Foreword

If you know of Fit as a testing tool, you don't need me to convince you that this book is important. You know that Fit is a tool used to write what I call "business-facing" tests, tests that help everyone believe that the software really will deliver business value. You may also know that Fit has been used mostly by early-adopter projects, the ones that are always pushing the envelope of their abilities, but that it's poised for mainstream acceptance.

And what is one of the things a tool needs to move into the mainstream? Authoritative texts, so that new users can quickly benefit from everything the scattered early adopters have learned through trial and error. That's what you're holding: a book by Fit's creator and one of the world's foremost Fitsters.

But Fit is more than a testing tool. Used to its fullest extent, it's a tool for *surprise.* Let me circle around to what I mean by that.

> To create a supple, knowledge-rich design calls for a versatile, shared team language, and a lively experimentation with language that seldom happens on software projects.
> —Eric Evans, *Domain-Driven Design* [Eva04, p. 24]

A software project is a place where different cultures come together. Some people face toward the business and its concerns; other people face toward the computer and its demands. To an expert in financial trading, a "bond" is something that's tangled up in all sorts of explicit and implicit legal, social, historical, and emotional meanings. To programmers, a Bond is an object in their program that they're trying to *keep* from getting too tangled up with other objects, lest their brains explode. Somehow, these people have to work together, and they do it by creating a shared language.

Most of that creation happens through the conversation that threads through the whole project. But some of it happens through writing.

> Ever since I can remember I have belonged to the zoological species of those for whom thought takes shape as I write.
> —Louis Aragon

Fit is a tool for people to sit down and collaboratively create tests. Tests are special things. Spoken conversation is free to be abstract, and it often is, but tests

are relentlessly concrete. It's a struggle to express so precisely what you want. And it's in such struggle that creative surprise happens. You push against the world. The world pushes back. You shift your position—maybe you're not using the right words, maybe you could break the problem down differently—and push again. Repeat, maybe until it seems you'll never get it right. And then, suddenly, you push and the resistance is gone. Eureka! You've figured out how to say something that the businesspeople, the programmers, and the computer can all work with.

> By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems.... Civilization advances by extending the number of important operations which we can perform without thinking about them.
> —A. N. Whitehead, *An Introduction to Mathematics*

It's not just through concreteness that Fit helps. When we think of what we want the computer to do, we tend to think linearly: "First I do A, so the computer will do B. Then I'll do C, so the computer will do D." It's natural to write tests the same way. But Fit uses a tabular format that encourages you to play around with both horizontal and vertical space as you look for a better way to lay out what the program is to do. When you find it, the details will drop away, handled behind the scenes, and you'll be surprised by what they obscured: overlooked special cases, conflicting needs, regularities that you can exploit, and the like.

A really top-notch software project is a strange beast. It's both relentlessly focused on delivering business value *and* ever prepared for that moment when what you know falls into a new configuration, and great new paths forward open up. For the latter to happen, you need tools, practices, habits, and patterns of communication that keep the funding flowing while gently jiggling what you know. Fit is such a tool, and this is the book that will open it up to you.

Read on.

Brian Marick
February 2005