chapter and progressing into deeper coverage of the topic. Some of the chapter topics are rich enough to deserve to be entire books. In these cases, my aim is to give the reader a solid understanding of the topic, and ideally the motivation needed for a deeper self-study of its mechanics.

If you are reading this to gain a high-level understanding of Agile DW/BI, the initial overview at the beginning of each chapter will suffice. My goal in these overviews is to provide an accurate portrayal of each of the Agile DW/BI practices, but these sections aren't intended to give you all the techniques needed to apply the practice.

If you are a data warehouse manager, project sponsor, or anyone who needs to have a good working understanding of the practices without getting bogged down in the technical details, I recommend reading the middle sections of each chapter, especially the project management chapters. These sections are designed to provide a deep enough understanding of the topic to either use the techniques or understand how they are used on your project.

If you are a member of the day-to-day project team (project managers, technical team members, business analysts, product managers, etc.), I recommend reading the details and examples in each of the project management chapters (Part I, "Agile Analytics: Management Methods"). These are designed to give you a concrete set of techniques to apply in your release planning, iteration planning, and all other project management and user collaboration activities. If you are a member of the technical community, the chapters in Part II, "Agile Analytics: Technical Methods," are intended for you.

A word about DW/BI technologies: I am a technology agnostic. I have done DW/BI development using a variety of technology stacks that are IBM-DB2-centric, Oracle-centric, SAS-centric, and Microsoft-centric, as well as a variety of hybrid technology stacks. While some technologies may lend themselves to Agile DW/BI better than others, I am confident that the guiding principles and practices introduced in this book are technology-independent and can be effective regardless of your tool choices.

As this book goes to press, there are an increasing number of data warehouse and business intelligence tool vendors that are branding their products as Agile. Tools and tool suites from forward-thinking vendors such as WhereScape, Pentaho, Balanced Insight, and others offer some exciting possibilities for enabling agility. While I do not believe that you must have these types of tools to take an Agile approach, they certainly do offer some powerful benefits to Agile delivery teams. The Agile software development community has greatly benefited from tools that help automate difficult development activities, and I look forward to the benefits that our community stands to gain from these vendors. At the same time I would caution you not to believe that you must have such tools before you can start being Agile. Instead, I encourage you to get started with Agile techniques and practices and adopt tools incrementally as you determine that they are of sufficient benefit.

This page intentionally left blank

Acknowledgments

I would never have gotten the experience and knowledge I needed to write this book without the contributions of several key people. These friends and colleagues have my respect and gratitude for the many valuable interactions I've had with them, and the collaborations that ultimately resulted in the Agile Analytics approach.

Foremost, my good friend Jim Highsmith has been my trusted adviser and mentor since the beginning of my Agile journey. Jim was just starting to write the first edition of *Agile Project Management* when I first met him, and he made book-writing look so easy that I decided to give it a try. As it turns out, it's much harder than he makes it look. My weekly breakfast discussions with Jim were critical in shaping the concepts in this book. He voluntarily served as my developmental editor, reviewing early drafts of sections and chapters and helping me pull things together in a more cohesive and coherent fashion. Jim continues to challenge my assumptions and gives me new ideas and new ways to think about the complexities of development. He also didn't give up on me when book-writing wasn't my highest priority. Thanks, Jim.

Jim introduced me to Luke Hohmann at a time when Luke was looking for somebody with both data warehousing experience and Agile knowledge. Luke is one of the most visionary people I've ever met. I was fortunate enough to be the chief architect for one of Luke's innovative ideas: a complex, hosted, enterprise DW/BI product offering from one of Luke's clients. The complexity of this project and Luke's deep knowledge of Agile techniques challenged me (and our team) to figure out how to apply Agile software methods to the nuances of DW/BI development. The concepts in this book stem from that experience and have been refined and matured on subsequent projects. Luke has become a great friend over the past seven years, and I value his wisdom and vision. Thanks, Luke.

My team on the aforementioned project remains one of the best Agile teams I have yet experienced either as a participant or as an Agile trainer. This team included David Brink, Robert Daugherty, James Slebodnick, Scott Gilbert, Dan O'Leary, Jonathon Golden, and Ricardo Aguirre. Each team member brought a special set of skills and perspectives, and over that first three-plus-year-long project these friends and teammates helped me figure

out effective ways to apply Agile techniques to DW/BI development. I've since had other project opportunities to work with many of these friends, further refining Agile Analytics concepts. These team members deserve much of the credit for validating and tweaking Agile Analytics practices in a complex and real-life situation. Thanks, guys.

Jim Highsmith also introduced me to Scott Ambler along the way. Scott has led the charge in applying Agile to data-centric systems development. Fortunately for all of us, Scott is a prolific writer who freely shares his ideas in his many books and on his ambysoft.com Web site. I have benefited greatly from the conversations I've had with him, as well as from his writings on Agile Modeling, Agile Data, Agile Unified Process, and Database Refactoring (together with Pramod Sadalage). In the early days of my focus on Agile in DW/BI, Scott and I regularly lamented our perceptions that the data community wasn't paying attention to the benefits of agility, while the software community wasn't paying attention to the unique challenges of database development and systems integration. Scott gave much of his time reviewing this book. He has given me much to think about and shared ideas with me that I might otherwise have missed. Thanks, Scott.

I don't think I truly understood what it means for somebody to have "the patience of a saint" before working with Addison-Wesley editor Chris Guzikowski and editorial assistant Raina Chrobak. As it turns out, I am a painfully slow author who is not very good at applying Agile principles to book-writing deadlines. Huge thanks go to Raina and Chris, who were amazingly patient as I slipped deadline after deadline. I hope I have future opportunities to redeem myself as an author.

Ralph Hughes's *Agile Data Warehousing* book hit the shelves as I was writing this book. Ralph and I were acquainted at that time and since have become friends and colleagues. I am grateful for his work in this area and for the discussions I've had with him and the experiences he has shared. Although I have tried not to duplicate what Ralph has already published, I am confident that our approaches are consistent with and complementary to one another. I look forward to future collaborations with Ralph as our ideas mature and evolve.

Finally, the ideas presented in this book have benefited tremendously from smart and thoughtful people willing to review its early drafts and give me guidance. In addition to Scott's and Jim's reviews, special thanks go to Jonathon Golden, my go-to guru on project automation, and Israel Gat, expert on Agile leadership and technical debt. My gratitude also goes to DW/BI experts Wayne Eckerson and Dale Zinkgraf and to Agile data expert Pramod Sadalage for their feedback. Their contributions were invaluable.

About the Author

Ken Collier got excited about Agile development in 2003 and was one of the first to start combining Agile methods with data warehousing, business intelligence, and analytics. These disciplines present a unique set of challenges to the incremental/evolutionary style of Agile development. Ken has successfully adapted Agile techniques to data warehousing and business intelligence to create the Agile Analytics style. He continues to refine these ideas as a technical lead and project manager on several Agile DW/BI project teams. Ken also frequently trains data warehousing and business intelligence teams in Agile Analytics, giving him the opportunity to exercise this approach with various technologies, team dynamics, and industry domains. He has been an invited keynote speaker on the subject of Agile DW/BI at several U.S. and international conferences, including multiple TDWI (The Data Warehousing Institute) World Conferences as well as HEDW (Higher Education Data Warehousing) annual conferences.

In nearly three decades of working in advanced computing and technology, Ken has experienced many of the trends that come and go in our field, as well as the ones that truly transform the state of our practices. With an M.S. and Ph.D. in computer science engineering, Ken is formally trained in software engineering, data management, and machine learning. He loves challenging problems in the areas of systems architecture and design, systems/software development lifecycles, project leadership, data warehousing, business intelligence, and advanced analytics. Ken also loves helping organizations adopt and tailor effective approaches and solutions that might not otherwise be apparent. He combines a deep technical foundation with sound business acumen to help bridge the gaps that often exist between technical and business professionals.

Ken is the founder and president of KWC Technologies, Inc., and is a senior consultant with the Cutter Consortium in both the Agile Development and Business Intelligence practice areas. Ken has had the privilege of working as a software engineer for a large semiconductor company. He has spent several years as a tenured professor of computer science engineering. He has directed the data warehousing and business intelligence solutions group for a major consulting firm. And, most recently, he has focused on enabling organizational agility, including Agile software engineering, Agile Analytics, and Agile management and leadership for client companies.

This page intentionally left blank

Chapter 1

Introducing Agile Analytics

Like Agile software development, Agile Analytics is established on a set of core values and guiding principles. It is not a rigid or prescriptive methodology; rather it is a style of building a data warehouse, data marts, business intelligence applications, and analytics applications that focuses on the early and continuous delivery of business value throughout the development lifecycle. In practice, Agile Analytics consists of a set of highly disciplined practices and techniques, some of which may be tailored to fit the unique data warehouse/business intelligence (DW/BI) project demands found in your organization.

Agile Analytics includes practices for project planning, management, and monitoring; for effective collaboration with your business customers and management stakeholders; and for ensuring technical excellence by the delivery team. This chapter outlines the tenets of Agile Analytics and establishes the foundational principles behind each of the practices and techniques that are introduced in the successive chapters in this book.

Agile is a reserved word when used to describe a development style. It means something very specific. Unfortunately, "agile" occasionally gets misused as a moniker for processes that are ad hoc, slipshod, and lacking in discipline. Agile relies on discipline and rigor; however, it is not a heavyweight or highly ceremonious process despite the attempts of some methodologists to codify it with those trappings. Rather, Agile falls somewhere in the middle between just enough structure and just enough flexibility. It has been said that Agile is simple but not easy, describing the fact that it is built on a simple set of sensible values and principles but requires a high degree of discipline and rigor to properly execute. It is important to accurately understand the minimum set of characteristics that differentiate a true Agile process from those that are too unstructured or too rigid. This chapter is intended to leave you with a clear understanding of those characteristics as well as the underlying values and principles of Agile Analytics. These are derived directly from the tried and proven foundations established by the Agile software community and are adapted to the nuances of data warehousing and business intelligence development.

ALPINE-STYLE SYSTEMS DEVELOPMENT

I'm a bit of an armchair climber and mountaineer. I'm fascinated by the trials and travails of climbing high mountains like Everest, Annapurna, and others that rise to over 8,000 meters above sea level. These expeditions are complicated affairs involving challenging planning and logistics, a high degree of risk and uncertainty, a high probability of death (for every two climbers who reach the top of Annapurna, another one dies trying!), difficult decisions in the face of uncontrollable variables, and incredible rewards when success is achieved. While it may not be as adventuresome, building complex business intelligence systems is a lot like high-altitude climbing. We face lots of risk and uncertainty, complex planning, difficult decisions in the heat of battle, and the likelihood of death! Okay, maybe not that last part, but you get the analogy. Unfortunately the success rate for building DW/BI systems isn't very much better than the success rate for high-altitude mountaineering expeditions.

Climbing teams first began successfully "conquering" these high mountains in the 1950s, '60s, and '70s. In those early days the preferred mountaineering style was known as "siege climbing," which had a lot of similarities to a military excursion. Expeditions were led in an autocratic command-and-control fashion, often by someone with more military leadership experience than climbing experience. Climbing teams were supported by the large numbers of porters required to carry massive amounts of gear and supplies to base camp and higher. Mounting a siege-style expedition takes over a year of planning and can take two months or more to execute during the climbing season. Siege climbing is a yo-yo-like affair in which ropes are fixed higher and higher on the mountain, multiple semipermanent camps are established at various points along the route, and loads of supplies are relayed by porters to those higher camps. Finally, with all this support, a small team of summit climbers launches the final push for the summit on a single day, leaving from the high camp and returning to the same. Brilliant teams have successfully climbed mountains for years in this style, but the expeditions are prohibitively expensive, time-consuming to execute, and fraught with heavyweight procedures and bureaucracy.

Traditional business intelligence systems development is a lot like siege climbing. It can result in high-quality, working systems that deliver the desired capabilities. However, these projects are typically expensive, exhibiting a lot of planning, extensive design prior to development, and long development cycles. Like siege-style expeditions, all of the energy goes into one shot at the summit. If the summit bid fails, it is too time-consuming to return to base camp and regroup for another attempt. In my lifetime (and I'm not that old

yet) I've seen multiple traditional DW/BI projects with budgets of $20 million or more, and timelines of 18 to 24 months, founder. When such projects fail, the typical management response is to cancel the project entirely rather than adjust, adapt, and regroup for another "summit attempt."

In the 1970s a new mountaineering method called "alpine-style" emerged, making it feasible for smaller teams to summit these high peaks faster, more cheaply, and with less protocol. Alpine-style mountaineering still requires substantial planning, a sufficient supporting team, and enough gear and supplies to safely reach the summit. However, instead of spending months preparing the route for the final summit push, alpine-style climbers spend about a week moving the bare essentials up to the higher camps. In this style, if conditions are right, summits can be reached in a mere ten days. Teams of two to three climbers share a single tent and sleeping bag, fewer ropes are needed, and the climbers can travel much lighter and faster. When conditions are not right, it is feasible for alpine-style mountaineers to return to base camp and wait for conditions to improve to make another summit bid.

Agile DW/BI development is much like alpine-style climbing. It is essential that we have a sufficient amount of planning, the necessary support to be successful, and an appropriate amount of protocol. Our "summit" is the completion of a high-quality, working business intelligence system that is of high value to its users. As in mountaineering, reaching our summit requires the proper conditions. We need just the right amount of planning—but we must be able to adapt our plan to changing factors and new information. We must prepare for a high degree of risk and uncertainty—but we must be able to nimbly manage and respond as risks unfold. We need support and involvement from a larger community—but we seek team self-organization rather than command-and-control leadership.

Agile Analytics is a development "style" rather than a methodology or even a framework. The line between siege-style and alpine-style mountaineering is not precisely defined, and alpine-style expeditions may include some siege-style practices. Each style is best described in terms of its values and guiding principles. Each alpine-style expedition employs a distinct set of climbing practices that support a common set of values and principles. Similarly, each Agile DW/BI project team must adapt its technical, project management, and customer collaboration practices to best support the Agile values and principles.[1]

1. I'm not the first Agile advocate to discuss the analogy between climbing and Agile development. Jim Highsmith made a similar analogy in his 2000 book, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems* (Highsmith 2000).

Premier mountaineer Ed Viesturs has a formula, or core value, that is his cardinal rule in the big mountains: "Getting to the top is optional. Getting down is mandatory." (Viesturs and Roberts 2006) I love this core value because it is simple and elegant, and it provides a clear basis for all of Ed's decision making when he is on the mountain. In the stress of the climb, or in the midst of an intensely challenging project, we need just such a basis for decision making—our "North Star." In 2000, a group of the most influential application software developers convened in Salt Lake City and formed the Agile Alliance. Through the process of sharing and comparing each of their "styles" of software development, the *Agile Manifesto* emerged as a simple and elegant basis for project guidance and decision making. The Agile Manifesto reads:[2]

> ***Manifesto for Agile Software Development***
>
> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> ***Individuals and interactions*** *over processes and tools*
> ***Working software*** *over comprehensive documentation*
> ***Customer collaboration*** *over contract negotiation*
> ***Responding to change*** *over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more.*

With due respect to the Agile Alliance, of which I am a member, I have adapted the Agile Manifesto just a bit in order to make it more appropriate to Agile Analytics:

> **Manifesto for Agile Analytics Development**
>
> We are uncovering better ways of developing data warehousing and business intelligence systems by doing it and helping others do it. Through this work we have come to value:
>
> **Individuals and interactions** over processes and tools
> **Working DW/BI systems** over comprehensive documentation
> **End-user and stakeholder collaboration** over contract negotiation
> **Responding to change** over following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more.

2. www.agilealliance.org

I didn't want to mess with the original manifesto too much, but it is important to acknowledge that DW/BI systems are fundamentally different from application software. In addition to dealing with large volumes of data, our efforts involve systems integration, customization, and programming. Nonetheless, the Agile core values are very relevant to DW/BI systems development. These values emphasize the fact that our primary objective is the creation of high-quality, high-value, working DW/BI systems. Every activity related to any project either (a) directly and materially contributes to this primary objective or (b) does not. Agile Analytics attempts to maximize a-type activities while acknowledging that there are some b-type activities that are still important, such as documenting your enterprise data model.

What Is Agile Analytics?

Throughout this book I will introduce you to a set of Agile DW/BI principles and practices. These include technical, project management, and user collaboration practices. I will demonstrate how you can apply these on your projects, and how you can tailor them to the nuances of your environment. However, the title of this section is "What Is Agile Analytics?" so I should probably take you a bit further than the mountaineering analogy.

Here's What Agile Analytics Is

So here is a summary of the key characteristics of Agile Analytics. This is simply a high-level glimpse at the key project traits that are the mark of agility, not an exhaustive list of practices. Throughout the remainder of this book I will introduce you to a set of specific practices that will enable you to achieve agility on your DW/BI projects. Moreover, Agile Analytics is a development style, not a prescriptive methodology that tells you precisely what you must do and how you must do it. The dynamics of each project within each organization require practices that can be tailored appropriately to the environment. Remember, the primary objective is a high-quality, high-value, working DW/BI system. These characteristics simply serve that goal:

- **Iterative, incremental, evolutionary.** Foremost, Agile is an iterative, incremental, and evolutionary style of development. We work in short *iterations* that are generally one to three weeks long, and never more than four weeks. We build the system in small *increments* or "chunks" of user-valued functionality. And we *evolve* the working system by adapting to frequent user feedback. Agile development is like driving around in an unfamiliar city; you want to avoid going very far without some validation that you are on the right course.

Short iterations with frequent user reviews help ensure that we are never very far off course in our development.

- **Value-driven development.** The goal of each development iteration is the production of user-valued features. While you and I may appreciate the difficulty of complex data architectures, elegant data models, efficient ETL scripts, and so forth, users generally couldn't care less about these things. What users of DW/BI systems care about is the presentation of and access to information that helps them either solve a business problem or make better business decisions. Every iteration must produce at least one new user-valued feature in spite of the fact that user features are just the tip of the architectural iceberg that is a DW/BI system.
- **Production quality.** Each newly developed feature must be fully tested and debugged during the development iteration. Agile development is not about building hollow prototypes; it is about incrementally evolving to the right solution with the best architectural underpinnings. We do this by integrating ruthless testing early and continuously into the DW/BI development process.[3] Developers must plan for and include rigorous testing in their development process. A user feature is "Done" when it is of production quality, it is successfully integrated into the evolving system, and developers are proud of their work. That same feature is "Done! Done!" when the user accepts it as delivering the right value.
- **Barely sufficient processes.** Traditional styles of DW/BI development are rife with a high degree of ceremony. I've worked on many projects that involved elaborate stage-gate meetings between stages of development such as the transition from requirements analysis to design. These gates are almost always accompanied by a formal document that must be "signed off" as part of the gating process. In spite of this ceremony many DW/BI projects struggle or founder. Agile DW/BI emphasizes a sufficient amount of ceremony to meet the practical needs of the project (and future generations) but nothing more. If a data dictionary is deemed important for use by future developers, then perhaps a digital image of a whiteboard table or a simple spreadsheet table will suffice. Since our primary objective is the production of high-quality, high-value, working systems, we must be able to minimize the amount of ceremony required for other activities.

3. Historically database and data warehouse testing has lacked the rigor, discipline, and automation that have benefited software development efforts (www.ambysoft.com/surveys/dataQualitySeptember2006.html).

- **Automation, automation, automation.** The only way to be truly Agile is to automate as many routine processes as possible. Test automation is perhaps the most critical. If you must test your features and system manually, guess how often you're likely to rerun your tests? Test automation enables you to frequently revalidate that everything is still working as expected. Build automation enables you to frequently build a version of your complete working DW/BI system in a demo or preproduction environment. This helps establish continuous confidence that you are never more than a few hours or days away from putting a new version into production. Agile Analytics teams seek to automate any process that is done more than once. The more you can automate, the more you can focus on developing user features.
- **Collaboration.** Too often in traditional projects the development team solely bears the burden of ensuring that timelines are met, complete scope is delivered, budgets are managed, and quality is ensured. Agile business intelligence acknowledges that there is a broader project community that shares responsibility for project success. The project community includes the subcommunities of users, business owners, stakeholders, executive sponsors, technical experts, project managers, and others. Frequent collaboration between the technical and user communities is critical to success. Daily collaboration within the technical community is also critical. In fact, establishing a collaborative team workspace is an essential ingredient of successful Agile projects.
- **Self-organizing, self-managing teams.** Hire the best people, give them the tools and support they need, then stand aside and allow them to be successful. There is a key shift in the Agile project management style compared to traditional project management. The Agile project manager's role is to enable team members to work their magic and to facilitate a high degree of collaboration with users and other members of the project community. The Agile project team decides how much work it can complete during an iteration, then holds itself accountable to honor those commitments. The Agile style is not a substitute for having the right people on the team.

Guiding Principles

The core values contained in the Agile Manifesto motivate a set of guiding principles for DW/BI systems design and development. These principles often become the tiebreaker when difficult trade-off decisions must be made. Similarly, the Agile Alliance has established a set of principles for

software development.[4] The following Agile Analytics principles borrow liberally from the Agile Alliance principles:

- Our highest priority is to satisfy the DW/BI user community through early and continuous delivery of working user features.
- We welcome changing requirements, even late in development. Agile processes harness change for the DW/BI users' competitive advantage.
- We deliver working software frequently, providing users with new DW/BI features every few weeks.
- Users, stakeholders, and developers must share project ownership and work together daily throughout the project.
- We value the importance of talented and experienced business intelligence experts. We give them the environment and support they need and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- A working business intelligence system is the primary measure of progress.
- We recognize the balance among project scope, schedule, and cost. The data warehousing team must work at a sustainable pace.
- Continuous attention to the best data warehousing practices enhances agility.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Take a minute to reflect on these principles. How many of them are present in the projects in your organization? Do they make sense for your organization? Give them another look. Are they realistic principles for your organization? I have found these not only to be commonsense principles, but also to be effective and achievable on real projects. Furthermore, adherence to these principles rather than reliance on a prescriptive and ceremonious process model is very liberating.

Myths and Misconceptions

There are some myths and misconceptions that seem to prevail among other DW/BI practitioners and experts that I have talked to about this style

4. www.agilemanifesto.org/principles.html

of development. I recently had an exchange on this topic with a seasoned veteran in both software development and data warehousing who is certified at the mastery level in DW/BI and data management and who has managed large software development groups. His misunderstanding of Agile development made it evident that myths and misconceptions abound even among the most senior DW/BI practitioners. Agile Analytics is not:

- **A wholesale replacement of traditional practices.** I am not suggesting that everything we have learned and practiced in the short history of DW/BI systems development is wrong, and that Agile is the new savior that will rescue us from our hell. There are many good DW/BI project success stories, which is why DW/BI continues to be among the top five strategic initiatives for most large companies today. It is important that we keep the practices and methods that work well, improve those that allow room for improvement, and replace those that are problematic. Agile Analytics seeks to modify our general approach to DW/BI systems development without discarding the best practices we've learned on our journey so far.
- **Synonymous with Scrum or eXtreme Programming (XP).** Scrum is perhaps the Agile flavor that has received the most publicity (along with XP) in recent years. However, it is incorrect to say that "Agile was formerly known as eXtreme Programming," as one skeptic told me. In fact, there are many different Agile development flavors that add valuable principles and practices to the broader collective known as Agile development. These include Scrum, Agile Modeling, Agile Data, Crystal, Adaptive, DSDM, Lean Development, Feature Driven Development, Agile Project Management (APM), and others.[5] Each is guided by the core values expressed in the Agile Manifesto. Agile Analytics is an adaptation of principles and practices from a variety of these methods to the complexities of data-intensive, analytics-based systems integration efforts like data warehousing and data mart development.
- **Simply iterating.** Short, frequent development iterations are an essential cornerstone of Agile development. Unfortunately, this key practice is commonly misconstrued as *the* definition of agility. Not long ago I was asked to mentor a development team that had "gone Agile" but wasn't experiencing the expected benefits of agility. Upon closer inspection I discovered that they were planning in four-week "iterations" but didn't expect to have any working features until

5. For a great survey of the various Agile flavors I highly recommend reading *Agile Software Development Ecosystems* (Highsmith 2002).

about the sixth month of the project. Effectively they had divided the traditional waterfall model into time blocks they called iterations. They completely missed the point. The aim of iterative development is to demonstrate working features and to obtain frequent feedback from the user community. This means that every iteration must result in demonstrable working software.

- **For systems integration; it's only for programming.** Much of our effort in DW/BI development is focused on the integration of multiple commercial tools, thereby minimizing the volume of raw programming required. DW/BI tool vendors would have us believe that DW/BI development is simply a matter of hooking up the tools to the source systems and pressing the "Go" button. You've probably already discovered that building an effective DW/BI system is not that simple. A DW/BI development team includes a heterogeneous mixture of skills, including extraction, transformation, load (ETL) development; database development; data modeling (both relational and multidimensional); application development; and others. In fact, compared to the more homogeneous skills required for applications development, DW/BI development is quite complex in this regard. This complexity calls for an approach that supports a high degree of customer collaboration, frequent delivery of working software, and frequent feedback—aha, an Agile approach!
- **An excuse for ad hoc behavior.** Some have mistaken the tenets of Agile development for abandonment of rigor, quality, or structure, in other words, "hacking." This misperception could not be farther from the truth. Agility is a focus on the frequent delivery of high-value, production-quality, working software to the user community with the goal of continuously adapting to user feedback. This means that automated testing and quality assurance are critical components of all iterative development activities. We don't build prototypes; we build working features and then mature those features in response to user input. Others mistake the Agile Manifesto as disdain of documentation, which is also incorrect. Agile DW/BI seeks to ensure that a sufficient amount of documentation is produced. The keyword here is *sufficient*. Sufficiency implies that there is a legitimate purpose for the document, and when that purpose is served, there is no need for additional documentation.

In my work with teams that are learning and adopting the Agile DW/BI development style, I often find that they are looking for a prescriptive methodology that makes it very clear which practices to apply and when. This is a natural inclination for new Agile practitioners, and I will provide some

recommendations that may seem prescriptive in nature. In fact you may benefit initially by creating your own "recipe" for the application of Agile DW/BI principles and practices. However, I need to reemphasize that Agile Analytics is a style, not a methodology and not a framework. Figuratively, you can absorb agility into your DNA with enough focus, practice, and discipline. You'll know you've reached that point when you begin applying Agile principles to everything you do such as buying a new car, remodeling a bathroom, or writing a book.

DATA WAREHOUSING ARCHITECTURES AND SKILL SETS

To ensure that we are working from a common understanding, here is a very brief summary of data warehouse architectures and requisite skill sets. This is not a substitute for any of the more comprehensive technical books on data warehousing but should be sufficient as a baseline for the remainder of the book.

Data Warehousing Conceptual Architectures

Figure 1.1 depicts an abstracted classical data warehousing architecture and is suitable to convey either a Kimball-style (Kimball and Ross 2002) or an Inmon-style (Inmon 2005) architecture. This is a high-level conceptual architecture containing multiple layers, each of which includes a complex integration of commercial technologies, data modeling and manipulation, and some custom code.

The data warehouse architecture includes one or more *operational source systems* from which data is extracted, transformed, and loaded into the data

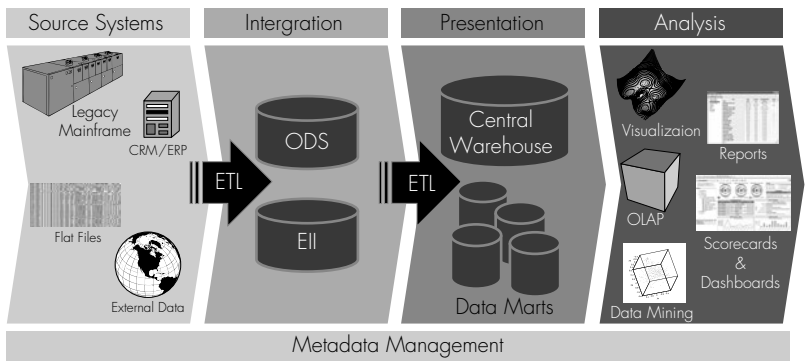

Figure 1.1 Classical data warehouse architecture

warehouse repositories. These systems are optimized for the daily transactional processing required to run the business operations. Most DW/BI systems source data from multiple operational systems, some of which are legacy systems that may be several decades old and reside on older technologies.

Data from these sources is extracted into an *integration tier* in the architecture that acts as a "holding pen" where data can be merged, manipulated, transformed, cleansed, and validated without placing an undue burden on the operational systems. This tier may include an operational data store or an enterprise information integration (EII) repository that acts as a system of record for all relevant operational data. The integration database is typically based on a relational data model and may have multiple subcomponents, including pre-staging, staging, and an integration repository, each serving a different purpose relating to the consolidation and preprocessing of data from disparate source systems. Common technologies for staging databases are Oracle, IBM DB2, Microsoft SQL Server, and NCR Teradata. The DW/BI community is beginning to see increasing use of the open-source database MySQL for this architectural component.

Data is extracted from the staging database, transformed, and loaded into a *presentation tier* in the architecture that contains appropriate structures for optimized multidimensional and analytical queries. This system is designed to support the data slicing and dicing that define the power of a data warehouse. There are a variety of alternatives for the implementation of the presentation database, including normalized relational schemas and denormalized schemas like star, snowflake, and even "starflake." Moreover, the presentation tier may include a single enterprise data warehouse or a collection of subject-specific data marts. Some architectures include a hybrid of both of these. Presentation repositories are typically implemented in the same technologies as the integration database.

Finally, data is presented to the business users at the *analysis tier* in the architecture. This conceptual layer in the system represents the variety of applications and tools that provide users with access to the data, including report writers, ad hoc querying, online analytical processing (OLAP), data visualization, data mining, and statistical analysis. BI tool vendors such as Pentaho, Cognos, MicroStrategy, Business Objects, Microsoft, Oracle, IBM, and others produce commercial products that enable data from the presentation database to be aggregated and presented within user applications.

This is a generalized architecture, and actual implementations vary in the details. One major variation on the Kimball architecture is the Inmon architecture (Inmon 2005), which inserts a layer of *subject-specific data marts* that contain subsets of the data from the main warehouse. Each data mart supports only the specific end-user applications that are relevant to the business subject area for which that mart was designed. Regardless of your preferences for Kimball- versus Inmon-style architectures, and of the variations found in implementation detail, Figure 1.1 will serve as reference architecture for the discussions in this book. The Agile DW/BI principles and practices that are introduced here are not specific to any particular architecture.

Diverse and Disparate Technical Skills

Inherent in the implementation of this architecture are the following aspects of development, each requiring a unique set of development skills:

- **Data modeling.** Design and implementation of data models are required for both the integration and presentation repositories. Relational data models are distinctly different from dimensional data models, and each has unique properties. Moreover, relational data modelers may not have dimensional modeling expertise and vice versa.
- **ETL development.** ETL refers to the *extraction* of data from source systems into staging, the *transformations* necessary to recast source data for analysis, and the *loading* of transformed data into the presentation repository. ETL includes the selection criteria to extract data from source systems, performing any necessary data transformations or derivations needed, data quality audits, and cleansing.
- **Data cleansing.** Source data is typically not perfect. Furthermore, merging data from multiple sources can inject new data quality issues. Data hygiene is an important aspect of data warehouse that requires specific skills and techniques.
- **OLAP design.** Typically data warehouses support some variety of online analytical processing (HOLAP, MOLAP, or ROLAP). Each OLAP technique is different but requires special design skills to balance the reporting requirements against performance constraints.
- **Application development.** Users commonly require an application interface into the data warehouse that provides an easy-to-use front end combined with comprehensive analytical capabilities, and one that is tailored to the way the users work. This often requires

some degree of custom programming or commercial application customization.

- **Production automation.** Data warehouses are generally designed for periodic automated updates when new and modified data is slurped into the warehouse so that users can view the most recent data available. These automated update processes must have built-in fail-over strategies and must ensure data consistency and correctness.
- **General systems and database administration.** Data warehouse developers must have many of the same skills held by the typical network administrator and database administrator. They must understand the implications of efficiently moving possibly large volumes of data across the network, and the issues of effectively storing changing data.

WHY DO WE NEED AGILE ANALYTICS?

In my years as a DW/BI consultant and practitioner I have learned three consistent truths: Building successful DW/BI systems is hard; DW/BI development projects fail very often; and it is better to fail fast and adapt than to fail late after the budget is spent.

First Truth: Building DW/BI Systems Is Hard

If you have taken part in a data warehousing project, you are aware of the numerous challenges, perils, and pitfalls. Ralph Kimball, Bill Inmon, and other DW/BI pioneers have done an excellent job of developing reusable architectural patterns for data warehouse and DW/BI implementation. Software vendors have done a good job of creating tools and technologies to support the concepts. Nonetheless, DW/BI is just plain hard, and for several reasons:

- **Lack of expertise.** Most organizations have not previously built a DW/BI system or have only limited experience in doing so.
- **Lack of experience.** Most organizations don't build multiple DW/BI systems, and therefore development processes don't get a chance to mature through experience.
- **Ambitious goals.** Organizations often set out to build an enterprise data warehouse, or at least a broad-reaching data mart, which makes the process more complex.
- **Domain knowledge versus subject matter expertise.** DW/BI practitioners often have extensive expertise in business intelligence but not in the organization's business domain, causing gaps in

understanding. Business users typically don't know what they can, or should, expect from a DW/BI system.

- **Unrealistic expectations.** Business users often think of data warehousing as a technology-based plug-and-play application that will quickly provide them with miraculous insights.
- **Educated user phenomenon.** As users gain a better understanding of data warehousing, their needs and wishes change.
- **Shooting the messenger.** DW/BI systems are like shining a bright light in the attic: You may not always like what you find. When the system exposes data quality problems, business users tend to distrust the DW/BI system.
- **Focus on technology.** Organizations often view a DW/BI system as an IT application rather than a joint venture between business stakeholders and IT developers.
- **Specialized skills.** Data warehousing requires an entirely different skill set from that of typical database administrators (DBAs) and developers. Most organizations do not have staff members with adequate expertise in these areas.
- **Multiple skills.** Data warehousing requires a multitude of unique and distinct skills such as multidimensional modeling, data cleansing, ETL development, OLAP design, application development, and so forth.

These unique DW/BI development characteristics compound the already complex process of building software or building database applications.

Second Truth: DW/BI Development Projects Fail Often

Unfortunately, I'm not the only one who has experienced failure on DW/BI projects. A quick Google search on "data warehouse failure polls" results in a small library of case studies, postmortems, and assessment articles. Estimated failure rates of around 50 percent are common and are rarely disputed.

When I speak to groups of business intelligence practitioners, I often begin my talks with an informal survey. First I ask everyone who has been involved in the completion of one or more DW/BI projects to stand. It varies depending on the audience, but usually more than half the group stands up. Then I ask participants to sit down if they have experienced projects that were delivered late, projects that had significant budget overruns, or projects that did not satisfy users' expectations. Typically nobody is left standing by the third question, and I haven't even gotten to questions about

acceptable quality or any other issues. It is apparent that most experienced DW/BI practitioners have lived through at least one project failure.

While there is no clear definition of what constitutes "failure," Sid Adelman and Larissa Moss classify the following situations as characteristic of limited acceptance or outright project failure (Moss and Adelman 2000):

- The project is over budget.
- The schedule has slipped.
- Some expected functionality was not implemented.
- Users are unhappy.
- Performance is unacceptable.
- Availability of the warehouse applications is poor.
- There is no ability to expand.
- The data and/or reports are poor.
- The project is not cost-justified.
- Management does not recognize the benefits of the project.

In other words, simply completing the technical implementation of a data warehouse doesn't constitute success. Take another look at this list. Nearly every situation is "customer"-focused; that is, primarily end users determine whether a project is successful.

There are literally hundreds of similar evaluations of project failures, and they exhibit a great deal of overlap in terms of root causes: incorrect requirements, weak processes, inability to adapt to changes, project scope mismanagement, unrealistic schedules, inflated expectations, and so forth.

Third Truth: It Is Best to Fail Fast and Adapt

Unfortunately, the traditional development model does little to uncover these deficiencies early in the project. As Jeff DeLuca, one of the creators of Feature Driven Development (FDD), says, "We should try to break the back of the project as early as possible to avoid the high cost of change later downstream." In a traditional approach, it is possible for developers to plow ahead in the blind confidence that they are building the right product, only to discover at the end of the project that they were sadly mistaken. This is true even when one uses all the best practices, processes, and methodologies.

What is needed is an approach that promotes early discovery of project peril. Such an approach must place the responsibility of success equally on the users, stakeholders, and developers and should reward a team's ability to adapt to new directions and substantial requirements changes.

As we observed earlier, most classes of project failure are user-satisfaction-oriented. If we can continuously adapt the DW/BI system and align with user expectations, users will be satisfied with the outcome. In all of my past involvement in traditional DW/BI implementations I have consistently seen the following phenomena at the end of the project:

- **Users have become more educated about BI.** As the project progresses, so does users' understanding of BI. So, what they told you at the beginning of the project may have been based on a misunderstanding or incorrect expectations.
- **User requirements have changed or become more refined.** That's true of *all* software and implementation projects. It's just a fact of life. What they told you at the beginning is much less relevant than what they tell you at the end.
- **Users' memories of early requirements reviews are fuzzy.** It often happens that contractually speaking, a requirement is met by the production system, but users are less than thrilled, having reactions like "What I really meant was . . ." or "That may be what I said, but it's not what I want."
- **Users have high expectations when anticipating a new and useful tool.** Left to their own imaginations, users often elevate their expectations of the BI system well beyond what is realistic or reasonable. This only leaves them disappointed when they see the actual product.
- **Developers build based on the initial snapshot of user requirements.** In waterfall-style development the initial requirements are reviewed and approved, then act as the scoping contract. Meeting the terms of the contract is not nearly as satisfying as meeting the users' expectations.

All these factors lead to a natural gap between what is built and what is needed. An approach that frequently releases new BI features to users, hears user feedback, and adapts to change is the single best way to fail fast and correct the course of development.

Is Agile Really Better?

There is increasing evidence that Agile approaches lead to higher project success rates. Scott Ambler, a leader in Agile database development and Agile Modeling, has conducted numerous surveys on Agile development in an effort to quantify the impact and effectiveness of these methods. Beginning in 2007, Ambler conducted three surveys specifically relating to IT

project success rates.[6] The 2007 survey explored success rates of different IT project types and methods. Only 63 percent of traditional projects and data warehousing projects were successful, while Agile projects experienced a 72 percent rate of success. The 2008 survey focused on four success criteria: quality, ROI, functionality, and schedule. In all four areas Agile methods significantly outperformed traditional, sequential development approaches. The 2010 survey continued to show that Agile methods in IT produce better results.

I should note here that traditional definitions of success involve metrics such as on time, on budget, and to specification. While these metrics may satisfy management efforts to control budgets, they do not always correlate to customer satisfaction. In fact, scope, schedule, and cost are poor measures of progress and success. Martin Fowler argues, "Project success is more about whether the software delivers value that's greater than the cost of the resources put into it." He points out that XP 2002 conference speaker Jim Johnson, chairman of the Standish Group, observed that a large proportion of features are frequently unused in software products. He quoted two studies: a DuPont study, which found that only 25 percent of a system's features were really needed, and a Standish study, which found that 45 percent of features were never used and only 20 percent of features were used often or always (Fowler 2002). These findings are further supported by a Department of Defense study, which found that only 2 percent of the code in $35.7 billion worth of software was used as delivered, and 75 percent was either never used or was canceled prior to delivery (Leishman and Cook 2002).

Agile development is principally aimed at the delivery of high-priority value to the customer community. Measures of progress and success must focus more on value delivery than on traditional metrics of on schedule, on budget, and to spec. Jim Highsmith points out, "Traditional managers expect projects to be on-track early and off-track later; Agile managers expect projects to be off-track early and on-track later." This statement reflects the notion that incrementally evolving a system by frequently seeking and adapting to customer feedback will result in building the right solution, but it may not be the solution that was originally planned.

The Difficulties of Agile Analytics

Applying Agile methods to DW/BI is not without challenges. Many of the project management and technical practices I introduce in this book are

6. The detailed results are available at www.ambysoft.com/surveys/.

adapted from those of our software development colleagues who have been maturing these practices for the past decade or longer. Unfortunately, the specific practices and tools used to custom-build software in languages like Java, C++, or C# do not always transfer easily to systems integration using proprietary technologies like Informatica, Oracle, Cognos, and others. Among the problems that make Agile difficult to apply to DW/BI development are the following:

- **Tool support.** There aren't many tools that support technical practices such as test-driven database or ETL development, database refactoring, data warehouse build automation, and others that are introduced in this book. The tools that do exist are less mature than the ones used for software development. However, this current state of tool support continues to get better, through both open-source as well as commercial tools.
- **Data volume.** It takes creative thinking to use lightweight development practices to build high-volume data warehouses and BI systems. We need to use small, representative data samples to quickly build and test our work, while continuously proving that our designs will work with production data volumes. This is more of an *impediment* to our way of approaching the problem rather than a *barrier* that is inherent in the problem domain. Impediments are those challenges that can be eliminated or worked around; barriers are insurmountable.
- **"Heavy lifting."** While Agile Analytics is a feature-driven (think business intelligence features) approach, the most time-consuming aspect of building DW/BI systems is in the back-end data warehouse or data marts. Early in the project it may seem as if it takes a lot of "heavy lifting" on the back end just to expose a relatively basic BI feature on the front end. Like the data volume challenge, it takes creative thinking to build the smallest/simplest back-end data solution needed to produce business value on the front end.
- **Continuous deployment.** The ability to deploy new features into production frequently is a goal of Agile development. This goal is hampered by DW/BI systems that are already in production with large data volumes. Sometimes updating a production data warehouse with a simple data model revision can require significant time and careful execution. Frequent deployment may look very different in DW/BI from the way it looks in software development.

The nuances of your project environment may introduce other such difficulties. In general, those who successfully embrace Agile's core values and

guiding principles learn how to effectively adapt their processes to mitigate these difficulties. For each of these challenges I find it useful to ask the question "Will the project be better off if we can overcome this difficulty despite how hard it may be to overcome?" As long as the answer to that question is yes, it is worth grappling with the challenges in order to make Agile Analytics work. With time and experience these difficulties become easier to overcome.

INTRODUCING FLIXBUSTER ANALYTICS

Now seems like a good time to introduce the running DW/BI example that I'll be revisiting throughout this book to show you how the various Agile practices are applied. I use an imaginary video rental chain to demonstrate the Agile Analytics practices. The company is FlixBuster, and they have retail stores in cities throughout North America. FlixBuster also offers video rentals online where customers can manage their rental requests and movies are shipped directly to their mailing address. Finally, FlixBuster offers movie downloads directly to customers' computers.

FlixBuster has customers who are members and customers who are nonmembers. Customers fall into three buying behavior groups: those who shop exclusively in retail stores, those who shop exclusively online, and those who split their activity across channels. FlixBuster customers can order a rental online or in the store, and they can return videos in the store or via a postage-paid return envelope provided by the company.

Members pay a monthly subscription fee, which determines their rental privileges. Top-tier members may rent up to three videos at the same time. There is also a membership tier allowing two videos at a time as well as a tier allowing one at a time. Members may keep their rentals indefinitely with no late charges. As soon as FlixBuster receives a returned video from a member, the next one is shipped. Nonmembers may also rent videos in the stores following the traditional video rental model with a four-day return policy.

Approximately 75 percent of the brick-and-mortar FlixBuster stores across North America are corporately owned and managed; the remaining 25 percent are privately owned franchises. FlixBuster works closely with franchise owners to ensure that the customer experience is consistent across all stores. FlixBuster prides itself on its large inventory of titles, the rate of customer requests that are successfully fulfilled, and how quickly members receive each new video by mail.

FlixBuster has a complex partnership with the studios producing the films and the clearinghouses that provide licensed media to FlixBuster and manage royalty payments and license agreements. Each title is associated with a royalty percentage to be paid to the studio. Royalty statements and payments are made on a monthly basis to each of the clearinghouses.

Furthermore, FlixBuster sales channels (e-tail and retail) receive a percentage of the video rental revenue. Franchise owners receive a negotiated revenue amount that is generally higher than for corporately owned retail outlets. The online channel receives still a different revenue percentage to cover its operating costs.

FlixBuster has determined that there is a good business case for developing an enterprise business intelligence system. This DW/BI system will serve corporate users from finance, marketing, channel sales, customer management, inventory management, and other departments. FlixBuster also intends to launch an intranet BI portal for subscription use by its clearinghouse partners, studios, franchisees, and possibly even Internet movie database providers. Such an intranet portal is expected to provide additional revenue streams for FlixBuster.

There are multiple data sources for the FlixBuster DW/BI system, including FlixBackOffice, the corporate ERP system; FlixOps, the video-by-mail fulfillment system; FlixTrans, the transactional and point-of-sale system; FlixClear, the royalty management system; and others.

FlixBuster has successfully completed other development projects using Agile methods and is determined to take an Agile Analytics approach on the development of its DW/BI system, FlixAnalysis. During high-level executive steering committee analysis and reviews, it has been decided that the first production release of FlixAnalysis will be for the finance department and will be a timeboxed release cycle of six months.

WRAP-UP

This chapter has laid the foundation for an accurate, if high-level, understanding of Agile Analytics. Successive chapters in this book serve to fill in the detailed "how-to" techniques that an Agile Analytics team needs to put these concepts into practice. You should now understand that Agile Analytics isn't simply a matter of chunking tasks into two-week iterations, holding a 15-minute daily team meeting, or retitling the project manager a "scrum master." Although these may be Agile traits, new Agile teams often

limit their agility to these simpler concepts and lose sight of the things that truly define agility. True agility is reflected by traits like early and frequent delivery of production-quality, working BI features, delivering the highest-valued features first, tackling risk and uncertainty early, and continuous stakeholder and developer interaction and collaboration.

Agile Analytics teams evolve toward the best system design by continuously seeking and adapting to feedback from the business community. Agile Analytics balances the right amount of structure and formality against a sufficient amount of flexibility, with a constant focus on building the right solution. The key to agility lies in the core values and guiding principles more than in a set of specific techniques and practices—although effective techniques and practices are important. Mature Agile Analytics teams elevate themselves above a catalog of practices and establish attitudes and patterns of behavior that encourage seeking feedback, adapting to change, and delivering maximum value.

If you are considering adopting Agile Analytics, keep these core values and guiding principles at the top of your mind. When learning any new technique, it is natural to look for successful patterns that can be mimicked. This is a valuable approach that will enable a new Agile team to get on the right track and avoid unnecessary pitfalls. While I have stressed that Agile development is not a prescriptive process, new Agile teams will benefit from some recipe-style techniques. Therefore, many of the practices introduced in this book may have a bit of a prescriptive feel. I encourage you to try these practices first as prescribed and then, as you gain experience, tailor them as needed to be more effective. But be sure you're tailoring practices for the right reasons. Be careful not to tailor a practice simply because it was difficult or uncomfortable on the first try. Also, be sure not to simply cherry-pick the easy practices while ignoring the harder ones. Often the harder practices are the ones that will have the biggest impact on your team's performance.

INDEX

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

Q

R

S

T

U

V

W

X

Y

Z