design. After this well-publicized failure, no suspension bridge was undertaken without extensive wind-tunnel testing of models.

The 1981 collapse of the suspended walkways at the Kansas City Hyatt Regency taught the engineer another lesson—that even a sufficient design can be subverted during construction. The drawings showed connections adequate to hold two walkways suspended, one over the other, from the lobby ceiling. But a shortcut was taken during construction and the connections were altered. This change turned out to be critical, for it made the upper walkway barely able to support its own weight, much less that of a crowd gathered on it. The price of the shortcut was ultimately paid in 114 lives, but the lesson to engineers was immensely valuable.

Petroski conducts an anatomy of failure, one thought-provoking to both the engineer and those who care about engineers and their craft. He argues persuasively for the study of failure: "...nothing can erase an engineering disaster. Yet no disaster need be repeated, for by talking and writing about the mistakes that escape us we learn from them, and by learning from them we can obviate their recurrence."

Petroski is a structural engineer and hence his examples of engineering failure are largely those of failure of bridge, building, and vehicle. But his premise generalizes to all engineering. The computer engineer will find a great deal to think about here, especially since his designs are arguably the most complex structures ever undertaken by man. It would be valuable for a computer engineer to carry Petroski's analysis of failure into the microelectronic world, especially since computer engineering is not a mature discipline in the way structural engineering is, and, presumably, is still at the point in its development where failure is routine. What do computer engineers make of their mistakes? How can the study of those mistakes improve the craft?

The computer scientist will find a chapter addressed directly to his concerns. "From Slide Rule to Computer: Forgetting How It Used To Be Done" examines the effects of building engineering expertise into computer programs. Who should write these programs? Should engineers trust such programs? Should engineering programs, like engineers themselves, be registered?

Computer engineer and computer scientist alike should read this book, especially as chip and code enter tool, building, and vehicle. As our dependence on computers grows, so does our dependence on the skill and humility of their designers. Computer engineers and computer scientists would do well to create their own literature of failure and to read it closely.