Preface

Early in my career, I saw a full-page advertisement in a magazine that showed one keyboard key, similar to the Enter key, labeled with the word "Integrate" (see Figure P-1). The text below the key read, "If only it were this easy." I am not sure who or what this ad was for, but it struck a chord with me. In considering software development, I thought, surely that would *never* be achievable because, on my project, we spent several days in "integration hell" attempting to cobble together the myriad software components at the end of most project milestones. But I liked the concept, so I cut out the ad and hung it on my wall. To me, it represented one of my chief goals in being an efficient software developer: to automate repetitive and error-prone processes. Furthermore, it embodied my belief in making software integration a "nonevent" (as Martin Fowler has called this) on a project—something that just happens as a matter of course. Continuous Integration (CI) can help make integration a nonevent on your project.

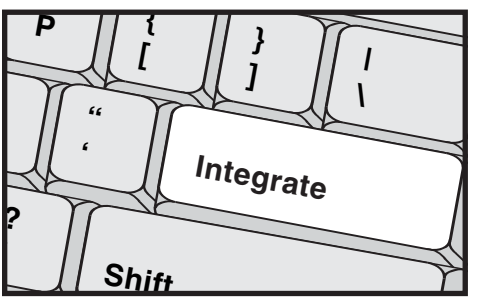

FIGURE P-1 Integrate!

What Is This Book About?

Consider some of the more typical development processes on a software project: Code is compiled, and data is defined and manipulated via a database; testing occurs, code is reviewed, and ultimately, software is deployed. In addition, teams almost certainly need to communicate with one another regarding the status of the software. Imagine if you could perform these processes at the press of a single button.

This book demonstrates how to create a virtual Integrate button to automate many software development processes. What's more, we describe how this Integrate button can be pressed continuously to reduce the risks that prevent you from creating deployable applications, such as the late discovery of defects and low-quality code. In creating a CI system, many of these processes are automated, and they run every time the software under development is changed.

What Is Continuous Integration?

The process of integrating software is not a new problem. Software integration may not be as much of an issue on a one-person project with few external system dependencies, but as the complexity of a project increases (even just adding one more person), there is a greater need to integrate and ensure that software components work together—early *and often*. Waiting until the end of a project to integrate leads to all sorts of software quality problems, which are costly and often lead to project delays. CI addresses these risks faster and in smaller increments.

In his popular "Continuous Integration" article,[1] Martin Fowler describes CI as:

> . . . a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily—leading to multiple integrations per day. Each integration is

1. See www.martinfowler.com/articles/continuousIntegration.html.

verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

In my experience, this means that:

- All developers run private builds[2] on their own workstations before committing their code to the version control repository to ensure that their changes don't break the integration build.
- Developers commit their code to a version control repository *at least* once a day.
- Integration builds occur several times a day on a separate build machine.
- 100% of tests must pass for every build.
- A product is generated (e.g., WAR, assembly, executable, etc.) that can be functionally tested.
- Fixing broken builds is of the highest priority.
- Some developers review reports generated by the build, such as coding standards and dependency analysis reports, to seek areas for improvement.

This book discusses the automated aspects of CI because of the many benefits you receive from automating repetitive and error-prone processes; however, as Fowler identifies, CI is the process of integrating work frequently—and this need not be an automated process to qualify. We clearly believe that since there are many great tools that support CI as an automated process, using a CI server to automate your CI practices is an effective approach. Nevertheless, a manual approach to integration (using an automated build) may work well with your team.

2. The Private (System) Build and Integration Build patterns are covered in *Software Configuration Management Patterns* by Stephen P. Berczuk and Brad Appleton.

Rapid Feedback

Continuous Integration increases your opportunities for feedback. Through it, you learn the state of the project several times a day. CI can be used to reduce the time between when a defect is introduced and when it is fixed, thus improving overall software quality.

A development team should not believe that because their CI system is automated, they are safe from integration problems. It is even less true if the group is using an automated tool for nothing more than compiling source code; some refer to this as a "build," which it is not (see Chapter 1). The effective practice of CI involves much more than a tool. It includes the practices we outline in the book, such as frequent commits to a version control repository, fixing broken builds immediately, and using a separate integration build machine.

The practice of CI enables faster feedback. When using effective CI practices, you'll know the overall health of software under development *several times a day.* What's more, CI works well with practices like refactoring and test-driven development, because these practices are centered on the notion of making small changes. CI, in essence, provides a safety net to ensure that changes work with the rest of the software. At a higher level, CI increases the collective confidence of teams and lessens the amount of human activity needed on projects, because it's often a *hands-off* process that runs whenever your software changes.

A Note on the Word "Continuous"

We use the term "continuous" in this book, but the usage is technically incorrect. "Continuous" implies that something kicks off once and never stops. This suggests that the process is constantly integrating, which is not the case in even the most intense CI environment. So, what we are describing in this book is more like "continual integration."

Who Should Read This Book?

In our experience, there is a distinct difference between someone who treats software development as a *job* and someone who treats it as a *profession.* This book is for those who work at their profession and find themselves performing repetitive processes on a project (or we will help you realize just how often you are doing so). We describe the practices and benefits of CI and give you the knowledge to apply these practices so that you can direct your time and expertise to more important, challenging issues.

This book covers the major topics relating to CI, including how to implement CI using continuous feedback, testing, deployment, inspection, and database integration. No matter what your role in software development, you can incorporate CI into your own software development processes. If you are a software professional who wants to become increasingly effective—getting more done with your time and with more dependable results—you will gain much from this book.

Developers

If you have noticed that you'd rather be developing software for users than fiddling with software integration issues, this book will help you get there without much of the "pain" you thought would be involved. This book doesn't ask you to spend more time integrating; it's about making much of software integration a nonevent, leaving you to focus on doing what you love the most: developing software. The many practices and examples in this book demonstrate how to implement an effective CI system.

Build/Configuration/Release Management

If your job is to get *working* software out the door, you'll find this book particularly interesting as we demonstrate that by running processes *every time* a change is applied to a version control repository, you can generate cohesive, working software. Many of you are

managing builds while filling other roles on your project, such as development. CI will do some of the "thinking" for you, and instead of waiting until the end of the development lifecycle, it creates deployable and *testable* software several times a day.

Testers

CI offers a rapid feedback approach to software development, all but eliminating the traditional pain of reoccurring defects even after "fixes" were applied. Testers usually gain increased satisfaction and interest in their roles on a project using CI, since software to test is available more often and with smaller scopes. With a CI system in your development lifecycle, you test *all along the way,* rather than the typical feast or famine scenario where testers are either testing into the late hours or not testing at all.

Managers

This book can have great impact for you if you seek a higher level of confidence in your team's capability to consistently and repeatedly deliver working software. You can manage scopes of time, cost, and quality much more effectively because you are basing your decisions on working software with actual feedback and metrics, not just task items on a project schedule.

Organization of This Book

This book is divided into two parts. Part I is an introduction to CI and examines the concept and its practices from the ground up. Part I is geared toward those readers not familiar with the core practices of CI. We do not feel the practice of CI is complete, however, without a Part II that naturally expands the core concepts into other effective processes performed by CI systems, such as testing, inspection, deployment, and feedback.

Part I: A Background on CI—Principles and Practices

Chapter 1, Getting Started, gets you right into things with a high-level example of using a CI server to continuously build your software.

Chapter 2, Introducing Continuous Integration, familiarizes you with the common practices and how we got to CI.

Chapter 3, Reducing Risks Using CI, identifies the key risks CI can mitigate using scenario-based examples.

Chapter 4, Building Software at Every Change, explores the practice of integrating your software for every change by leveraging the automated build.

Part II: Creating a Full-Featured CI System

Chapter 5, Continuous Database Integration, moves into more advanced concepts involving the process of rebuilding databases and applying test data as part of every integration build.

Chapter 6, Continuous Testing, covers the concepts and strategies of testing software with every integration build.

Chapter 7, Continuous Inspection, takes you through some automated and continuous inspections (static and dynamic analysis) using different tools and techniques.

Chapter 8, Continuous Deployment, explores the process of deploying software using a CI system so that it can be functionally tested.

Chapter 9, Continuous Feedback, describes and demonstrates the use of continuous feedback devices (such as e-mail, RSS, X10, and the Ambient Orb) so that you are notified on build success or failure as it happens.

The Epilogue explores the future possibilities of CI.

Appendixes

Appendix A, CI Resources, includes a list of URLs, tools, and papers related to CI.

Appendix B, Evaluating CI Tools, assesses the different CI servers and related tools on the market, discusses their applicability to the practices described in the book, identifies the advantages and disadvantages of each, and explains how to use some of their more interesting features.

Other Features

The book includes features that help you to better learn and apply what we describe in the text.

- Practices—We cover more than forty CI-related practices in this book. Many chapter subheadings are practices. A figure at the beginning of most chapters illustrates the practices covered and lets you scan for areas that interest you. for example, *use a dedicated integration build machine* and *commit code frequently* are both examples of practices discussed in this book.
- Examples—We demonstrate how to apply these practices by using various examples in different languages and platforms.
- Questions—Each chapter concludes with a list of questions to help you evaluate the application of CI practices on your project.
- Web site—The book's companion Web site, www.integratebutton.com, provides book updates, code examples, and other material.

What You Will Learn

By reading this book, you will learn concepts and practices that enable you to create cohesive, working software many times a day. We have taken care to focus on the practices first, followed by the application of these practices, with examples included as demonstration wherever possible. The examples use different development platforms, such as Java, Microsoft .NET, and even some Ruby. CruiseControl (Java and .NET versions) is the primary CI server used throughout the book; however, we have created similar examples using other servers and tools on the companion Web site (www.integratebutton.com) and in Appendix B.

As you work your way through the book, you gain these insights:

- How implementing CI produces *deployable software* at every step in your development lifecycle.
- How CI can *reduce the time* between when a defect is introduced and when that defect is detected, thereby lowering the cost to fix it.
- How you can *build quality into your software* by building software often rather than waiting to the latter stages of development.

What This Book Does Not Cover

This book does not cover every tool—build scheduling, programming environment, version control, and so on—that makes up your CI system. It focuses on the implementation of CI practices to develop an effective CI system. CI practices are discussed first; if a particular tool demonstrated is no longer in use or doesn't meet your particular needs, simply apply the practice using another tool to achieve the same effect.

It is also not possible, or useful, to cover every type of test, feedback mechanism, automated inspector, and type of deployment used by a CI system. We hope that a greater goal is met by focusing on the range of key practices, using examples of techniques and tools for database integration, testing, inspection, and feedback that may inspire applications as different as the projects and teams that learn about them. As mentioned throughout the book, the book's companion Web site, www.integratebutton.com, contains examples using other tools and languages that may not be covered in the book.

Authorship

This book has three coauthors and one contributor. I wrote most of the chapters. Steve Matyas contributed to Chapters 4, 5, 7, 8, and Appendix A, and constructed some of the book's examples. Andy Glover wrote Chapters 6, 7, and 8, provided examples, and made contributions elsewhere in the book. Eric Tavela wrote Appendix B. So when sentences use first-person pronouns, this should provide clarity as to who is saying what.

About the Cover

I was excited when I learned that our book was to be a part of the renowned Martin Fowler Signature Series. I knew this meant that I would get to choose a bridge for the cover of the book. My coauthors and I are part of a rare breed who grew up in the Washington, D.C.,

area. For those of you not from the region, it's a very transient area. More specifically, we are from Northern Virginia and figured it would be a fitting tribute to choose the Natural Bridge in Virginia for the cover. I had never visited the bridge until early 2007—after I had chosen it for the book cover. It has a very interesting history and I found it incredible that it's a functioning bridge that automobiles travel on every day. (Of course, I had to drive my car over it a couple of times.) I'd like to think that after reading this book, you will make CI a natural part of your next software development project.

Acknowledgments

I can't tell you how many times I've read acknowledgments in a book and authors wrote how they "couldn't have done it by (themselves)" and other such things. I always thought to myself, "They're just being falsely modest." Well, I was dead wrong. This book was a massive undertaking to which I am grateful to the people listed herein.

I'd like to thank my publisher, Addison-Wesley. In particular, I'd like to express my appreciation to my executive editor, Chris Guzikowski, for working with me during this exhaustive process. His experience, insight, and encouragement were tremendous. Furthermore, my development editor, Chris Zahn, provided solid recommendations throughout multiple versions and editing cycles. I'd also like to thank Karen Gettman, Michelle Housley, Jessica D'Amico, Julie Nahil, Rebecca Greenberg, and last but definitely not least, my first executive editor, Mary O'Brien.

Rich Mills hosted the CVS server for the book and offered excellent ideas during brainstorming sessions. I'd also like to thank my mentor and friend, Rob Daly, for getting me into professional writing in 2002 and for providing exceptionally detailed reviews throughout the writing process. John Steven was instrumental in helping me start this book's writing process.

I'd like to express my gratitude to my coauthors, editor, and contributing author. Steve Matyas and I endured many sleepless nights to create what you are reading today. Andy Glover was our clutch writer, providing his considerable developer testing experience to the project.

Lisa Porter, our contributing editor, tirelessly combed through every major revision to provide edits and recommendations which helped increase the quality of the book. A thank you to Eric Tavela, who wrote the CI tools appendix, and to Levent Gurses for providing his experiences with Maven 2 in Appendix B.

We had an eclectic cadre of personal technical reviewers who provided excellent feedback throughout this project. They include Tom Copeland, Rob Daly, Sally Duvall, Casper Hornstrup, Joe Hunt, Erin Jackson, Joe Konior, Rich Mills, Leslie Power, David Sisk, Carl Tallis, Eric Tavela, Dan Taylor, and Sajit Vasudevan.

I'd also like to thank Charles Murray and Cristalle Belonia for their assistance, and Maciej Zawadzki and Eric Minick from Urbancode for their help.

I am grateful for the support of many great people who inspire me every day at Stelligent, including Burke Cox, Mandy Owens, David Wood, and Ron Wright. There are many others who have inspired my work over the years, including Rich Campbell, David Fado, Mike Fraser, Brent Gendleman, Jon Hughes, Jeff Hwang, Sherry Hwang, Sandi Kyle, Brian Lyons, Susan Mason, Brian Messer, Sandy Miller, John Newman, Marcus Owen, Chris Painter, Paulette Rogers, Mark Simonik, Joe Stusnick, and Mike Trail.

I also appreciate the thorough feedback from the Addison-Wesley technical review team, including Scott Ambler, Brad Appleton, Jon Eaves, Martin Fowler, Paul Holser, Paul Julius, Kirk Knoernschild, Mike Melia, Julian Simpson, Andy Trigg, Bas Vodde, Michael Ward, and Jason Yip.

I want to thank the attendees of CITCON Chicago 2006 for sharing their experiences on CI and testing with all of us. In particular, I'd like to acknowledge Paul Julius and Jeffrey Frederick for organizing the conference, and everyone else who attended the event.

Finally, I'd like to thank Jenn for her unrelenting support and for being there through the ups and downs of making this book.

Paul M. Duvall
Fairfax, Virginia
March 2007