Foreword

It's not often that an author comes along in our field and opens up a whole new line of inquiry. That's what Diomidis Spinellis did with his first book, *Code Reading.* There is a desperate need in our field for books that broach the subject of reading code rather than writing it. There is a whole school of software thought that says it is more important to teach new students how to read code before they learn to write it, on the grounds that (a) reading-before-writing is the way other language subjects conduct their pedagogy, and (b) the task of most programmers in our new millennium is to modify existing code (which means reading it first), not developing new code. So I was deeply pleased when Spinellis acknowledged the importance of that topic, and provided a well-thought-through text on how to do it.

But that raised the interesting question of what Spinellis would do for an encore! How many times can you be the one to open up a whole new line of inquiry? Well, the bad news is that his new book, *Code Quality,* doesn't open up such a new line. But the very good news is that in this, his second book, Spinellis tackles what I would assert is the most important and the most perplexing topic in software engineering, software quality. The topic is important, of course, because without sufficient quality code may well be worthless. And it is perplexing because it often seems like there are as many definitions of quality in the software field as there are people writing about it.

Not only does Spinellis tackle this important and perplexing topic, but he tackles it very well indeed. In a world where most discussions of software quality are management-focused and high-level, Spinellis attacks the nitty-gritty, vital subject of the technology of quality as it is reflected in code quality. In my own (highly biased) view, management level discussions of quality come close to being worthless, because the factors that make up the subject of quality can only be discerned at the level of the code that implements it. Take maintainability and portability, two quality attributes that Spinellis discusses, as examples. It is simply not possible to understand how maintainable and portable software is without such a code analysis.

For those readers who aspire to leap past the technology of software into the rarefied echelons of its management—and, in my view, there are all too many of those out there—this is not the right book to understand quality. But for those readers who understand that quality is deeply technical before it can ever be a management topic, this book is the right place to start. The author starts off his description of his book by saying "from this book you will learn how to judge the quality of software code." Hooray for him!

Robert L. Glass
January, 2006