UML Xtra-Light

UML Xtra-Light

How to Specify Your Software Requirements

Milan Kratochvíl **Barry McGibbon**

CAMBRIDGE UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011–4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2002

Printed in the United States of America

Typefaces Stone Serif 10.5/13 pt., Stone Sans, and Informal *System* QuarkXPress® [GH]

A catalogue record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data

Kratochvíl, Milan.
UML xtra-light : how to specify your software requirements / Milan Kratochvíl, Barry McGibbon.
p. cm.
Includes bibliographical references and index.
ISBN 0-521-89242-2
1. Application software – Development. 2. UML (Computer science) I. McGibbon, Barry, 1947– II. Title.
QA76.76.A65 K72 2002
005.1 – dc21
200219253

ISBN 0 521 89242 2 paperback

To my father, Jiří, and to generations of composers before him who realized that architectural standards, components, and reuse all boost creativity and invention.

Milan Kratochvíl

To my wife, Vicky, for her lifetime of love and support

Barry McGibbon

Contents

Foreword ix
Preface xi
Acknowledgments xiii
About the Authors xv
How to Customize This Book xvii

Chapter 1 • Introduction 1
Software – Yet Another Knowledge Industry 1
Classifying the Knowledge Industry 2
Consequences of the Knowledge Industry 3
Sharing the Knowledge 5
Sharing the Responsibility for Getting It Right 6
Methods and Processes 8
Summary 11

Chapter 2 • Aligning to the Business 13
Using UML Activity Diagrams 15
Using Business Use-Case Diagrams 23
But What About the Data? 25
Summary 26

Chapter 3 • Adding Rigor to the Requirements 27
Use Cases 27
Use-Case Example 29
Meeting the Devil 34
Use-Case Analysis at Two Levels, At Least 36
How to Avoid Messing Up Use Cases 39
Summary 46

Chapter 4 • Sketching the Inside Structure 47
Class Diagrams 48
The Class Diagram 50
Understanding Class Relationships 52
Summary 59

Chapter 5 • Sketching the Inside Dynamics 61
State Diagrams 61
Tying It All Together 67
UML Collaboration Diagrams 70
Other UML Diagrams 70
Summary 71

Chapter 6 • Moving Toward Components 73
Components Communicate with Everyone 76
Impact of the Component-Based Approach 79
Reusing Components 81
Building a Component Library 83
Sharing Components in Your Organization 84
Avoiding the Traps 85
Automating the Bid Process 87
Summary 88

Chapter 7 • Mapping from Classes to Data Models 89
Use Appropriate Diagrams and Standards 90
Mapping Relationships 91
Summary 95

Chapter 8 • Concluding Remarks 97
Think Big, Start Small, and Sustain the Effort 97
UML Under Time Constraints 98

Some Suggested Readings 101
Index 103

Foreword

Yet another book about UML! Since its initial version, the Unified Modeling Language has gone an impressive way in the IT community. Over the past couple of years, we have been loading our bookcase with quite a few UML books. Many of them deal with applying or extending UML for a specific domain: UML for project management, UML for business modeling, UML for Java, real-time UML, UML for components, UML for web applications, and so forth.

This book takes a somewhat different and, in our opinion, long-awaited approach. It goes back to the *basics* of UML: *improving the communication* among different stakeholders of a (software) project. As the authors of the book write: "a UML made easy for people who specify, buy, or manage complex software systems." Many of these stakeholders are non-IT professionals in much need of an easy-to-digest introduction to UML.

Looking back on the IBM SanFrancisco project – one of the *real success projects* in the field of object-oriented *business applications* – where we, at IBS, played a central role as initiators and principal development partner to IBM, a key success factor was the alignment among domain experts, sponsors, and object experts – through a minimum set of concepts and techniques.

In addition, this book focuses on the new paradigm in software development: fast delivery of applications based on components sourced from various suppliers. Even though UML initially placed considerable focus on creating applications from scratch, successful software projects today are all about *creating, buying, and integrating software components*. This leaves us (who intend to stay competitive and successful!) extremely dependent on a *standard notation* for any software-related communication, specification, and knowledge sharing. The IBM SanFrancisco/WebSphere Business components provide an extreme case for this. By using a standard notation language for the component specs, any potential application builder will be able to understand, integrate, and extend the components. Furthermore, tool vendors have been easily able to integrate the components in their tool sets for modeling and code generation.

Enjoy!

Staffan Ahlberg
CEO
IBS AB
www.ibs.se

Tomas Bräne
VP Research & Development
IBS AB
tomas.brane@ibs.se

Preface

The excellent idea of writing a lightweight book on the Unified Modeling Language (UML) wasn't ours, we admit. This idea originated from Milan's customers. Having taught more than a hundred courses and seminars on component approaches to software development and on UML over the past few years, he was repeatedly asked for "UML made easy" for people who specify, buy, or manage complex software systems, yet don't program them. This demand seems logical given the way UML is being used in projects and read of in the success stories[1] – as well as the increasing specification workload in any knowledge industry (see Introduction). However, as we moved on into this book project, both of us became increasingly enthusiastic about the idea, as did Cambridge University Press (CUP). Luckily, a majority of our readers are quite familiar with CUP from their own (variety of) fields; so this book is likely to be seen as accessible in most senses of the word.

Any system specification can state requirements on functionality, usability, reliability, performance, and supportability, as well as legal and technical constraints where relevant. In UML projects, we start from a view of the business – its processes and activities – and move into functionality, increment-

[1] The Object Management Group (OMG) owns and upgrades the UML standard; visit *www.omg.org*.

ing all the remaining, nonfunctional, bullet lists as we go. These are then resolved later, during construction, rather than during specification. As stressed in the chapter on components as well as implied throughout the book, wherever we're on the scale between "buy" and "build," the specification work and business analysis just don't simply disappear. Even with an off-the-shelf system, we still specify our requirements, and we still need to understand the essence of all those UML diagrams.

To keep this book lightweight, we stay reasonably lightweight on the art of balancing the content of internal/technical UML views. This kind of balance is key down the road, that is, later on in a software development project. It requires modeling the right aspects in the appropriate diagram view at a right level of detail in the initial stage of a project. However, we chose to appeal to the reader's common sense by pointing out the natural boundaries between the process view, the use-case view, and the structural (or conceptual) view, with the strengths and limitations of each view. Neither a blueprint of a building nor one of a software system can show everything at once. Some drawings depict the walls and the roof, others electricity, and yet others heating, air conditioning, water, and drainage; that is, we separate the concerns. What is noteworthy is that people proposing buildings learn quickly to keep away (hide) electricity aspects from the exterior view and vice versa.

Standards and components are a *serious boost to productivity* in software. In our experience, however, these are more likely to be practiced when introduced in a step-by-step, nonacademic, and not too reserved manner, as outlined in this book. As an enterprise sets its mind on component reuse, all professionals from junior programmers to top management become involved and, consequently, need to be offered a brief guidebook within their frame of reference. So, for software specialists struggling to shift from a detailed code-based approach to the conceptual models of the software design and architecture, we recommend exploring UML beyond this lightweight version.

Acknowledgments

As mentioned in the Preface, many people have gradually made us realize the need for a lightweight book like this, thus indirectly pushing it through. Thanks to many people at the Object Management Group, Aonix-Select UK, Linsoft, Cell Network, Rational Scandinavia, IBS. Thanks to Marie-Louise Westerberg and Esa Falkenroth (Swedish Meteorological and Hydrological Institute), Leif-Åke Andersson (Swedish Customs/IT), Eva Backe (Integra Enterprise Systems), Anna Hermansson (Ericsson Telecom), Bjørn-Erik Willoch (Institute of Process Innovation, now CAP Management Consulting), Stanislav Mlynář CEO, LBMS Prague), Esa Rantanen (Sema Group), Annika Hansen-Eriksson (Royal Institute of Technology, now at Sema Group), and many others. For hints on knowledge enterprises, thanks to Peter Stevrin (associate professor, IT Management, Blekinge Institute of Technology) and Leif Edvinsson (Manager Intellectual Capital, at Skandia). For my blueprint thinking as well as the life-cycle aspect introduced in the earliest decades of systems development, thanks to my earlier employer, Michael A. Jackson. Thanks to Johan Wretö (Wreto.com) and Dennis Parrot (Select Software Tools, now at iPlanet) for interesting hints on teaching UML to others. Special thanks to Richard M. Soley (Chair and CEO, Object Man-

agement Group) for his enthusiasm and encouragement at an early stage, after the OMG day in Stockholm.

Finally, our publisher, Lothlórien Homet at Cambridge University Press, certainly deserves considerable thanks for keeping her humor throughout the process and for balancing the text skillfully between "practically nothing" and "impractically heavyweight" (also, Lothlórien more or less banned most of my favorite, extremely compact but neither tidy nor especially comprehensible stock phrases, so thanks on behalf of the readers, too).

Milan Kratochvíl

To all my friends and colleagues over the years, especially Steve Latchem, Dave Piper, Baz Maybank, Chris Simons, Adam Partridge, Dave West; to Lothlórien Homet, my brilliant publisher; and to all my super clients, without whom this book would not have been written.

Barry McGibbon

About the Authors

Milan Kratochvíl

A degree in Business & Administration and Data processing, Stockholm University.

Born in Prague, living in the dynamic silicon area around Stockholm-Kista (ranked by *Wired* as a global number 2).

Working since 1977 as an IT consultant, instructor, and writer in methodology; independent since 1989, focusing on areas where IT and knowledge-intensive business meet.

Taught far more than a hundred courses and seminars on a commercial basis for developers, managers, or buyers of complex systems. Published several articles, reports, congress papers on methodology and knowledge management. An initiator/catalyst and project leader of three experience-exchange pools with The Swedish Computer Society in Stockholm a few years ago.

Lessons learned: there are two things in this world you should reuse every day – jokes and components.

Barry McGibbon

Worked in the IT industry since 1966, gaining a wide variety of experience, ranging from programming through to holding senior management positions with leading computing services and product providers.

A consultant since 1985, with involvement in numerous major initiatives for significant enterprises in the United States, Europe, and the United Kingdom.

Provides advice and counsel on managing software development, methodologies, improvement strategies, capability evaluations, and quality management systems.

Lectures widely in the United States, the United Kingdom, and Europe. Author of *Managing Your Move to Object Technology: Guidelines & Strategies for a Smooth Transition,* published by SIGS Books Inc., and a contributor on *Component Based Software Engineering,* published by Addison-Wesley. Technical chairman for Europe's largest component and object technology conference and a series editor for Cambridge University Press.

How to Customize This Book

Most readers of this book suffer from a lack of time, so here's a guide on how you can focus on the key chapters relevant to your role.

Process owners, reengineers, and similar roles usually consider the topics of *Chapters 1 and 2* as the essence of a project, so you're advised to read those chapters thoroughly and browse through the rest.

End-user representatives or other roles involved in man-machine interaction (MMI), manuals, user training, user interfaces (UI), or interfaces to other systems are advised to focus on *Chapter 3*.

Domain experts are advised to concentrate on *Chapter 4* and to browse through *Chapters 3 and 5*.

Managers, project-plan coordinators, venture capitalists, headhunters, or PA people could read *Chapters 1, 2, and 3* quickly and focus on *Chapter 6*.

Others reading the book simply out of curiosity might want to browse through the art first, and then choose topics that interest them for a second iteration.

In general, those not interested in specific details can skip the footnotes and boxes.