Foreword by Grady Booch

There is a wonderful book by Anique Hommels called *Unbuilding Cities: Obduracy in Urban Sociotechnical Change* that speaks of the technical, social, economic, and political issues of urban renewal. Cities grow, cities evolve, cities have parts that simply die while other parts flourish; each city has to be renewed in order to meet the needs of its populace. From time to time cities are intentionally refactored (many cities in Europe were devastated in World War II and so had to be rebuilt); most times cities are refactored in bits (the Canary Warf renewal in London comes to mind), but sometimes the literal and figurative debt is so great a once great city falls into despair (present day Detroit, for example).

Software-intensive systems are like that. They grow, they evolve, sometimes they wither away, and sometimes they flourish. The concept of technical debt is central to understanding the forces that weigh upon systems, for it often explains where, how, and why a system is stressed. In cities, repairs on infrastructure are often delayed and incremental changes are made rather than bold ones. So it is again in software-intensive systems. Users suffer the consequences of capricious complexity, delayed improvements, and insufficient incremental change; the developers who evolve such systems suffer the slings and arrows of never being able to write quality code because they are always trying to catch up.

What delights me about this present book is its focus on technical debt and refactoring as the actionable means to attend to it.

When you have got a nagging tiny gas leak on a city block, a literal smell will lead you to the underlying cause. Software-intensive systems are like that as well, although the smells one may observe therein are far more subtle and invisible to all the senses save to the clever cognitive ones. As you read on in this book, I think you will find one of the more interesting and complex expositions of software smells you will ever find. Indeed, you have in your hand a veritable field guide of smells, sort of like the very real *Birds of the West Indies*, except about software, not sparrows.

Abstraction, encapsulation, modularization, and hierarchy: these are all elements fundamental to software engineering best practices, and you will see these highlights explained as well. In all, I think you will find this a delightful, engaging, actionable read.

Grady Booch
IBM Fellow and Chief Scientist for
Software Engineering, IBM Research