UML与面向对象设计影印丛书

高级用况建模

ADVANCED USE CASE MODELING

SOFTWARE SYSTEMS

FRANK ARMOUR
GRANVILLE MILLER 编著

OBJECT TECHNOLOGY SERIES

BOOCH
JACOBSON
RUMBAUGH

SERIES EDITORS

ADDISON-WESLEY

科学出版社
www.sciencep.com

Pearson Education

UML与面向对象设计影印丛书

高级用况建模

Frank Armour
Granville Miller 编著

科学出版社
北 京

内 容 简 介

软件开发最为重要的是问题定义阶段，用况驱动的分析技术正成为软件建模的首选，但这方面著作尚不多见。本书通过诸多实例，具体而直观地为读者介绍了复杂软件项目的用况模型创建过程，内容包括基本知识、项目启动、初级用况模型、用况模型扩展等。本书提供了一个用于创建和维护用况模型的框架，读者可将其定制以适合自己的系统。

本书可供软件开发过程各阶段的技术人员阅读。

图书在版编目(CIP)数据

高级用况建模=Advanced Use Case Modeling/（美）米勒（Miller.F.A.G）著.—影印本. —北京：科学出版社，2003

ISBN 7-03-011398-5

Ⅰ.高… Ⅱ.米… Ⅲ.软件开发—建立模型—英文 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2003）第 030810 号

策划编辑：李佩乾/责任编辑：李佩乾
责任印制：吕春珉/封面制作：东方人华平面设计室

科学出版社 出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com
双青印刷厂 印刷
科学出版社发行 各地新华书店经销
*
2003年5月第一版 开本：787×960 1/16
2003年5月第一次印刷 印张：28 1/2
印数：1—2 000 字数：545 000

定价：48.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

影印前言

随着计算机硬件性能的迅速提高和价格的持续下降，其应用范围也在不断扩大。交给计算机解决的问题也越来越难，越来越复杂。这就使得计算机软件变得越来越复杂和庞大。20 世纪 60 年代的软件危机使人们清醒地认识到按照工程化的方法组织软件开发的必要性。于是软件开发方法从 60 年代毫无工程性可言的手工作坊式开发，过渡到 70 年代结构化的分析设计方法、80 年代初的实体关系开发方法，直到面向对象的开发方法。

面向对象的软件开发方法是在结构化开发范型和实体关系开发范型的基础上发展而来的，它运用分类、封装、继承、消息等人类自然的思维机制，允许软件开发者处理更为复杂的问题域和其支持技术，在很大程度上缓解了软件危机。面向对象技术发端于程序设计语言，以后又向软件开发的早期阶段延伸，形成了面向对象的分析和设计。

20 世纪 80 年代末 90 年代初，先后出现了几十种面向对象的分析设计方法。其中，Booch, Coad/Yourdon、OMT 和 Jacobson 等方法得到了面向对象软件开发界的广泛认可。各种方法对许多面向对象的概念的理解不尽相同，即便概念相同，各自技术上的表示法也不同。通过 90 年代不同方法流派之间的争论，人们逐渐认识到不同的方法既有其容易解决的问题，又有其不容易解决的问题，彼此之间需要进行融合和借鉴；并且各种方法的表示也有很大的差异，不利于进一步的交流与协作。在这种情况下，统一建模语言(UML)于 90 年代中期应运而生。

UML 的产生离不开三位面向对象的方法论专家 G. Booch、J. Rumbaugh 和 I. Jacobson 的通力合作。他们从多种方法中吸收了大量有用的建模概念，使 UML 的概念和表示法在规模上超过了以往任何一种方法，并且提供了允许用户对语言做进一步扩展的机制。UML 使不同厂商开发的系统模型能够基于共同的概念，使用相同的表示法，呈现彼此一致的模型风格。1997 年 11 月 UML 被 OMG 组织正式采纳为标准的建模语言，并在随后的几年中迅速地发展为事实上的建模语言国际标准。

UML 在语法和语义的定义方面也做了大量的工作。以往各种关于面向对象方法的著作通常是以比较简单的方式定义其建模概念，而以主要篇幅给出过程指导，论述如何运用这些概念来进行开发。UML 则以一种建模语言的姿态出现，使用语言学中的一些技术来定义。尽管真正从语言学的角度看它还有许多缺陷，但它在这方面所做的努力却是以往的各种建模方法无法比拟的。

从 UML 的早期版本开始，便受到了计算机产业界的重视，OMG 的采纳和大公司的支持把它推上了实际上的工业标准的地位，使它拥有越来越多的用户。它被广泛地用

于应用领域和多种类型的系统建模，如管理信息系统、通信与控制系统、嵌入式实时系统、分布式系统、系统软件等。近几年还被运用于软件再工程、质量管理、过程管理、配置管理等方面。而且它的应用不仅仅限于计算机软件，还可用于非软件系统，例如硬件设计、业务处理流程、企业或事业单位的结构与行为建模，等等。

在 UML 陆续发布的几个版本中，逐步修正了前一个版本中的缺陷和错误。即将发布的 UML2.0 版本将是对 UML 的又一次重大的改进。将来的 UML 将向着语言家族化、可执行化、精确化等理念迈进，为软件产业的工程化提供更有力的支撑。

本丛书收录了与面向对象技术和 UML 有关的 12 本书，反映了面向对象技术最新的发展趋势以及 UML 的新的研究动态。其中涉及对面向对象建模理论研究与实践的有这样几本书：《面向对象系统架构及设计》主要讨论了面向对象的基本概念、静态设计、永久对象、动态设计、设计模式以及体系结构等近几年来面向对象技术领域中的新的理论知识与方法；《用 UML 进行用况对象建模》主要介绍了面向对象的需求阶段、分析阶段、设计阶段中用况模型的建立方法与技术；《高级用况建模》介绍了在建立用况模型中需要注意的高级的问题与技术；《UML 面向对象设计基础》则侧重于经典的面向对象理论知识的阐述。

涉及 UML 在特定领域的运用的有这样几本：《UML 实时系统开发》讨论了进行实时系统开发时需要对 UML 进行扩展的技术；《用 UML 构建 Web 应用程序》讨论了运用 UML 进行 Web 应用建模所应该注意的技术与方法；《面向对象系统测试：模型、视图与工具》介绍了将 UML 应用于面向对象的测试领域所应掌握的方法与工具；《对象、构件、框架与 UML 应用》讨论了如何运用 UML 对面向对象的新技术——构件-框架技术建模的方法策略。《UML 与 Visual Basic 应用程序开发》主要讨论了从 UML 模型到 Visual Basic 程序的建模与映射方法。

介绍面向对象编程技术的有两本书：《COM 高手心经》和《ATL 技术内幕》，深入探讨了面向对象的编程新技术——COM 和 ATL 技术的使用技巧与技术内幕。

还有一本《Executable UML 技术内幕》，这本书介绍了可执行 UML 的理念与其支持技术，使得模型的验证与模拟以及代码的自动生成成为可能，也代表着将来软件开发的一种新的模式。

总之，这套书所涉及的内容包含了对软件生命周期的全过程建模的方法与技术，同时也对近年来的热点领域建模技术、新型编程技术作了深入的介绍，有些内容已经涉及到了前沿领域。可以说，每一本都很经典。

有鉴于此，特向软件领域中不同程度的读者推荐这套书，供大家阅读、学习和研究。

北京大学计算机系　蒋严冰 博士

Foreword

When I came up with the use case concept in 1986, it was based on many years of work in component-based system development. We had had many other different techniques to do the job, techniques that were overlapping and had gaps. With use cases we got a tool with many facets. Some of them are:

- Use cases are the requirements capture vehicle.
- Use cases are the base for defining functional requirements.
- Use cases facilitate envisioning applications.
- Use cases assist in system delimitation.
- Use cases are the means to communicate with end users and customers.
- Use cases provide the dynamic, black-box view of the system.
- Use cases are the base for object derivation; objects naturally fall out of use cases.
- Use cases provide a tool for requirements traceability.
- Use cases are the base for user interface and experience design.
- Use cases facilitate moving from functional requirements to object and component structures.
- Use cases are the base for allocating functionality to components and objects.

- Use cases are the mechanism to define object interaction and object interfaces.
- Use cases define access patterns to a database.
- Use cases help us with dimensioning of processor capacity.
- Use cases are the base for integration testing.
- Use cases define test cases.
- Use cases are the base for incremental development.
- Use cases help us with estimation of project size and required resources.
- Use cases provide a base for user documentation and manuals.
- Use cases are a tool for controlling a project.
- Use cases drive the development activities.
- Use cases have become the standard way of representing business processes.
- Use cases are used to describe what a legacy system is doing in a reengineering activity.
- Use cases are used when reengineering a business to become an e-business company.

And the list goes on. Of course, I certainly recognize that use cases are not the snake oil or the silver bullet of software development. However, the development of the use case idea has just started.

Advanced Use Case Modeling provides a set of guidelines for developing a use case model for software systems. It provides a toolkit of techniques to be utilized by experienced use case modelers. As with any toolkit, each tool has an intended purpose and the right tool should be selected for each job. Frank and Randy have done an excellent job of providing techniques that reflect their vast experience in the industry.

The use case continues to drive business processes, software systems, and component engineering projects. Some of the original concepts behind this modeling technique have evolved through the work of researchers, practitioners, and the standards bodies, but the fundamental ideas remain the same. The use case is an elegant way of communicating the needs of a business or software system. I am sure that, over time, new communication needs will lead to other ways of employing use cases; the possibilities are endless.

Ivar Jacobson

Preface

The use case approach is increasingly popular with our customers, to an extent that some will only specify systems using them.
—Anthony Heritage and Phil Coley [Heritage 1995]

In this rapidly changing business and technological environment, use case modeling has emerged as one of the premier techniques for defining business processes and software systems. Business engineers now employ use cases to define complex business processes across lines of business and even to define entire businesses. Use cases are also the standard for defining requirements for the software systems created using today's object-oriented development languages such as Java, Smalltalk, and C++. In the field of software components, a very young industry whose market is estimated to be more than $12 billion in 2001 [Hanscome 1998], use cases are rapidly becoming a method of communication between suppliers and vendors.

The users of this technique for defining systems are as diverse as its uses. Use case modeling is already being employed by most Fortune 1000 companies and is being taught at many academic institutions all over the world, and the popularity of this modeling technique continues to grow.

Business process and software requirements engineering are rapidly evolving fields. Research in these areas continues to propose new methods of dealing with potential problems, even while actual practice is slow to adopt only a fraction of those proposed. This slow-moving partial adoption has been termed the "research–practice gap" [Berry 1998]. Creating yet another use case book without an extensive experience base would merely add to this gap. Our approach is significant because we present a practitioner's approach firmly grounded in the real world.

Goals

Over the past six years, we have worked on some large, ambitious projects involving software development and business engineering. To create the best possible use case models, we found it necessary to extend the seminal work of Ivar Jacobson in certain areas. This book details our extensions, which complement Ivar's ongoing work. The flexibility of use case modeling and the Unified Modeling Language, which we use to describe these models, allows us to produce extensions to solve real-world problems successfully.

The goal of this book is to further the advancement of use case modeling in software and business engineering. To achieve this goal, the book provides a comprehensive yet readable guide to use case modeling for the practitioner. Specifically, it explains advanced use case modeling concepts, describes a process for implementing use case modeling, and discusses various use case modeling issues.

Audience

The audience for this book is anyone involved in the conceptualization, development, testing, management, modeling, and use of software products and business processes. Although it contains a sizable amount of content related to business processes, this book is geared toward all of us in the software industry. Software professionals are the largest body of use case engineers because use case development was first introduced as a software requirements vehicle.

Business analysts will agree that use case engineering has undergone the greatest transformations on their front. Business analysts and their software process brethren are quickly learning that automation via software is not the only reason for employing use cases. In fact, more and more of business process modeling using use cases is not geared toward the generation and production of new software but is being done to understand, and in some cases, standardize and optimize key business processes across multiple lines of business.

Many of the techniques described in this book transcend the software or business arenas of the reader community. The well-established link between business use cases and software system use cases is described as we illustrate the ways in which software systems can be derived from a business process. The only thing we ask is that our business readers be patient as we start on the software side.

Academic institutions will also find this book useful. This book can be used as a text in an object-oriented analysis (OOA) course in which use cases play a key role.

How to Use This Book

The theory of use case development often differs from the actual practice of use case development. One reason for this difference is that very few software development projects are "green fields"; most are started with a preconceived notion of a legacy process for successfully creating software. We are not advocating the removal of the legacy processes. In fact, many of the artifacts involved in these processes may be necessary due to the nature of the problem that is being solved through software development. Some of these artifacts may also be mandatory for getting the necessary approval to begin a software development project.

Use case modeling cannot be successful in isolation. The process of creating use case models must be put in the context of the specific organization. Every organization has unique cultural aspects. Luckily, we find some commonality as well as differences in nearly every facet of the business engineering and software development processes across organizations.

Experience in one organization can often be useful in another. When patterns of failure have emerged from our use case adventures, we have attempted to capture the factors that have been directly responsible. The pitfalls of use case modeling generally fall into two categories: those in the use case development process itself and those found when use cases are integrated with commonly used software development practices. Some of the pitfalls are so significant that they can stop the development of a system dead in its tracks.

This book provides a process framework for creating models of software systems. A **process framework** is a set of activities used to develop a process. Our frameworks should be customized specifically for your organization. This book describes the second of the three process frameworks (Figure P-1), the conceptualization and specification of software systems.

Each process framework is independent and fully defined. They may all be performed in concert or separately. For example, software system and component engineering may be used together to provide requirements for software system development using components. The combination of business process and software system engineering creates an understanding of the elements necessary for business process automation. A business process is not usually completely automated via software systems. The requirements for the business process, therefore, become a superset of those of the software systems used by people carrying out the business process.

The three frameworks provide a means for specifying the requirements for engineering all of the systems required for business process automation, incorporating software building blocks. When process frameworks are combined, the outputs created during the previous framework may be utilized as inputs to the next.

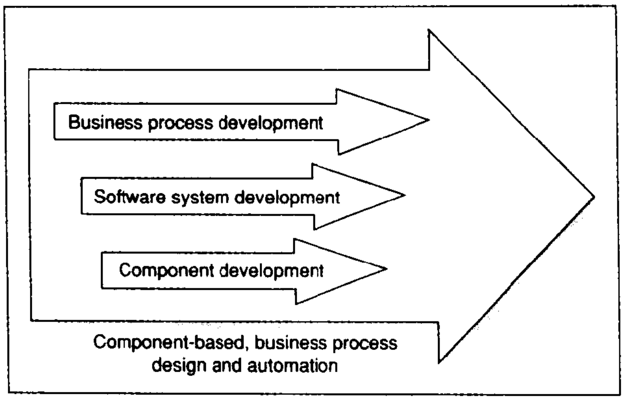
Business process development
Software system development
Component development
Component-based, business process design and automation

FIGURE P-1 Process frameworks of the advanced use case modeling process

To make the most of this book, we recommend following an established software development process. We respect the notion that not all companies are capable of following a software development process in exactly the same way. The **ceremony**, or amount of formality, involved usually differs dramatically from company to company and even from project to project [Booch 1996].

Ceremony helps define how much of a process framework to use [Miller 2000]. High-ceremony projects tend to utilize more of the activities, perhaps adopting advanced use case modeling wholesale. Low-ceremony projects use only a portion of the material described. Regardless of the level of ceremony, you will certainly find use cases in some form useful for the definition of requirements for a project.

Organization and Content

There are many books on use cases available on the market today. Ours is unique in its coverage of the role of use cases in software development. We also present some substantially new material not found in any other paper or book. We balance this new material with a comprehensive survey of the existing work in the field of use case modeling.

To allow this book to stand on its own, we present two chapters of fundamental material. These two chapters begin after an introduction to advanced use case modeling. Chapter 1 discusses the conceptual role of actors in the use case model. A detailed account of how to recognize actors is provided to prepare the use case modeler to discover use cases. Chapter 2 discusses the general format and proto-

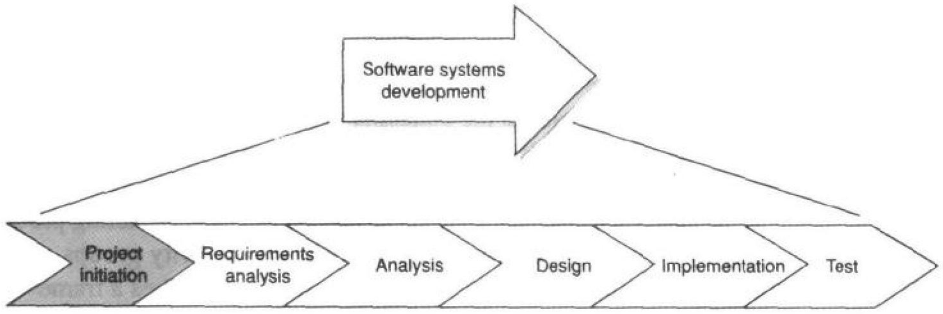

FIGURE P-2 Generic phases of a software development process

col for creating use cases. The Unified Modeling Language, the Object Management Group (OMG) standard for use case modeling, is explained.

Part 2 starts with the first phase of software development, project initiation (Figure P-2). Chapter 3 focuses on this phase by looking at the things that define the system scope—the problem that is to be solved and the business opportunity created by the new or improved system, and the financial feasibility of building a software system to address this opportunity.

Chapter 4 describes use case modeling in the requirements analysis phase of the software development process (Figure P-3). Use cases help to describe the functions

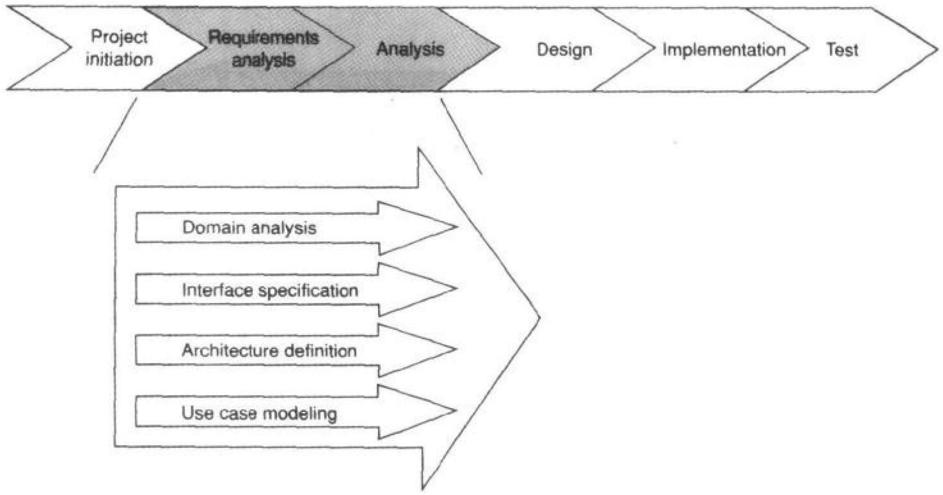

FIGURE P-3 Decomposition of the requirements analysis and partial analysis phases of the system and to balance the use case model; form is provided with a well-designed architecture that can enhance the use case model.

In Part 3, we introduce a bank loan application example that is used throughout the book to illustrate the concepts of use case modeling. The example does not represent any actual loan system. The necessary functionality for an actual loan application has been streamlined for purposes of the example.

In Part 3 we also describe the advanced use case modeling process framework. Chapter 5 decomposes use case modeling into activity groups, or groups of logically related activities (Figure P-4). The chapter describes a framework for use case modeling that is used to describe system use case modeling through Chapter 15.

Chapter 6 describes the initial steps in setting up a use case modeling effort. The selection and customization of use case frameworks, the selection of standards and techniques, and the consideration of training and mentoring needs are outlined.

Chapter 7 discusses the initial steps of creating the use case model. The outcome of this activity group is a use case model that captures a "conceptual" picture of what the system will need to do.

Part 4 focuses on expanding the use case model. Chapter 8 begins the discussion of how initial use case descriptions are expanded to become base use cases with more detailed requirements and how this increased complexity is modeled. Chapter 9 discusses the practice of placing conditional and iterative logic within a use case's flow of events. Two techniques for modeling these concepts are presented.

Chapter 10 describes the use of extend, include, and generalization relationships to model the alternatives, variations, and commonality in the use case model.

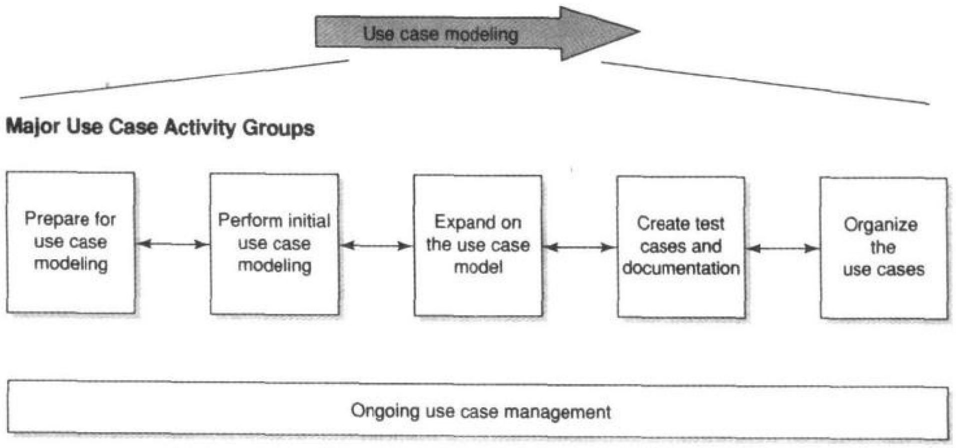
Use case modeling
Major Use Case Activity Groups
Prepare for use case modeling
Perform initial use case modeling
Expand on the use case model
Create test cases and documentation
Organize the use cases
Ongoing use case management

FIGURE P-4 The advanced use case modeling process framework

Chapter 11 discusses the capture of additional or supplemental information associated with an individual use case. Chapter 12 discusses the importance of mapping the use cases to the analysis object model. Techniques such as CRUD matrixes, object to use case tables, and sequence diagrams are outlined. Chapter 13 discusses the concept and utilization of scenarios to complement the use case model.

The final phase of any software engineering process is testing. Chapter 14 discusses testing and documenting the system and the role use cases play in driving these activities. Chapter 15 examines organizing use cases by business functional packages and by dependencies based on use case preconditions and postconditions. A discussion of various views of the use case model is presented. A wrap-up of key use case artifacts is also presented.

Part 5, Additional Topics, begins with Chapter 16. This chapter examines the effect of use cases on user interface design. Transactions are used to segment the use case model to provide elements for conceptual user interface development. Grouping techniques allow screens to be built from the transactions.

Chapter 17 examines the effect of change on the use case model. In successful software systems, changes that affect the functionality of the system are inevitable. Change may occur during the project or after it has shipped.

Chapter 18 discusses some of the necessary considerations for deploying advanced use case modeling. All or part of the process framework may be utilized depending on the needs of the project. This chapter outlines the elements that determine how much to use. It also describes how to document this process.

The final chapter, Chapter 19, discusses the quality attributes of a good use case model. It also describes the various roles that use case modeling can play within a system analysis effort. Finally, iterative and incremental development with advanced use case modeling is briefly outlined.

Complementary Works

This book stands on its own and can be read without referring to other works. However, quite a bit of helpful material is available on requirements engineering, use case development, and process improvement.

Software development process

- Ivar Jacobson, Grady Booch, and James Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, Reading, MA, 1999.
- Dean Leffingwell and Don Widrig, *Managing Software Requirements: A Unified Approach*, Addison-Wesley, Reading, MA, 2000.

- Geri Schneider and Jason P. Winters, *Applying Use Cases: A Practical Guide*, Addison-Wesley, Reading, MA, 1998.
- Rational Software Corporation, *Rational Unified Process*. 2000.

Business process engineering

- Ivar Jacobson, Maria Ericsson, and Agneta Jacobson, *The Object Advantage: Business Process Reengineeering with Object Technology*, Addison-Wesley, Reading, MA, 1995.
- Michael Hammer and James Champy, *Reengineering the Corporation*, Harper Business, New York, 1993.
- Rational Software Corporation, *Rational Unified Process*. 2000.

Component development

- Ivar Jacobson, Martin Griss, and Patrik Jonsson, *Software Reuse: Architecture, Process, and Organization for Business Success*, Addison-Wesley, Reading, MA, 1997.
- Clemens Szyperski, *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, Reading, MA, 1998.

You may notice a number of references to other works in the body of this book. We did an extensive survey of the use case literature that predates the publication of this book and found many ideas worthy of inclusion. We also found many areas where we had developed solutions independently that were similar to those found in the literature. In these cases, we refer to the work in which the idea originally appeared. This gives the reader the flexibility to explore these references to get other viewpoints and gives credit to the other deserving authors.

For the latest information on use cases, supplemental and additional material, or how to contact the authors, visit us at our website, www.advancedusecases.com.

Acknowledgments

Much of this book is based on the work of Ivar Jacobson, inventor and visionary in the field of use case engineering. We have attempted to remain true to his original work throughout this book. We thank Zina Znayenko Miller for tirelessly editing the book, often on very short notice. We would also like to thank our reviewers, who provided valuable feedback in the various stages of the book, especially Monica Gupta and Todd Hansen. This book would not be where it is today without them.

We thank American Management Systems and its clients for many use case modeling opportunities and experiences. We would also like to thank the following individuals: Andy Baer, Chris Ball, Jeff Bitner, Susan Bowler, Lorrie Boyd, Mike Bradley, Bob Brodd, Bill Catherwood, Judy Cohen, Dennis de Champeaux, Peter Dimitrious, Sean Furey, Mary Gorman, Kevin Heineman, Peter Knowles, Steve Larue, John McGregor, Les Moore, Perri-Ann Sims, Mark Schroeder, Chris Tatum, and Patrick Wall. We would also like to thank Brenda Damario, Christine Milliken, and Nora Parker from the U.S. Census Bureau, and Karin Palmkvist from Enea Business Software AB.

Thanks also to these Addison-Wesley reviewers: Jeff Bitner, Senior Principal, Corporate Technology Group, American Management Systems; Maria Ericsson; Monica S. Gupta, former Director of Middleware and Web Integration Lab, AMS Center for Advanced Technologies; Todd Hansen, Senior Architect, Make Systems; Mark Schroeder, Folio[*fn*]; and Sam Supakkul, Senior Architect, Digital Pockets.

We would like to thank all the students of the systems and requirements analysis courses at American University and George Mason University, whose experiences and insights helped refine the book. An early version of this work was presented at a tutorial at OOPSLA '98. The attendees of this tutorial asked some very thought-provoking questions that influenced this book. We would also like to thank Sarah Alijani from American University and Steve Kaisler of the Sergeant of Arms of the U.S. Senate for their contributions.

Thanks also go out to those involved in the publication of this book, Kristin Erickson, Krysia Bebick, Carter Shanklin, and all the folks at Addison-Wesley. Special thanks to Diane Freed, project manager, and Kim Arney, who typeset the book. And finally, we thank all the people in the industry who have been willing to share their experiences.

We hope this book is as useful to you and your organization as it has been to ours.

Frank Armour
Granville Miller
November, 2000

Introduction

The productivity of knowledge is going to be the determining factor in the competitive position of a company, an industry, an entire country.
—Peter F. Drucker [Drucker 1993]

The nature of business has changed, and the signs of this change are all around us. Fundamental methods of business, forged during the industrial revolution and perfected over the last 300 years, have now been forever replaced. The new methods are primarily based on information and technology and driven by global competition. This era, in which production methods are based on knowledge rather than labor and competition, has been coined the *information age*. The pioneers of this age have created new companies so powerful that they can rival or threaten the powerhouses of the established business world.

There is much more to these new business methods than meets the eye. Information and technology are critical but not necessarily sufficient to foster a change of this magnitude to the business landscape. A small start-up company such as Amazon.com is not capable of challenging a Barnes and Noble with technology alone. It has been able to *use* technology and information to create productivity increases and competitive advantage. And it has created competitive advantage in a very organic and imaginative way.

The news of productivity increases and competitive advantage has not been lost on those of us who have not been the actual pioneers of the information age. The rapid acceleration of computer and telecommunications technology has generally resulted in a significant boost in the velocity of business [Greenspan 1999]. It seems that most of us have benefited, even if we don't ourselves use the ad hoc methodologies that propel 10 percent of the start-ups to success. It is, however, important to understand the critical factors that contribute to the general well-being