

foreword

It is the rare computer-science book that truly captivates me. Sometimes, it's the raw power of the writer's deep intellect and mastery, as with Guy Steele's *Common LISP: The Language*, which I remember reading straight through while lying on a sunny Hawaiian beach. Some would certainly chuckle at such "geekiness"—and maybe they are right to do so—but for me, each of Steele's successive chapters awoke a hunger to understand more and more. I couldn't put it down.

Then there's the seeming fairy tale packed with amazing revelation after revelation. A book that simultaneously forces you to suspend reality, yet hammers you in the cerebellum with the deep-but-fleeting truths you've been desperately seeking, truths you know you should have already recognized but have somehow missed. Tom DeMarco's *The Deadline: A Novel About Project Management* was such a book. I couldn't put it down.

Bruce Tate has, with *Bitter Java*, created another of these rare, captivating works. As with DeMarco's Morovian kidnapping, Bruce's personal "extreme sports" kayaking, mountain biking, and hot-air ballooning adventures carried me from "hydraulic" point to point, paddling as fast as I could to get to the next pattern or point. As with Steele, I couldn't wait for the next successive insight—and as with DeMarco, I just couldn't put *Bitter Java* down.

My advice? Don't start reading this book unless you can drop everything else on your schedule for the rest of the day. If it's late in the day, I feel for you, because you're going to be very tired tomorrow. If you're on a sunny Hawaiian beach, you'd better use SPF 99. There's no escaping it.

Bitter Java was a thrill for me, and I fully expect it will be for you too. If you develop software or work with those who do, you'll relate to chapter after chapter. I fully expect to be quoting Bruce in my next design review. *Bitter Java* is simply loaded with the wisdom and experience any good software engineer seeks. I found chapter 9's Java coding standards worthy of being its own publication—one that I sincerely wish all Java programmers would read and heed.

As Bruce's analogies clearly express, software engineering is very much like running dangerous rivers, and even though it's not necessarily as life-threatening as your standard class IV+ hydraulic, failure can be just as catastrophic to your livelihood and that of those you lead. So I recommend that you study this excellent guidebook very carefully. Bruce has packed it solid with the clearly written, fun-to-read, hard-earned wisdom of a true *white-water* master.

Prepare yourself well so you can maximize the thrill of the ride *and* live to do it again—and don't drop your paddle unless you've got your "hands Eskimo roll" down pat!

Have fun!

Hays W. "Skip" McCormick III
Coauthor of *AntiPatterns: Refactoring Software,
Architectures, and Projects in Crisis*