Foreword

When Jim Rumbaugh, Ivar Jacobson, and I set out to define the original version of the Unified Modeling Language (UML), we observed then—as we still do now—that the role of the UML was to "visualize, specify, construct, and document the artifacts of a software-intensive system." Notice our emphasis on visualization: we visualize in order to reason about complex structures and behavior. By choosing the right visualization, we are able to rise above the details of specific implementation languages and other technologies so that we can specify, construct, and document the patterns and the multitude of other design decisions that shape a system's architecture.

Now, the term "architecture" is an emotionally laden term. To some, it represents a heavyweight artifact of a high-ceremony process; to others, it's just the same as design. For me, however, software architecture represents the significant design decisions that form a software-intensive system; all software architecture is design, but not all design is software architecture. Furthermore, my experience has been that every such system has an architecture, although some are intentional while others are accidental. Nonetheless, every successful organization tends to grow its system's architecture incrementally and iteratively.

I've had the pleasure of working with Terry for many years, and she deeply understands the importance and the practice of visualizing with the UML. In this book, she focuses on visualizing various aspects of a system's architecture as it grows through the life cycle, from use cases at inception through analysis, design, implementation, and even test cases during elaboration and implementation.

Terry's style is always direct, approachable, and pragmatic. Abstraction is hard, and visualizing abstractions is yet another hard problem, but here she'll guide you in doing both using Rational Software Architect.

Grady Booch
IBM Fellow