UML与面向对象设计影印丛书

用UML构建Web应用程序

BUILDING WEB APPLICATIONS WITH UML

JIM CONALLEN 编著

OBJECT TECHNOLOGY SERIES

BOOCH
JACOBSON
RUMBAUGH

ADDISON-WESLEY

SERIES EDITORS

科学出版社
www.sciencep.com

Pearson Education

UML 与面向对象设计影印丛书

用 UML 构建 Web 应用程序

Jim Conallen　编著

科 学 出 版 社

北 京

内 容 简 介

UML 逐渐成为软件系统的标准建模语言，也是 Web 应用程序建模的首选工具。本书第一部分内容介绍了 Web 程序及相关技术，包括 HTTP、HTML、XML、表单、框架、动态客户、安全性等；第二部分完整讲述了 Web 程序构建的过程以及 UML 的具体应用，包括架构定义、需求分析、系统设计、实施等。全书结构清晰，讲求实用。

本书适合 Web 软件项目经理、分析员、设计员及程序员阅读。。

图书在版编目(CIP)数据

用 UML 构建 Web 应用程序=Building Web Applications with UML/（美）科纳伦（Conallen,J.）著.
—影印本. —北京：科学出版社，2003

ISBN 7-03-011402-7

Ⅰ.用… Ⅱ.科… Ⅲ.面向对象语言，UML—程序设计—英文 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 030823 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：19 3/4
印数：1—2 000　字数：377 000

定价: 35. 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

Preface

Late in 1996, I downloaded the preview edition of Microsoft's Active Server Pages. It was my first taste of what could be done on the Web. Even then I could see the potential for sophisticated Web applications. I began to investigate alternative architectures: CGI (Common Gateway Interface) and Allaire's Cold Fusion. Even before then, I had started tinkering with the Java beta and later bought Symantec's Café to experiment with this new language.

At that time, I was an independent consultant working for AT&T in New Jersey. The project had nothing to do with the Web, so my only opportunity to experiment with this technology was during the evenings and whatever spare time I could find. In the end, it was all worth it. I learned a lot and was prepared for the coming onslaught and frenzy of Web application development.

My first opportunity to build a real Web application came at the request of a friend whose father owned a live cut rose wholesale and retail company, Hortico Nurseries Inc. Hortico was interested opening up a retail sales front on the newly emerging Internet. Together with a mutual friend, Jeff Wilkinson, we built our first production e-commerce site. The site was simple. It allowed customers to browse and to search a database of more than 1,400 varieties of roses and even to place orders. At first, the site didn't generate as many orders as we had hoped, but it did expose Hortico to a new market and certainly helped its sales grow in other ways. To the best of our knowledge, Hortico was the first Web site to make a comprehensive catalog of rose varieties and pictures available to the Internet community. Jeff has pretty much taken over the management of the site, and I help when I can. He has gone on to win awards for Web site design for some of his other projects, and I moved on to other contracts.

My first professional contract dealing with Web applications was with a small start-up company in the healthcare business. This experience got me even more involved with the subtleties of building Active Server Pages (ASP) applications, especially with the issues of managing server-side resources and transaction management in a Web application environment. I learned a lot about the use of client-side scripting, applets, and ActiveX controls. I also learned a valuable lesson about testing applications: Client machines with different operating systems can behave differently with the exact same HTML, Java, and browser code. All of these experiences have driven me even more to a belief that Web applications need to be modeled and built just like any other complex software system. In the years that followed, I continued to experiment with the latest Web technologies and consulted with other companies with Web-related issues.

All throughout my Web application experiences, I tried to practice my object-oriented skills in the area of Web application development. I had little problem applying use case analysis, and it wasn't until I started creating analysis and design models that I realized that things were going to get difficult. When creating a Web application, my conceptual focus was always on the Web page. My idea of a model kept revolving around the concept of a site map. I knew that the navigation paths throughout the system were incredibly important to the understanding of the application and that any model of the system would have to include them.

My earliest attempts at modeling Web applications started with Rumbaugh's OMT (Object Modeling Technique); later, when UML version 0.8 was publicly released, I began to apply it. I knew that for any modeling technique to be useful, it needed to both capture the relevant semantics of Web-specific elements, such as Web pages and hyperlinks and their relations to the back-end elements of the system—middle tier objects and databases. At the time, I found both OMT and UML inadequate to express the things I thought were important in a Web application.

Being a somewhat successful object practitioner and engineer, I jumped to the conclusion that a whole new development methodology and notation were needed. After all, if the existing methods and notation didn't have what I needed, the obvious solution was to invent new ones. This, of course, is a trap that many of us in the software industry fall into. In my free time, I started to draft new graphical and semantic ways to represent Web application architectures. Proud of my work, I began showing it to two of my colleagues: Joe Befumo and Gerald Ruldolph, both experienced object practitioners. Their immediate reaction was: *Why?* I tried to explain the issues involved with Web application development and the need for visually expressing their designs. Yet everyone I spoke with continued to think that developing a new method and notation was a little overkill.

I started to rethink what I was doing. I wasn't so arrogant to think that I was still right and everyone else wrong. I had more homework to do. I reexamined my original needs: to express Web application designs at the appropriate level of abstraction and detail, and most important, as a part of the rest of the system's design. Since UML was taking the industry by storm, I realized that anything I did would have to work with UML.

So I went back to the UML. By now, it was in version 0.91, and a new concept was included: stereotypes. At first, I was clueless to what a stereotype was. The UML specification is not the easiest reading, after all. It was long and difficult, but I knew that any success in the area of modeling Web applications had to come from this direction. Eventually, I started to understand what was meant by stereotyping and the other extension mechanisms: tagged values and constraints. I was finally starting to see light at the end of the tunnel.

I now had a mechanism with which I could introduce new semantics into the UML grammar without disturbing the existing semantics. I always knew that the key was to provide a consistent and coherent way to model Web-specific elements at the right level of abstraction with the models of the rest of the system. The UML extension mechanism provided me with the framework to do so.

The next step was to start defining the extension by creating stereotypes, tagged values, and constraints. For me, the ability to use custom icons in diagrams with stereotyped elements went a long way to ease my concern for intuitive diagrams; also, Rational Rose, my visual modeling tool of choice,[1] had just introduced a way to use one's own stereotypes in Rose models. I quickly created a set of icons for Web page abstractions. I tried to make them consistent, mostly rectangular with the stereotype indication in the upper-left corner. I used filled-in dog ears[2] to represent pages and unfilled dog ears to denote components. Icons without any dog ears typically represented contained classes, which cannot be requested directly by a Web browser. The icon for Web page components is similar to the icon used by the three amigos—Grady Booch, James Rumbaugh, and Ivar Jacobson—in their book, *The Unified Modeling Language User Guide* (Addison Wesley Longman, 1999).

Looking back, I remember spending less than a day to draw up the icons. I didn't spend much time on it then, since I always believed that eventually someone with a little more experience would design some meaningful ones. In the almost two years since then, they have remained essentially the same. I am surprised that I have received absolutely no comments on the style of the icons from the hundred or more people who have been using them. I think that for this version of the extension, the style of icons is going to stick.

As the extension evolved and a lot of the details and inconsistencies were getting corrected, I always kept an eye out for code-generation possibilities. In my mind, the modeling technique could be validated if it were possible, in theory only, to unambiguously generate and reverse engineer code. Since most of my experience was with Microsoft Active Server Pages, I began creating Rational Rose scripts to forward engineer ASP code. I've tailored the scripts to create Java Server Pages code also; from a code structure point of view the two are very similar.

From that point, things proceeded at a tremendous rate. I published a white paper on the Internet and presented the topic at the 1998 Rational User's Conference in Orlando, Florida. Grady Booch took an interest in the work and encouraged me. Addison Wesley Longman asked whether I was interested in expanding the topic into a book. If I had only known how difficult was going to be to write, I'm not sure that I would have agreed. I followed the original white paper with a stream of other articles for both online and print publications and started to get a regular stream of e-mail comments on the extension.

1. All of the sample models used in this effort were developed with Rational Rose. I had worked with the Rose tools for many years prior to this and have recently given up independent consulting to join the Rational team. (My praise of the Rose tool, however, would have been made even if I were not a current Rational employee.)

2. A *dog ear* is the slang term for a bent or folded corner of paper.

By the time this book hits the streets, I will have introduced the topic at five professional conferences and written at least a dozen articles and white papers on the topic. Ideally, this book will continue to propel the recognition that Web application development is a serious topic and one from which we can learn and adopt the successful practices of the past.

Acknowledgments

Authoring a book has always been a dream of mine,.and like most realized dreams, they are never done alone. This book is more than an essay of my experiences with object-oriented web application development; rather, it is the joined effort of a number of individuals and the sacrifices of many close to me.

As a software engineer I never needed to understand the publishing industry. Publishing a book is certainly more than just writing words, as I found out. The team at Addison Wesley Longman—J. Carter Shanklin, Krysia Bebick, and Kristin Erickson—helped me through this process, and I thank them dearly for all their work, encouragement, and gentle reminders of impending deadlines.

This book has been reviewed by a number of people, all of whom I graciously thank for their time, effort, and honesty. Going through those reviews was a humbling experience, yet it was essential in ensuring that this book would be readable and interesting. My thanks go to Ron Lusk, Compaq Computer Corporation, Craig Olague, Grady Booch, Ben P. Ansell, Neil Williams, and Jeffrey Hammond, Rational Software.

I must give extra thanks to Ben Ansell who early on found some of my initial material and worked with me to improve it. Extra thanks also goes to Grady Booch who, after finding my obscure white paper, was kind enough to send a short e-mail message to me saying it was a "great paper." That little note of encouragement helped me put aside nagging fears that what I was doing was wrong and inappropriate.

The writing of this book took much more time that I had originally thought. During the first few weeks of writing, my wife and I welcomed our second child into this world. I can remember many hours writing and working the material in this book when I really just wanted to be cuddling and playing with my two children. There were moments when things got difficult and frustrations high, but through it all I had constant support from Brenda, my wife, without which I would have ever been able to complete this book.

Jim Conallen
August 1999

Foreword

As Jim points out, there's a big difference between building a simple web site and building a web application: the former is relatively static, but the latter is much more dynamic, full of rich content and capable of changing the state of the business as a result of user interaction. A web site is sufficient if all you want to do is publish information, but you really need the scope of a web application for anything else, such as e-business, collaborative content, and distributed communities on the web.

Building a web site is relatively easy, because the barriers to entry are low and development is largely uncomplicated. Building a web application, however, is hard work. Because of the rich content and its importance to the business, you'll have to deal with many different stakeholders, ranging from graphic artists to code warriors to lawyers. Additionally, you'll have to architect your system for continuous change, because a web application that is stagnant is a web application that is dead. If you are webifying an existing client/server system, you'll have to cope with the challenge of integrating legacy. Finally, you'll have to prepare yourself for periods of peak interaction; a system that fails at the most critical moments is one that will seriously harm the business.

You'll find lots of good books that explain how to build web sites. There are also many good books that teach you the details of a specific web technology, such as HTML, XML, EJB, SSL, CGI, TCP/IP, ASP, JSP, and lots of other acronym-enabled things. However, this book is different, for it tells you how to build whole web applications upon these disparate technologies. Building a quality web application in a predictable and repeatable fashion is a team sport, and requires the use of proven development practices such as iterative development, a focus on architecture, and visual modeling. Jim's book addresses all of these practices, and more.

Jim's an experienced web developer, and I'm delighted that he's found this medium to share his experience. I think you'll be delighted as well.

Grady Booch
Chief Scientist
Rational Software Corporation

Contents

Part One

Introduction and Summary of Web-Related Technologies