FOREWORD

Programming is fun, but developing quality software is hard. In between the nice ideas, the requirements or the "vision," and a working software product, there is much more than programming. Analysis and design, defining how to solve the problem, what to program, capturing this design in ways that are easy to communicate, to review, to implement, and to evolve is what lies at the core of this book. This is what you will learn.

The Unified Modeling Language (UML) has become the universally-accepted language for software design blueprints. UML is the visual language used to convey design ideas throughout this book, which emphasizes how developers really apply frequently used UML elements, rather than obscure features of the language.

The importance of patterns in crafting complex systems has long been recognized in other disciplines. Software design patterns are what allow us to describe design fragments, and reuse design ideas, helping developers leverage the expertise of others. Patterns give a name and form to abstract heuristics, rules and best practices of object-oriented techniques. No reasonable engineer wants to start from a blank slate, and this book offers a palette of readily usable design patterns.

But software design looks a bit dry and mysterious when not presented in the context of a software engineering process. And on this topic, I am delighted that for his new edition, Craig Larman has chosen to embrace and introduce the Unified Process, showing how it can be applied in a relatively simple and low-ceremony way. By presenting the case study in an iterative, risk-driven, architecture-centric process, Craig's advice has realistic context; he exposes the dynamics of what really happens in software development, and shows the external forces at play. The design activities are connected to other tasks, and they no longer appear as a purely cerebral activity of systematic transformations or creative intuition. And Craig and I are convinced of the benefits of iterative development, which you will see abundantly illustrated throughout.

So for me, this book has the right mix of ingredients. You will learn a systematic method to do Object-Oriented Analysis and Design (OOA/D) from a great teacher, a brilliant methodologist, and an "OO guru" who has taught it to thousands around the world. Craig describes the method in the context of the Unified Process. He gradually presents more sophisticated design patterns—this will make the book very handy when you are faced with real-world design challenges. And he uses the most widely accepted notation.

I'm honored to have had the opportunity to work directly with the author of this major book. I enjoyed reading the first edition, and was delighted when he asked me to review the draft of his new edition. We met several times and exchanged many e-mails. I have learned much from Craig, even about our own process work on the Unified Process and how to improve it and position it in various organizational contexts. I am certain that you will learn a lot, too, in reading this book, even if you are already familiar with OOA/D. And, like me, you will find yourself going back to it, to refresh your memory, or to gain further insights from Craig's explanations and experience.

Happy reading!

Philippe Kruchten
Professor of Software Engineering, University of British Columbia

formerly,
Rational Fellow and Director of Process Development for the RUP
Rational Software
Vancouver, British Columbia