# Demystifying Agile Development

**Stephen Mellor**

**Agile Software Development Ecosystems** *by Jim Highsmith, Addison-Wesley, 2002, ISBN 0-201-76043-6, 448 pp., US$44.99.*

In some measure, the Extreme movement, and its slightly less out-there cousin, the Agile Alliance, are reactions to heavyweight, command-and-control, process-heavy methods. The thesis of planning, controlling, and predicting deliberate methods has been countered by a responsive, playful, and adaptive software development approach.

"We thrive on chaos!" say some. "Thinking is the enemy!" say others. The very name extreme is revolutionary in nature, as is the call-to-arms of Kent Beck's 40-hour week. Horrors!

The language is weird, too—for example *chaordic*, a made-up agglomeration of *chaos* and *order*, or *ecosystem* as applied to software development practices. Each invokes mystery and not a little fear. Whatever this agile stuff is, it's different, it's dangerous, and it has come to overthrow software development's established order.

Or so you might think. Jim Highsmith's book *Agile Software Development Ecosystems* admirably demystifies the entire affair and makes it approachable without body armor in an easy-to-read 400 or so pages.

Part I, "Problems and Solutions," describes the software development challenge and proposes a solution in the form of agile development processes. Part II, "Principles and People," features interviews with several agile processes proponents (including one with the author himself, conducted by Alistair Cockburn). These interviews are interspersed with chapters that describe the principles that animate the several luminaries. Part III, "Agile Software Development Ecosystems," abstracts up one layer to describe how methods respond to a team as it works on a project. Part IV describes how to develop an agile software development ecosystem for yourself in an appropriately recursive agile way.

The book's strength is that it is a chaordic ecosystem. One great difficulty in teaching a contrary concept is that the reader has an existing framework of knowledge that must be unlearned. Bald assertions such as "changing code is cheap," as Beck might say, have a certain adolescent shock value, but they can lead to the idea's rejection.

A milder, more effective approach is to chip away at the reader's assumptions until that reader comes out on the other end without knowing it. There's no direct, single, or simple path through this. Rather, you have to be gently immersed in the ideas, reading them from different angles, some of which stick. It's not predictable; it's chaotic. At the same time, it has an order to it all because the underlying ideas do conform to a framework that can be revealed only by a skillful writer at the reader's pace. Jim's writing is clear, respectful, and nonconfrontational. You can't help but be drawn in.

And what you're drawn into is an ecosystem. It's illuminating and fascinating to be faced with so many definitions of a software development process, some flatly contradictory. At the same time, the purpose—delivering value to the customer—links the luminaries together. This came through for me in Part IV, "Developing an Agile Software Development Ecosystem." Here, you can clearly see the interplay of the ideas, albeit at a high level of abstraction.

This book isn't a guide for a specific agile development method, but it does provide a view for what each one does. With that information, you can choose which guru to study more closely. Nor is it a developer's guide to a generalized agile process drawn from the author's experience. Rather, it's an intelligent effort to describe honestly how each approach works. Finally, this book is not a polemic. Highsmith recognizes the strengths of deliberate development processes, and he gives them fair play.

I approached *Agile Software Development Ecosystems* in a suspicious frame of mind—any software book with *ecosystem* in its title can't possibly be serious. I came away with a new appreciation for the idea and the ability to write *chaordic* without irony. Through its breadth, clarity, and honesty, this book might yet form the basis for a synthesis.

**Stephen Mellor** is chief scientist in the Embedded Systems Division of Mentor Graphics. Contact him at steve@projtech.com.

*Integration and Product Improvement* offers a comprehensive introduction to the CMMI, attempting to include all relevant model structuring and informative aspects while avoiding the overlaps of the different CMMI flavors.

## Distilled CMMI

*CMMI Distilled: A Practical Introduction to Integrated Process Improvement*—the title alone is appealing, especially if you know about the CMMI's complexity. The book explains the CMMI without getting lost in the forest of process areas and model types. It looks specifically at the integrated approach's benefits, with its expectations and many proposals for multidisciplinary process improvement. It also characterizes the impacts and effects of implementing the CMMI's process areas from a user perspective.

The book includes a good selection of best practices for implementing the CMMI. Specifically, I liked Chapter 7, which translates the process areas into concrete implementation guidelines. Es-