100%

ONE HUNDRED PERCENT

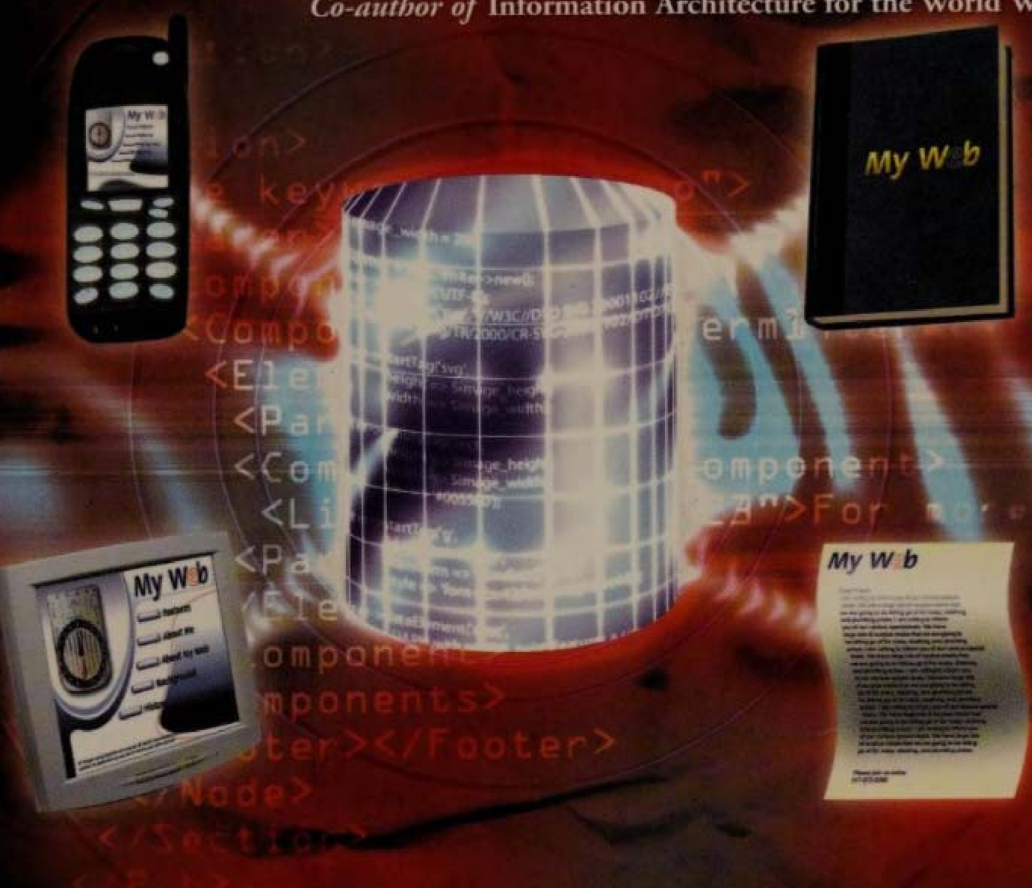
COMPREHENSIVE
AUTHORITATIVE
WHAT YOU NEED

ONE HUNDRED PERCENT

"Broad enough to provide the big picture of an emerging field, deep enough to be practical, and, remarkably, just plain fun to read."

— *Louis Rosenfeld*

Co-author of Information Architecture for the World Wide Web



ame the "information beast" with solutions from a leading CM expert

Design and orchestrate a complete content management system for any organization

Harness XML and database technologies to construct the components of your CMS

Content Management Bible

Companion Web Site
with searchable full text, summaries, discussions, tools, and more!

Bob Boiko

Founder and President, Metatorial Services Inc.

Content Management Bible
Published by
Hungry Minds, Inc.
909 Third Avenue
New York, NY 10022
`www.hungryminds.com`

Copyright © 2002 Hungry Minds, Inc. All rights reserved. No part of this book, including interior design, cover design, and icons, may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording, or otherwise) without the prior written permission of the publisher.

Library of Congress Control Number: 2001093387

ISBN: 0-7645-4862-X

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

1B/RT/RR/QR/IN

Distributed in the United States by Hungry Minds, Inc.

Distributed by CDG Books Canada Inc. for Canada; by Transworld Publishers Limited in the United Kingdom; by IDG Norge Books for Norway; by IDG Sweden Books for Sweden; by IDG Books Australia Publishing Corporation Pty. Ltd. for Australia and New Zealand; by TransQuest Publishers Pte Ltd. for Singapore, Malaysia, Thailand, Indonesia, and Hong Kong; by Gotop Information Inc. for Taiwan; by ICG Muse, Inc. for Japan; by Intersoft for South Africa; by Eyrolles for France; by International Thomson Publishing for Germany, Austria, and Switzerland; by Distribuidora Cuspide for Argentina; by LR International for Brazil; by Galileo Libros for Chile; by Ediciones ZETA S.C.R. Ltda. for Peru; by WS Computer Publishing Corporation, Inc., for the Philippines; by Contemporanea de Ediciones for Venezuela; by Express Computer Distributors for the Caribbean and West Indies; by Micronesia Media Distributor, Inc. for Micronesia; by Chips Computadoras S.A. de C.V. for Mexico; by Editorial Norma de Panama S.A. for Panama; by American Bookshops for Finland.

For general information on Hungry Minds' products and services please contact our Customer Care department within the U.S. at 800-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

For sales inquiries and reseller information, including discounts, premium and bulk quantity sales, and foreign-language translations, please contact our Customer Care department at 800-434-3422, fax 317-572-4002 or write to Hungry Minds, Inc., Attn: Customer Care Department, 10475 Crosspoint Boulevard, Indianapolis, IN 46256.

For information on licensing foreign or domestic rights, please contact our Sub-Rights Customer Care department at 212-884-5000.

For information on using Hungry Minds' products and services in the classroom or for ordering examination copies, please contact our Educational Sales department at 800-434-2086 or fax 317-572-4005.

For press review copies, author interviews, or other publicity information, please contact our Public Relations department at 317-572-3168 or fax 317-572-4168.

For authorization to photocopy items for corporate, personal, or educational use, please contact Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, or fax 978-750-4470.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND AUTHOR HAVE USED THEIR BEST EFFORTS IN PREPARING THIS BOOK. THE PUBLISHER AND AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS BOOK AND SPECIFICALLY DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THERE ARE NO WARRANTIES WHICH EXTEND BEYOND THE DESCRIPTIONS CONTAINED IN THIS PARAGRAPH. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES REPRESENTATIVES OR WRITTEN SALES MATERIALS. THE ACCURACY AND COMPLETENESS OF THE INFORMATION PROVIDED HEREIN AND THE OPINIONS STATED HEREIN ARE NOT GUARANTEED OR WARRANTED TO PRODUCE ANY PARTICULAR RESULTS, AND THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY INDIVIDUAL. NEITHER THE PUBLISHER NOR AUTHOR SHALL BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES.

Trademarks: Hungry Minds and the Hungry Minds logo are trademarks or registered trademarks of Hungry Minds, Inc. All other trademarks are the property of their respective owners. Hungry Minds, Inc., is not associated with any product or vendor mentioned in this book.

Hungry Minds™ is a trademark of Hungry Minds, Inc.

Credits

Acquisitions Editor
Chris Webb

Project Editor
Michael Koch

Technical Editor
Rita Warren

Copy Editors
Bill Barton
Greg Robertson
Mary Kathleen Flynn

Editorial Manager
Colleen Totz

Senior Vice President, Technical Publishing
Richard Swadley

Vice President and Publisher
Joseph B. Wikert

Project Coordinator
Ryan T. Steffen

Graphics and Production Specialists
Sean Decker, Joyce Haughey,
Jill Piscitelli, Betty Schulte,
Brian Torwelle

Quality Control Technicians
Laura Albert, Carl Pierce,
Dwight Ramsey

Proofreading and Indexing
TECHBOOKS Production Services

About the Author

Bob Boiko is a teacher, consultant, writer, programmer, and itinerant businessman. Bob teaches in the iSchool at the University of Washington (`http://www.ischool.washington.edu`) and conducts seminars and lectures as part of his business. He has consulted to a number of the world's top technology and publishing firms, including Microsoft, Boeing, Motorola, and Reed Elsevere. In addition to this book, Bob has written more white papers, articles, and reports than he cares to remember.

Bob began programming in 1977 and has practiced it since (it was always a great way to make money when he was broke). He entered the modern computer age, however, not as a programmer but as a writer. After earning undergraduate degrees in physics and oceanography and a Master's degree in human communication, Bob got a job as a technical writer on contract at Microsoft. Among other projects, he wrote more than half of the MS DOS 5.0 User's Guide and one of Microsoft's first all-electronic User's Guides. From there, he began to develop electronic information systems. In pursuit of electronic information, and then content management, he has created scores of applications and three businesses. He is currently president of Metatorial Services Inc. (`http://www.metatorial.com`).

Credits

[illegible]

About the Author

[illegible]

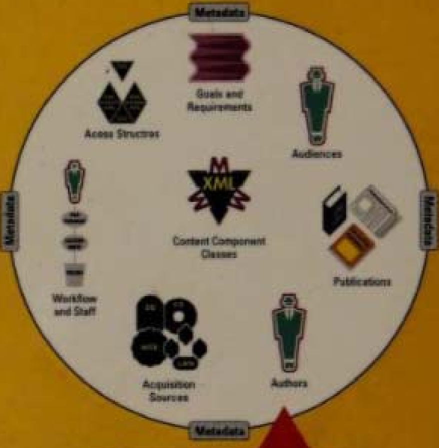
To Laura and the boys for the space

The know-how you need to tame the "information beast". . .

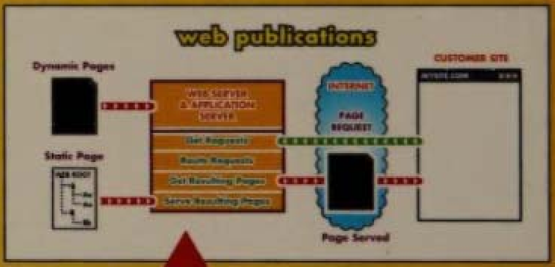
If you're trying to create print and Web publications from the same content, deliver personalized Web content, run a large and complex site, make money in e-commerce, or orchestrate an online community, then content management is the key to your success. In this unique guide, content management guru Bob Boiko explains the theory behind content management — and shows you step by step how to plan and implement your own content management system. From savvy tips on publication templates and workflow models to detailed guidance on how to select the products that are right for you, it's all you need to streamline content gathering, target content delivery — and boost your e-business results.

Inside, you'll find complete coverage of content management

- Find out how content differs from data and information
- Understand CM fundamentals: content collection, management, and publication
- See what it takes to mastermind a CM project, from securing a project mandate to choosing hardware and software products
- Delve into the design of a CM system, from working with metadata and designing content components to devising content access structures
- Discover how to catalog audiences, design publication templates, and account for authors and acquisition sources
- Use XML, XPath, XLink, XSLT, and other markup technologies to prepare your content
- Get hands-on advice for building content collection, management, and publishing systems


Metadata
Goals and Requirements
Access Structres
Audiences
XML
Content Component Classes
Metadata
Metadata
Publications
Workflow and Staff
Acquisition Sources
Authors
Metadata

Understand the entities of content management


web publications
Dynamic Pages
WEB SERVER & APPLICATION SERVER
Get Requests
Route Requests
Get Resulting Pages
Serve Resulting Pages
Static Page
INTERNET
PAGE REQUEST
Page Served
CUSTOMER SITE

Learn how to design a Web-publishing system that can produce a fully dynamic site

Visit the Companion Web Site at www.metatorial.com

www.hungryminds.com

$49.99 USA
$74.99 Canada
£39.99 UK

Reader Level:
Beginning to Advanced

Shelving Category:
Web Development

Preface

I am creating this book because I have to. For more than 10 years, I've been stuffing my head so full of the design, programming, management, and content of information systems that I have to let some out before I can learn anymore. Seriously, from the first time I matched a printed user's guide against the capabilities of Windows 3.0 Help, to the last time I sat with a dot-com client and discussed the impact of massive content management on the architecture of an e-commerce site, I have been living the transition from print to the computer screen. I've seen a ton of technologies and a slew of systems. I've learned enough to know that there is a lot to discuss and figure out. My thinking on what I have experienced has reached some sort of embryonic maturity and is ready to hatch, so here it comes!

My big points are:

- To do e-business, you need the organization and focus that a content management system provides.
- To be useful on the Web and beyond, information must be designed for reuse and must be packaged so that it can be located and automatically organized into targeted publications.
- Content is the information and interactivity that organizations must harness in order to deliver value to their customers.
- Content management systems (CMS) collect, manage, and publish this information and interactivity.
- A CMS is not a CD-ROM that you install, start, and forget about. Rather, it is an ongoing process of knowing your information and your audiences and how to match the two together in a set of publications.

This book attempts to lay a comprehensive foundation under these concepts and create a solid methodology for the practice of content management and, by implication, e-business.

Who Should Read This Book

This book presents a complete model of content management as well as a practical guide to doing content management. It brings together current trends in content management, with my own experience and thinking, into a unified framework. It will be of foremost use to people tasked with designing or implementing content systems (programmers, designers, writers, and managers). It will also be of use to decision makers within an organization who need to develop an e-business or content management strategy.

The people who have the most to gain from this book fall broadly into these categories:

- **Managers** in charge of a content project. These managers need to know how to get a project initiated and completed. The information in Part III, "Doing Content Management Projects," is most directly targeted to managers. In addition, senior level managers in charge of e-business strategy will benefit from this book. These people need to figure out how to do e-business and, then, how to create an organization that can build and run an e-business system.
- **Practitioners** in the areas such as content creation (writers and editors), content administration (database and information managers), and content publishers (site masters, designers, publishers, Web designers). For these people there are relevant examples, methods, and overviews.
- **Programmers**, IT staff, and other technical professionals who need to understand how their skills fit into the broader framework of content management. The material in Part V, "Building a CMS," will be of particular interest to technical folks.
- **Information architects** who want to better understand the nature of a large, organization-wide content management initiative. For these people, Part IV, "Designing a CMS," may be particularly interesting.
- **Project staff**, who consist of the in-the-trenches creative, administrative, or publishing staff who want insight into their own jobs and to understand how their contribution fits into a larger framework.
- **Students** of business, technology, or other disciplines who want to keep up with the latest trends in information management. For them, content management represents a viable future occupation.
- **Business analysts and consultants**, who are called upon to help others figure out a content management strategy. With few standard sources to draw on and a rapidly changing field, those who help others will find help of their own in this book.

I also believe that there is value in this book for anyone else who wants to know what the new age of information management might look like. Even if you are not personally faced with creating a CMS, you may want to know what one is and how one is put together.

Why You Need This Book

To obtain maximum benefit from this book, you need to have felt the pain of too much content and not enough system to handle it. For those with a small site or only a little content to distribute, this work is overkill. For those who have tried and failed to, as we say, "tame the information beast," this book can help. It provides tools that you can use to win the battle next time. For those who are faced with helping to create a large Web site or other publication type for the first time, this book can help you approach the problem in a coherent and reasoned way.

Content management is important. It can underlie today's most significant digital technologies (including, among others, eCommerce, customer relationship management, personalization, advanced Web sites, and electronic communities). By understanding and properly implementing a CMS, organizations will have laid the groundwork on which the rest of these systems can stand. In so doing, they can save a tremendous amount of time and money and can unite these disparate systems with a single, enduring infrastructure.

This book matters for two additional reasons.

- ✦ The field of content management is in its infancy. I believe that this book helps define it. In my work I daily experience the confusion and frustration of people who need to define or implement a content management system for their organization and do not know how to approach it. These people are being bombarded by product-centered white papers and superficial ad-speak that present an all-too-simple picture. In contrast, this book provides a thoroughgoing and impartial framework upon which to base an understanding of the problems and the solutions of content management.
- ✦ There is a definite need for the kind of practical knowledge this book provides. The processes and practices that I, along with my colleagues, have developed can be of great use to people who need to implement and staff a content management system. From job descriptions to conversion code samples, you will find a good supply of methodologies, pointers, and insights. This practical knowledge, woven into an overall framework for implementing a content management system, should provide a powerful resource to anyone needing to understand or do content management.

When I began speaking on the subject of content management, my audiences consisted mostly of writers, marketing people, managers, editors, and librarians who were tasked with putting together a large Web site. Most had in-depth knowledge in their respective disciplines and some experience creating Web sites, but little resource for tackling the job they had taken on.

Today, my audiences consist of much the same people, but now they have job titles like Content Manager, Director of e-Business Strategy, and Chief Information Officer. In addition to creating an Internet presence, many are being tasked with developing an entire enterprise system for controlling the creation and dissemination of information. They have bigger titles and greater responsibility but few extra tools to help them meet these new responsibilities.

How This Book Is Organized

This book consists of five parts.

Part I: What Is Content?

Content is information and functionality that has been harvested and organized toward some particular use. In this section, I dive into this definition and describe what is meant by organization and use. I use this definition to show why content is just information but also more than information.

Part II: What Is Content Management?

Today, many people see content management as a way to create Web sites. As I present it, content management is a much broader process of collecting, managing, and publishing information to whatever medium you need. In this section, I present a full definition of content management and a complete model that you can use to understand and scope your own content management problems and solutions.

Part III: Doing Content Management Projects

This section describes content management from the manager's perspective. Given my model of content management, I turn to the practical problem of how you go about creating a CMS. Unfortunately, it is not simply a matter of buying a suitable product, installing it, and turning it on. No CMS you create or buy will tell you what content your organization should manage or why. Rather, doing content management is a process of getting your organization behind your project, designing all of the details behind the system, selecting an appropriate system, and finally implementing it.

Part IV: The Logical Design of a CMS

This section describes content management from the designer's perspective. Usually called information architect, the designer's job is to study and develop an interconnected system of collection, management, and publishing. The heart of this analysis is a metadata framework that ties all content together. In this section, I talk about this process in depth. I detail the exact sorts of information you have to amass to fully understand the system you want to create. I break the information to be collected down into a set of interrelated entities that rest on what I call the wheel of content management.

Part V: Building a CMS

This section describes content management from the perspective of the builder. Generally, CMS builders are content handlers, programmers, and information technology professionals. This section has information of interest to all three types of builders. For content handlers (who convert and tag content), it provides an under-the-hood picture of how content is constructed and how it can be processed. For programmers, it provides details on constructing publishing templates and programming content conversion systems. For IT professionals, it describes the software and hardware behind a CMS and how you might decide what is right for your organization.

The Companion Web Site

A complete version of the content of this book is available online at `www.metatorial.com`. The Web site is generated from the same XML content base as was this book. Using the content from the book and the features of HTML and the Web, the site includes:

- **Active summaries** that let the reader quickly scan through the work and get the main concepts. These concepts appear above the content they summarize. The user can browse the site by sections (in the usual manner) or by concepts. Quick links move the reader effortlessly from one mode to others.
- **Photographs and audio interviews** with industry leaders, practitioners, academics, and organizational leaders. The interviews are tied to particular concepts in the work and are edited into 5-minute sections.
- **Audio commentary** from the author on the concepts in the work. The commentary is in the form of conversations and monologues of not more than 5 minutes each.
- **Slide presentations** from the major sections of the work. These presentations abstract and package the main concepts of the work.
- **Code samples and sample databases** that illustrate the techniques I discuss.
- **Alternative forms of key sections** of the book that you can download and use in your own CMS analysis.

The Web site companion to this book, which is accessible only to owners of the book by entering specific facts from the book, will do what no printed book can—it will grow and change based on what I learn from readers of this book.

Conventions Used in This Book

The book uses a variety of conventions to help you scan through it and quickly find information of interest. Most of the code samples in the book are either HTML or XML. In both of these sorts of code, I use upper case letters for tag names and upper and lower case letters for tag attributes. In XML the capitalization of tag names really matters (it is a case-sensitive language). So, in a few cases, I break my all upper case convention where the sample code needs to be consistent with a system that uses a different convention.

Here are some of the other conventions you will see in the book:

Note This icon presents a quick aside to the general topic. You will find information of particular note or important caveats to the current discussion.

Tip This icon presents a short idea that you may want to implement as part of your own content management system design or implementation. There is advice, of course, throughout the book, but these paragraphs highlight "advice bytes" that you can immediately use.

Cross-Reference This icon introduces a link between the current discussion and another that is related to it. Cross-references also appear within sentences to more closely link them to the ideas to which they apply.

This Is a Sidebar

Sidebars present an extended aside. They contain stories and other devices that give you another perspective or angle on the discussion at hand. My sidebars are often written in a lighter tone and emphasize some of the quirkier aspects of content management.

Acknowledgments

Rita Warren stands out in my mind as the person next most responsible for this book. In 1997, Rita was the first person to look at these ideas and say, "There is something here." She served for years as chief editor, interpreter, and refiner of my ideas (many of which are actually hers). Even up to her amazing technical edit of the book itself, Rita has stuck with it and believed but always questioned. Thanks.

Michael Koch of MK Publication Services served as lead editor and sanity warden of the book process. The book would not have made it without his patience, perseverance, and wit. Chris Webb of Hungry Minds found me a man with a half-finished book and turned me into an author. His vision for a book on content management has helped guide the project home. Finally, I'd like to extend my thanks to the copy editors and production staff at Hungry Minds for making my prose consistent and bringing to life my graphics and putting up with my last-minute changes.

Most of this material was developed during my years as President and then Chief Technology Officer at Chase Bobko Inc. Without the support of my business partners Jayson Antonoff and Patricia Chase and my life partner, Laura McCormick, I'd still be on Chapter 1. The staff of Chase Bobko deserves my hearty thanks for putting up with my interminable lectures and continually changing CMS methodologies. In particular, our marketing group helped to build an identity and set of CMS offerings that forced me to think and think again. Our product group put flesh on many of my skeletons and gave them life. Our IA and consulting staff went forth into the world to solve real problems with ideas that were not always fully cooked. In that vein, I'd also like to thank my many clients who over the years have seen the value in this approach and have supported it through its many twists and turns.

I would also like to thank PLAN, an international, child-focused, development organization with no religious, political or governmental affiliations. PLAN gave kind permission to include screen shots from their web site at www.plan-international.org. All other examples relating to PLAN have been created by the author and are included for illustrative purposes only and are not necessarily an accurate representation of how PLAN has developed or structured its websites.

Finally, I'd like to thank Mike Eisenberg and the staff of the University of Washington iSchool for their confidence and advice, and their encouragement of this project.

Contents at a Glance

Preface ix
Acknowledgments xv
Introduction xxxiii

Part I: What Is Content? 1
Chapter 1: Defining Data, Information, and Content 3
Chapter 2: Content Has Format 13
Chapter 3: Content Has Structure 21
Chapter 4: Functionality Is Content, Too! 31
Chapter 5: But What Is Content Really? 41

Part II: What Is Content Management? 63
Chapter 6: Understanding Content Management 65
Chapter 7: Introducing the Major Parts of a CMS 81
Chapter 8: Knowing When You Need a CMS 111
Chapter 9: The Roots of Content Management 129
Chapter 10: The Branches of Content Management 151

Part III: Doing Content Management Projects 175
Chapter 11: Staffing a CMS 177
Chapter 12: Working within the Organization 205
Chapter 13: Getting Ready for a CMS 231
Chapter 14: Securing a Project Mandate 251
Chapter 15: Doing Requirements and Logical Design 265
Chapter 16: Selecting Hardware and Software 303
Chapter 17: Implementing the System 367
Chapter 18: Rolling Out the System 401

Part IV: The Logical Design of a CMS 427
Chapter 19: The Wheel of Content Management 429
Chapter 20: Working with Metadata 453
Chapter 21: Cataloging Audiences 485
Chapter 22: Designing Publications 507
Chapter 23: Designing Content Components 583

Chapter 24: Accounting for Authors 613
Chapter 25: Accounting for Acquisition Sources 633
Chapter 26: Designing Content Access Structures 651
Chapter 27: Designing Workflow and Staffing Models 689

Part V: Building a CMS 723
Chapter 28: What Are Content Markup Languages? 725
Chapter 29: XML and Content Management 743
Chapter 30: Processing Content 767
Chapter 31: Building Collection Systems 803
Chapter 32: Building Management Systems 839
Chapter 33: Building Publishing Systems 887

Epilogue 921
Index 925

Contents

Preface ix
Acknowledgments xv
Introduction xxxiii

Part I: What Is Content? 1

Chapter 1: Defining Data, Information, and Content 3

- Content Is Not Data 3
- Content Is Information Put to Use 6
- Content Is Information Plus Data 9
- From Data to Content and Back 10
- Summary 11

Chapter 2: Content Has Format 13

- Binary Format: Storing Information 13
- Rendering Format: Presenting Information 14
- Dealing with Formatting 15
- Categorizing Formatting 17
  - Formatting for effect 17
  - Formatting by type of effect 18
  - Formatting by scope 19
- Summary 20

Chapter 3: Content Has Structure 21

- Structure Is Important 21
- Structure Can Be Difficult to Create 23
- How to Categorize Structure 26
  - Structure by purpose 26
  - Structure by type 26
  - Structure by scope 28
- Summary 28

Chapter 4: Functionality Is Content, Too! 31

- What Is Functionality? 32
- Monolithic vs. Mix-and-Match Functionality 32
- Functionality Interspersed with Information 34

Managing Functionality Like Information 35
    Publishing functionality on the Web 37
    Where information and functionality diverge 38
Summary 39

Chapter 5: But What Is Content Really? 41

Content, Context, and Meaning 41
Content Organization Starts with Purpose 45
Content Is Named Information 47
From Data to Wisdom 49
    Data is raw and discrete 49
    Data is nondiscursive and out of context 51
    Information is processed and continuous 53
    Information is discursive and full of context 54
    Knowledge and wisdom can be information 56
Why Does Text Get All of the Attention? 59
    Text is codified language 60
    Text has the lead 61
Summary 62

Part II: What Is Content Management? 63

Chapter 6: Understanding Content Management 65

Defining Content Management 65
The Content Management Industry 67
Systems for Managing Content 68
    The nominal Web CMS 68
    The dynamic Web site 73
    The full Web CMS 76
The Enterprise CMS 77
Summary 79

Chapter 7: Introducing the Major Parts of a CMS 81

A CMS Overview 82
The Collection System 83
    Authoring 84
    Acquiring 85
    Converting 88
    Aggregating 89
    Collection services 96
The Management System 97
    The repository 98
    The administration system 101
    The workflow system 102
    Connections 103

The Publishing System 104
    Publishing templates 104
    Publishing services 106
    Connections 106
    Web publications 107
    Other publications 108
Summary 109

Chapter 8: Knowing When You Need a CMS 111

Gauging the Amount of Content 112
Managing the Size of the Contribution Base 113
Anticipating the Amount of Change 116
Knowing the Number of Publications 118
Estimating Complexity 121
    Vacation company "A" 123
    Vacation company "B" 124
    Vacation company "C" 125
    Evaluating your results 126
Summary 127

Chapter 9: The Roots of Content Management 129

The Knowledge Base of the Publishing Industry 129
    The publication 130
    Content collection 130
    Abstracting the process 131
The Principles of Document Management 132
    Files contain content 132
    Files store BLOBs 133
    Publications are files 134
    Document is an empty word 135
    Document management versus content management systems . . . 135
The Limits of IT Groups 137
    IT departments have avoided the Web 137
    Content begins where IT groups leave off 138
The Legacy of the Multimedia Industry 139
    Electronic publications 140
    End-to-end content management 140
The Challenges of Documentation Projects 141
    Huge information bases 141
    Simultaneous publications 142
The Traditions of Library and Information Science 143
    Categorization and information seeking 143
    The structure of information 144
The Technology of Software Development 145
    Collection technologies 145
    Management technologies 145
    Publishing technologies 146
    The functionality in electronic publications 146

The Premises of Marketing 148
Summary 149

Chapter 10: The Branches of Content Management 151

Personalization 152
  What is personalization? 152
  Content management underlies personalization 153
Advanced Web Sites 155
Multiple Publications 157
E-commerce 161
  Catalogs and content management 162
  E-commerce functionality and content management 163
Knowledge Management 164
  What is knowledge management? 164
  Knowledge is content to manage 165
Online Communities 166
  What is a community? 166
  How are online communities constructed? 166
Summary 172

Part III: Doing Content Management Projects 175

Chapter 11: Staffing a CMS 177

About the CMS Jobs 177
Managers 179
  Content manager 179
  Project manager 180
  Production manager 181
Business Analysts 182
Information Architects 184
  Content analyst 185
  Metator 186
Infrastructure Staff 187
  CMS administrator 187
  Deployment analyst 189
  Trainer and documentation specialist 190
Software Developers 190
  Software analyst 190
  Template and CMS developer 192
  Custom application developer 192
  Software integrator 193
  Test analyst and test engineer 194
Publications Staff 194
  Publication analyst 195
  Publication designer 196
  Page developer 197
  User interface specialist 198

Content Processing Staff . . . 198
    Conversion analyst . . . 198
    Tool developer . . . 199
    Content processor . . . 200
    Content QA specialist . . . 201
Content Creation Staff . . . 201
    Acquisitions manager . . . 201
    Traffic cop . . . 201
    Writers and other content creators . . . 202
    Editor . . . 202
Summary . . . 203

Chapter 12: Working within the Organization . . . 205

Content Management and the Organization . . . 205
Tracking Information Flow in the Organization . . . 208
    Understanding your information . . . 211
    Understanding your functionality . . . 212
Understanding Organizational Roles . . . 213
    Business units generate value . . . 213
    Editorial teams unify content . . . 213
    Marketing teams direct and unify publications . . . 214
    IT groups build and maintain infrastructure . . . 214
Exploring Organizational Models . . . 215
    Collection variations . . . 216
    Publishing variations . . . 217
    Management variations . . . 220
Using Functional Collection and Publishing . . . 221
    Organizing collection systems and publications by type . . . 221
    Creating functional teams . . . 223
Identifying Your CMS Hurdles . . . 226
Summary . . . 229

Chapter 13: Getting Ready for a CMS . . . 231

Understanding the CMS Project Process . . . 231
Techniques for Getting the Job Done . . . 234
    Start with the project team . . . 234
    Look for pain in the organization . . . 237
    Assess the current mandate . . . 238
    Assess the organization's assumptions . . . 238
Taking Stock of the Deliverables . . . 241
    The readiness assessment . . . 241
    Document inventory and analysis . . . 243
    The "State-of-the-Content System" Report . . . 245
    The education plan . . . 246
    A preliminary project plan . . . 247
    A risk assessment . . . 248
Summary . . . 248

Chapter 14: Securing a Project Mandate . . . 251

- What to Agree On . . . 251
- Techniques for Getting the Job Done . . . 252
  - Recognize sponsors . . . 252
  - Learn about your sponsors . . . 254
- Taking Stock of the Deliverables . . . 255
  - Sponsor profiles . . . 255
  - A hierarchy of issues . . . 256
  - Notes and minutes . . . 257
  - The mandate process . . . 258
  - The project mandate . . . 261
- Summary . . . 264

Chapter 15: Doing Requirements and Logical Design . . . 265

- What Are Requirements? . . . 265
- What Is Logical Design? . . . 266
- Techniques for Getting the Job Done . . . 268
  - Gathering requirements . . . 268
  - Doing logical design . . . 274
- Taking Stock of the Deliverables . . . 282
  - The requirements plan of attack . . . 283
  - The requirements document . . . 284
  - The design plan of attack . . . 285
  - The collection design document . . . 286
  - The management design document . . . 287
  - An audience analysis . . . 289
  - The publication design documents . . . 290
  - A localization plan . . . 294
  - A risk assessment plan . . . 295
  - A revised project plan . . . 296
  - An executive summary . . . 298
- Taking Stock of Your Staffing Needs . . . 298
- Summary . . . 301

Chapter 16: Selecting Hardware and Software . . . 303

- The Product Paradox . . . 303
- Deciding Whether to Build or Buy . . . 310
- Techniques to Get the Job Done . . . 312
  - How to select decision makers . . . 312
  - How to select a product . . . 313
- Taking Stock of the Deliverables . . . 326
  - Product files . . . 327
  - The selection criteria and RFP . . . 327
  - The score card . . . 328
  - The design diagrams . . . 328
  - The decision report . . . 329
  - Risk assessment update . . . 329
  - Project plan update . . . 331

Taking Stock of Your Staffing Needs 331
Sorting Through the CMS Selection Criteria 332
    Business criteria 333
    Overall criteria 334
    Collection criteria 341
    Management criteria 347
    Publishing criteria 356
Summary 364

Chapter 17: Implementing the System 367

Looking at the Process So Far 368
Looking at the Project So Far 369
Techniques for Getting the Job Done 370
    Cutting back 370
    Considering one project vs. many projects 371
    Detailing the implementation process 373
Taking Stock of the Deliverables 388
    Collection specifications 390
    Management specifications 392
    Publication specifications 393
    The project plan 396
    The risk assessment plan 398
Taking Stock of Your Staffing Needs 399
Summary 400

Chapter 18: Rolling Out the System 401

What Is Deployment? 401
Techniques to Get the Job Done 402
    Creating documentation 402
    Doing training 410
    Powering up the system 413
    Preparing content 418
    Revising the system 418
Taking Stock of the Deliverables 420
    The deployment specification 420
    The staffing plan 421
    The training plan 422
    The documentation plan 423
    The maintenance plan 424
Summary 425

Part IV: The Logical Design of a CMS 427

Chapter 19: The Wheel of Content Management 429

The Content Management Entities 429
  Introducing the CMS wheel 431
  Entities add a layer of abstraction 433
  Goals and requirements 434
  Audiences 437
  Publications 439
  Component classes 441
  Authors 444
  Acquisition sources 445
  Access structures 447
  Workflow and staffing 448
Think, Plan, Integrate 449
PLAN International—An Example Organization 450
Summary 451

Chapter 20: Working with Metadata 453

What Is Metadata? 453
  What does meta mean? 453
  What does metadata mean? 455
  What does metatorial mean? 457
  The narrow view of metadata 459
  The wide view of metadata 460
  Metadata and content management 460
Understanding the Types of Metadata 461
  Structure metadata 462
  Format metadata 465
  Access metadata 467
  Management metadata 468
  Inclusion metadata 469
Categorizing Metadata Fields 472
Introducing the Metatorial Framework 474
  Metatorial processing 475
  The metator 477
  The metatorial guide 478
Localizing Metadata 482
Summary 483

Chapter 21: Cataloging Audiences 485

Serving vs. Exploiting an Audience 485
What Is an Audience? 486
  Audiences and communicators 487
  Audiences and marketing 488
  Audiences and users 489
  How many audiences do you have? 489

Audiences and Localization 491
  What is localization? 492
  What are your localities? 492
  Audiences and locality 494
  What gets localized? 494
  Localization and content management 496
An Example Audience Set 497
Analyzing Audiences 498
  Think 498
  Plan 499
  Integrate 505
Summary 506

Chapter 22: Designing Publications 507

What Is a Publication? 507
  Publication purpose 509
  Publishers 510
  Audiences 511
  Messages 512
  Authorship 515
  Publication format 516
  Publication structure 517
  Publications can be good or bad 518
Publications and Templates 519
  Bridging worlds 520
  Mixing the static and the dynamic 522
  Building pages 525
  Creating a system of publications 528
  Using a template processor 529
  Understanding template logic 531
  Using templates within templates 532
  Web templating 534
  Print templating 542
  Fax templating 545
  E-mail templating 546
Publications and Personalization 546
  Personalization and the audience 547
  Personalization and components 548
  Personalization and rules 549
  Personalizations in templates 552
  Customization vs. personalization 553
  Dynamic and static personalization 554
Analyzing Publications 555
  Think 555
  Plan 556
  Integrate 561

Analyzing Templates 562
    Think 562
    Plan 563
    Integrate 569
Analyzing Personalization 570
    Think 570
    Plan 571
    Integrate 581
Summary 582

Chapter 23: Designing Content Components 583

What Are Components? 583
    A supermarket of content 584
    The basic unit of content management 585
    Components are like objects 586
    How do you divide content? 587
    Components vs. pages 588
    Components have classes and instances 589
    Locality in components 591
Components Have Elements 592
    Elements have types and values 593
    Unique identification 593
What Do Components Look Like? 594
    Components in flat files 594
    Components in structured files 594
    Components in relational databases 595
    Components in object databases 596
The Relationship between Functionality and Components 597
    A functionality component 598
    Functionality classes and instances 601
Analyzing Components 602
    Think 603
    Plan 603
    Integrate 611
Summary 612

Chapter 24: Accounting for Authors 613

What Is Authoring? 613
    Who can become an author? 614
    Harvesting creative product 614
    Authoring tools 615
    Authoring prior to the CMS 615
The Author's Attitude 616
    Changing the author 616
    Changing the content 618

Your Attitude Toward Authors 618
    Technical savvy 619
    Influence 621
    Incentives 622
    A savvy/influence matrix 623
Analyzing Authors 624
    Think 624
    Plan 625
    Integrate 631
Summary 632

Chapter 25: Accounting for Acquisition Sources 633

What Is Acquisition? 633
    Found sources 634
    Syndicated content 635
    To syndicate or connect? 637
    Acquiring functionality 638
Is It an Author or an Acquisition Source? 640
Analyzing Sources 641
    Plan 642
    Integrate 649
Summary 650

Chapter 26: Designing Content Access Structures 651

Understanding Access Structures 651
    Publication navigation vs. CMS access structure 654
    The content domain 655
    Hierarchies 659
    Indexes 663
    Cross-references 666
    Sequences 671
    Full-text search 673
Analyzing Access Structures 674
    Think 675
    Plan 675
    Integrate 687
Summary 688

Chapter 27: Designing Workflow and Staffing Models 689

Understanding Workflow 689
    Workflow triggers 694
    Workflow objects 695
    A perspective on workflow 697
    Push and pull workflow 699

Introducing Tasks, Jobs, and Steps 699
  Staff modeling 701
  Staff, jobs, and tasks 702
  Task-time calculations 705
  Staffing calculations 706
  Fudge factors 707
Fitting Localization into Your Workflow 708
Analyzing Workflow 710
  Think 711
  Plan 711
  Integrate 717
Analyzing Staffing 717
  Think 717
  Plan 718
  Integrate 721
Summary 721

Part V: Building a CMS 723

Chapter 28: What Are Content Markup Languages? 725

A Brief and Selective History of Markup Languages 725
What Is a Markup Language? 728
A Taxonomy of Markup Languages 729
  ASCII vs. binary 730
  Format vs. structure 731
  Extendable vs. nonextendable 732
  Range of coverage 733
Working with Markup 733
  Anatomy of a tag 734
  The language vs. the interpreter 735
  Representing representation 736
  Don't be baffled by syntax 736
  The concept of nesting 738
  The benefits of white space 739
  People play in the margins 741
Summary 742

Chapter 29: XML and Content Management 743

What Is XML? 743
  XML and data interchange 744
  XML tagging 744
  Management by DTD 749
  Adding formatting 751
Using XML in Content Management 753
  XML in collection 753
  XML in management 754

    XML in publishing 755
    XML in integration 756
    Help from the rest of the XML gang 756
  Programming in XML 757
    Who needs to know XML? 758
    Introducing DOM XML programming 759
  Summary 766

Chapter 30: Processing Content 767

  What Is Content Processing? 767
  Content Processing and the CMS 770
    Focusing on the long-term benefits 771
    Focusing on the short-term benefits 774
    Distilling the essence of a process 775
    Bringing people into processing 775
    Tracking the master copy 777
  Managing the Processing Project 778
    Taking stock of the content inventory 779
    Drafting the processing specification 780
    Defining testing methods 781
  Getting to the Core of Content Mechanics 782
    Understanding the principles of mapping content 784
    Converting content from Microsoft Word to XML 789
  Summary 800

Chapter 31: Building Collection Systems 803

  Minimizing Disruption 803
  Forms, Files, and Batch Processes 804
    Web-based forms 806
    Updates and deletes 809
  Selecting Collection Tools 809
    Authoring tools 809
    Acquisition tools 811
    Conversion tools 812
    Aggregation tools 813
  Doing Collection Physical Design 814
    Authoring considerations 816
    Acquisition considerations 821
    Aggregation considerations 823
  An Example Collection System 827
    Input and output 828
    XML authoring 828
    Showing the hierarchy 831
    Showing the content 832
    Showing the allowed content 834
    From example to reality 836
  Summary 838

Chapter 32: Building Management Systems 839

What's in a Management System? 839
Building a Repository 840
  Essential and recommended repository functions 841
  The content model 843
Storing Content 844
  Relational database repositories 846
  XML-based repositories 864
  File-based repositories 875
Implementing Localization Strategies 877
Doing Management Physical Design 879
  A repository-wide DTD 880
  Link checking 880
  Media checking 881
  Search and replace 882
  Management integrations 884
Summary 884

Chapter 33: Building Publishing Systems 887

Detailing the Parts of a Publication System 887
  Publication platforms 888
  Publication templates 889
  A full programming language 890
Publication Gotchas 891
Mapping Access Structures to Navigation 895
  Mapping hierarchies 896
  Mapping indexes 899
  Mapping cross-references 902
  Mapping sequences 905
An Example Publication System 908
  About XSLT 910
  The book 912
Summary 919

Epilogue 921

Index 925

Introduction

When I began working on content management, the terms "content management" and "e-business" did not exist. Two years ago, when my organization decided to strategically pursue content management as a focused practice, only a handful of software companies and industry analysts were using the terms "e-business" and "content management." Today, content management is a fairly well-recognized product and industry category. Dozens of products claim to do content management, and all of the major analyst groups (Gartner, Forrester, and the like) have projected major investments in this area and a growing commercial sector. For example, Goldman Sachs says that content management software sales will reach $4.5 billion by 2002. In addition, professional services for Internet technologies is growing at 29 percent per year and is projected to reach $56 billion by 2005.

Interestingly, I believe that there is nothing new in content management and at the same time it is completely new. Most of the concepts that I build upon are from other fields. On the other hand, by bringing these ideas together, a new way of thinking about content and publishing emerges. For example, you need to know your audiences to do content management. This is not a new idea. Writers, marketers, and even computer programmers have been doing audience analysis in one form or another for a long time. On the other hand, by combining the way these three groups have looked at audiences and applying it to the very particular needs of personalization in a content management system, I can create an overall concept and practice of audiences that goes beyond what any of these disciplines has done. The results are not simply conceptual, they are a practical set of questions to answer and data to collect to make your audiences an integral part of your content management system implementation.

This book is my response to the questions I have been repeatedly asked. What I hear is, "This is so different I don't even know how to approach it" and "It's at a level of complexity far beyond what we have ever done before" and "This is so new that we can't find anyone with enough experience to pull it off." What I see most often is that people try to use their old methods and understanding to tackle a new problem, and it does not work. This book attempts to propose and detail a new approach that borrows heavily form older disciplines but forms a new discipline around the new needs of large-scale information creation, management, and publishing.

Three Good Reasons to Do Content Management

I see three major ways to enter into the ideas of content management. I'd like to present them to help you understand why content management is so important. These onramps are the following:

- **Content management gives flesh and bones to today's notions of e-business.** If e-business is the process of reaching your constituents with the right information and transactions at the right time, then content management is the way to make an e-business real and workable.
- **Content management is an antidote to today's information frenzy.** Web sites are getting out of control; we are expected to harvest good information from wider and wider contributor bases; and we are expected to make information entirely reusable so that it can be distributed at anytime to anyone. Content management can help you organize and direct your information to keep it under control.
- **Content management addresses one of the key unanswered questions of the coming Information Age:** "How is it possible to give particular value and substance to a 'piece' of information?" Content management systems create and manage pieces of information and tag them with all the information you might need to figure out what they are worth.

Content Management Underlies E-business

A newly created magazine proclaims in its advertisements: "Forget everything you know about business. It's all changed now." The assumption, of course, is that the basic fabric of business has changed. This is a recent but widespread belief fed by the fantastic appeal of the new information technologies. Before the Web, there was just business. Although processes and technology changed, it was still possible to know and understand how to *do* business. Lately, as the capability to do business over the Web has become a reality, many organizations have lost their balance and have the uneasy feeling that the rules have changed somehow. They have been led to believe that the new capabilities of the Web alter the basic nature of business.

I don't believe this is true. Rather, I side with the client of mine who recently said, "Someday there will just be business again. We will just happen to do it electronically." I believe that the basic rules of business have not changed. Know your customer, project a good image, and provide real value are as true today as they were in the markets and bazaars of a thousand years ago. Electronic business is simply the latest way to put the basic rules into action.

Tip

I've seen many strong businesspeople forget what they know as they wrestle with what they don't know. By remaining centered on the parts of business that do not change, you can retain a sharp ability to ask the right questions and appropriately judge the kaleidoscope of technologies that will pass in and out of focus in the coming years.

What is e-business?

In a nutshell, e-business is the process of delivering any part of your business to any audience wherever it is. E-business does not change the basis of business, but it does change the practice of doing business. In particular, e-business brings a quantitative change in the following aspects of business:

- **Ubiquity:** Parts of your business can appear on any computer screen anywhere there is connectivity. Time and space are no longer barriers to conducting business.
- **Depth:** You can deliver as much detail and background as you can manage to create in a convenient and easy-to-consume fashion. Rather than racks full of catalogs and technical documentation, a simple URL is all that is needed to fully detail your organization's offerings.
- **Speed:** The slowness of your human processes is the only necessary delay between the creation of information and its general availability. Second-by-second changes to information are now possible and entirely practical.
- **Personalization:** Your ability to understand and serve your audiences is the only impediment to tailoring messages and offerings that are as individualized as you want them to be.

To explore further, I'll drill through the definition of e-business one piece at a time.

The process of doing business

E-business is not a set of static outputs but rather a dynamic "best fit" of your practices to the environment inside and outside your organization. Because the technology is continually shifting, it is not possible to stand still. Because your organization, its audiences, information, and business practice are constantly shifting, you had better not stand still.

The technologies behind e-business are new and will change. Applications for customer relationship management, campaign management, self-service procurement, vendor relations, automated fulfillment, and a host of others are barely out of their first versions. In addition, the underlying infrastructure on Web servers is also in flux. Finally, new delivery platforms (such as Web-enabled phones) are appearing all the time. What you can do is changing so rapidly that you have no choice but to consider e-business a process that is never complete.

Tip

I've seen many strong businesspeople forget what they know as they wrestle with what they don't know. By remaining centered on the parts of business that do not change, you can retain a sharp ability to ask the right questions and appropriately judge the kaleidoscope of technologies that will pass in and out of focus in the coming years.

What is e-business?

In a nutshell, e-business is the process of delivering any part of your business to any audience wherever it is. E-business does not change the basis of business, but it does change the practice of doing business. In particular, e-business brings a quantitative change in the following aspects of business:

- **Ubiquity:** Parts of your business can appear on any computer screen anywhere there is connectivity. Time and space are no longer barriers to conducting business.
- **Depth:** You can deliver as much detail and background as you can manage to create in a convenient and easy-to-consume fashion. Rather than racks full of catalogs and technical documentation, a simple URL is all that is needed to fully detail your organization's offerings.
- **Speed:** The slowness of your human processes is the only necessary delay between the creation of information and its general availability. Second-by-second changes to information are now possible and entirely practical.
- **Personalization:** Your ability to understand and serve your audiences is the only impediment to tailoring messages and offerings that are as individualized as you want them to be.

To explore further, I'll drill through the definition of e-business one piece at a time.

The process of doing business

E-business is not a set of static outputs but rather a dynamic "best fit" of your practices to the environment inside and outside your organization. Because the technology is continually shifting, it is not possible to stand still. Because your organization, its audiences, information, and business practice are constantly shifting, you had better not stand still.

The technologies behind e-business are new and will change. Applications for customer relationship management, campaign management, self-service procurement, vendor relations, automated fulfillment, and a host of others are barely out of their first versions. In addition, the underlying infrastructure on Web servers is also in flux. Finally, new delivery platforms (such as Web-enabled phones) are appearing all the time. What you can do is changing so rapidly that you have no choice but to consider e-business a process that is never complete.

Note

With or without the new electronic medium, business is best done dynamically. Regardless of the way you conduct business, you are best served by continually reassessing your audiences, your products, and the ways you present your products to your audiences. As any one of the three changes, the other two must adjust to make use of new possibilities and jettison things that no longer work. Thus, business with no "e" before it is itself is a process that continually shifts.

To do e-business, you create a system for production, not a particular product. You focus more on how you will constantly readjust to the changing landscape of technology, your audiences, information, practices, and publications than on any particular configuration that you might happen to be in right now.

Tip

Recognizing that business is a process, the wise organization will create the right feedback channels so that your audiences can help you continually reexamine what information and functionality you are collecting and managing and what publications serve your goals and their goals most fully.

Delivering any part of your business

E-business is the process of electronically delivering the right parts of your business at the right time. Generally, you deliver your business through computer networks and onto someone's computer screen. The essence of e-business, however, is not in the particular way in which you deliver your business, but in the fact that the essential parts of your business are stored and managed digitally and can be delivered in whatever medium you desire. By digitizing the information and services your organization provides, you make it available for delivery over digital or nondigital channels. Your own Internet site is only one of these channels. Others include the following:

- Your intranet site and any extranet sites you host to communicate with partners
- The Web sites of other partner and promoter organizations
- Kiosks and other offline digital publications
- Print publications that include shared information and "paper functionality"
- Screens for devices such as Web-enabled phones and personal digital assistants (PDAs)
- Personalized broadcast e-mail messages

What are the parts of your business? Loosely, you can break these parts into information and interactions, as follows:

- **Information** is the text, sound, image, and motion that communicate what you would like to convey to your constituents.

- **Interactions** are the capabilities you would like to project, such as buying product, asking for help, contacting an individual in the organization, or participating in an organization-sponsored discussion. Interactions are pieces of functionality that your organization can digitize and treat just the same as information.

To do e-business, the wide and possibly unorganized information and interactions that are key to your organization must be identified, digitized, and segmented into useful chunks so that they can be delivered individually to those who need them—customers, members, staff, partners, or constituents.

Cross-Reference

In Chapter 4, "Functionality Is Content, Too!" I focus on interaction. More aptly called functionality, interaction is the user interface and computer code that gives audiences the ability to communicate back to the organization in order to carry on their business with the organization. In Chapter 21, "Cataloging Audiences," I talk a lot more about audiences.

Reaching your audiences wherever they are E-business means being present and ready to interact with someone wherever they are when they want to interact with you. Today, the overwhelming majority of organizations are focused on their Web site as the place through which to communicate with the outside world. They see the Web as *the* medium for communication, and they want their site to be a prime location on the Net. The assumption is that if you build it and it is good, they will find you and come. With enough marketing or public relations, this might be true. But for the vast majority of organizations, getting people to find them and come is a tricky proposition. Undoubtedly, each organization needs a Web site that has its official and comprehensive information. The essence of e-business, however, is the ability to go to your audiences rather than wait for them to come to you.

As one example among many, consider a government organization that wants to communicate new regulations to its offices. They will certainly want to list the new regulations completely on their Web site. But are there other ways they could reach their intended audiences (staff members, say, in many locations)? Rather than expecting that staff will come to the Web site, the organization could send targeted e-mail messages to the staff, highlighting just the relevant sections of the new regulations. In addition, they could send automatic faxes to individuals that show the most important changes. Finally, they could produce small, printable color posters and send them via e-mail to the office managers, along with instructions to print them out and hang them in a prominent place. In this way, the information is targeted and delivered to the intended audience rather than passively listed somewhere.

Knowing where a person is when she needs to do business with you and being there with the right information and functionality is essential to e-business.

How do you do e-business?

E-business is the process of delivering any part of your business to any audience wherever they are. Even if this definition of e-business is clear, how to do it is not. In fact, it is often not clear at all to organizations how to make all of this happen. If you follow my definition of e-business, however, the following dictates quickly surface:

- **Know your audiences.** As you will see, to know your audiences, you must study them to fully understand what they want and how they want it, and then segment them into groups based on traits you can discover and track. Finally, for each audience, you need to create a value proposition, including what the audience wants, what you want from them, and how you are going to give value equal to what you want from them.
- **Know your business.** To know your business, you first must study it to understand how your business can be segmented into small, useful information and functionality parts. You must name and organize the parts and understand how the parts are created, maintained, delivered, and destroyed.
- **Relate the business to the audience.** You need to decide which audience wants what information and functionality in which contexts (on which pages, on which sites, in what other publications, and so on). You must then create a set of rules so that staff can decide who gets what when.

Organizations have been doing this more or less formally forever. On the other hand, establishing technology and process to effectively divide the organization into parts and deliver them electronically is new and far from well understood.

I believe that you can do e-business the same way you do content management. That is, you can create a process to collect, manage, and publish information and functionality to a set of target audiences. Information and functionality are the content of e-business. To do e-business or content management, you must do the following:

- **Collect content.** Your organization must set up systems that efficiently capture the information and the functionality you want to deliver. To collect content, you need editorial and *metatorial* systems. Editorial systems ensure that the content has appropriate and consistent format and style. Metatorial systems ensure that the content is appropriately and consistently tagged to be part of the organization's content scheme. See the section "The Metatorial Framework" in Chapter 20, "Working with Metadata," for more information on metatorial systems.
- **Manage content.** Your organization must set up a system to store and organize information and functionality outside of any particular delivery channel. Management systems are usually databases of one sort or another that store and categorize content, making that content easy to find and retrieve.

- **Publish content.** Your organization must set up a system to design and deliver the right information and functionality in the ways that your audiences expect and respond to favorably. The publishing system undoubtedly will manage a comprehensive Web site, but could and should manage the other sorts of publications your organization needs.

If your organization is wise, it will look past the hype and vague promises of e-business and focus instead on how to use the new technologies that become available to do what it has always needed to do. This book ought to give you an approach to all of e-business, not simply to the application of content management.

Content Management Cures Information Frenzy

It used to be that you could get away with creating new communications without worrying much about the old ones. Except for the occasional bibliography or citation, we could do most of our writing, recording, or filming in a vacuum. It was up to the reader, listener, or watcher to make connections between this new stuff and the rest of what might exist in the world. We also had the luxury of believing that our communications were single "things." We created memos, articles, books, songs, sitcoms, and movies. We could study and internalize the standards and techniques of our chosen genre, and then create one. In our vacuum, we could assume that a book was just a book, and a movie just a movie. Although it is simple and comfortable, in the Information Age, life in a vacuum sucks.

Today, the comfortable vacuum of the single communication product has been replaced by the unfathomably complex, fully connected, multiply targeted content component! Starting with the Web and ending who-knows-where, your job now as a communicator is no longer to create a single item — not a book or a movie. You now create content. Content is meant to be connected, used, and reused with no great assumptions about how and when. The Web is the first (and only the first) big place where the premium is on how your communications are connected to others rather than on how complete they are in themselves. Books, songs, movies: They all have a beginning, an end, and a point — the good ones are nice little stand-alone packages of meaning.

The good Web sites are the ones that are entirely enmeshed in connections with other sites and resources on the Web. Far from standing alone, they integrate into the wider Web of communications around them. The old communications are designed to be taken as a whole. You're not supposed to watch half a movie or listen to half a song. The Web is the first (and only the first) big place where all of these bets are decidedly off. If I am lucky, bits and pieces of this book will be spread across the Web. This can happen only if I'm smart and make it so that these bits of book have value apart from the whole. If I am realistic, I will realize that to be digitally successful, my work can never be finished. Each time you come back, it will have to be more up to date, more connected, revised, and reinvented. In short, for this book to be a good electronic publication, it will have to be fully connected, highly divisible, and in continuous production and evolution.

This is no small order. There are no established models for how to create this sort of work and there are no established rewards for doing it. Not only are there currently no good tools for connection, division, and continuous production, but also the concepts themselves are vague and ill defined. Add to this confusion the sheer volume of communication being created and disseminated, and you have a real mess. And that is how most people view the current situation — as a mess. Whether they are talking about their mailbox, office desk, company information, or the World Wide Web, most people suffer from the overload of disorganized, indigestible, always-in-your-face-but-never-there-when-you-need-it information.

So, we are stuck for a time in a tight dilemma. The force of history drives us toward faster and faster production of more communication. The force of the Web (which is the world's first global information repository) drives us to replace many of our time-honored notions of how to produce communication. Future generations might romanticize this era as the opening of the information frontier, but we mostly see it as a pain.

This book is my attempt to roll the experience and thinking of my colleagues and me into a complete picture of this new notion of communication and how it can address the dilemma of overload. By organizing content into wider and wider schemes of relationships, it becomes more accessible and memorable to the consumer, and more manageable to the creator.

Content Management in the Information Age

People talk about the Information Age as a fait accompli. They suppose that because there is a worldwide communication network, we are fully connected. They assume that computer technologies already have turned the world into a constant digital information stream. The coup-de-grace, many contend, is the World Wide Web, which has combined the global network with the latest computer technology to create the first tangible information economy, complete with transactions and stock market value for information assets.

In fact, we are at only the very beginning of the Information Age. The global network connects only a small minority of the world's population at very low speed. Computer-based information is viable only as long as you reduce the enormous complexity of information to the absurd simplicity of data. The World Wide Web proves only that information *can* be valuable. It has helped frame the issues of an information economy, but it has not begun to really address them. The tentative steps we have made so far prove that there *will be* an Information Age, not that it is really here. We are no further into our new age than the first cave dweller was in hers when she felled her first wild boar with a stone-tipped spear.

Still, the signs are all around us that the Information Age is coming, and coming quickly. The volume of information that needs to be produced and somehow managed has grown to the level that, without serious planning and organization, the cycle of information creation and use bogs down or simply crashes. For many organizations, this is the key dilemma that drives them toward more sophisticated systems for information management.

What will finally push us into the heart of the Information Age, I believe, is the information itself. When it is as easy to put a value on a small chunk of information as it is to put a price on a manufactured good, the Information Age will really be here.

The quantity of information to be handled drives us forward, but what really heralds the new Information Age is qualitative, not quantitative. The most important signs are the more subtle, qualitative ways in which our notions of information are changing, as follows:

- **Information is gaining value.** Traditionally considered to be a necessary evil on the way to their true goals, organizations are slowly bestowing upon information the status of an asset, not a burden.
- **Individual works are being subsumed by wider information webs.** Information is beginning to coalesce across creators, organizations, disciplines, and industries. We are beginning to ask authors not to create a standalone work but rather to contribute to an existing content domain. Creators are beginning to contribute to overall repositories of reusable information, where their work is related to and cross-referenced with other contributions to produce a growing web of content.
- **Information publication is disengaging from creation.** The way information is consumed is beginning to be unlinked from the way it is created. How a particular piece of information is formatted, delivered, and connected to other pieces of information can now effectively be varied and changed, based on the needs of the consumers. Thus, an author may not know how or when her work will be delivered.

In the heart of the Information Age, the exact value of the information you create will be known. In addition, each piece of information will contribute specifically to some very large and very rapidly expanding schema of content. Finally, consumers will simply expect that the information they receive is delivered in the way that is most convenient and useful to them — and they will gladly pay for it.

The time between now and the heart of the Information Age is a period of transition. We will continue to move in fits and starts toward a world where information is more and more freed from the context of its creation but more and more constrained to fit within a known system of categories and relationships. This transition is spurred by the technologies we newly have at our disposal to create, organize, and easily distribute information. It should be guided by the needs of the organizations that are using these technologies to interact more and more efficiently and intimately with their constituencies.

This book can help ease your transition from the information overload period we are now in to the information economy toward which we are headed. It tries to answer the following key questions:

- What does a system that handles massive amounts of information look like?
- How can a system be created that recognizes the value of each piece of information and guides contributors to most easily contribute to a growing knowledge scheme?
- How can a single system produce a very wide range of well-targeted custom publications from the same information base?
- How can all of this automation and systematization of information happen without endangering the very important relationship between the author and her readers, listeners, or viewers?
- What are the steps and processes you need to create such a system?
- How can this sort of system fit into and serve an organization's overall goals and initiatives?

So, I am promising you a lot in return for your commitment to content management. I am promising you that you will have a way to conceive of e-business and begin on your organization's journey into e-business. I am promising that you will have the tools you need to combat the onslaught of information that your organization is facing. Finally, I am promising that you will be closer than you were before to becoming part of the coming information economy.

You won't get this sort of return if you focus entirely on the products and problems of today. Instead, you must take a somewhat wider view of your organization, its constituents, and the content and functionality that your organization has to deliver to its constituents. Thus, I have really tried to discuss issues of the day without getting too stuck in today's muddle and confusion. I try when possible to stay clear of issues that have currency but no durability. Time will judge how well I accomplish this task.

My biggest hope is that this book raises the right questions and proposes answers that move you toward a successful system for matching content and consumers. I have no fantasy that this work is the last word on content management. I do believe, however that it is one of the first.

PART

I

What Is Content?

◆ ◆ ◆ ◆

In This Part

Chapter 1
Defining Data, Information, and Content

Chapter 2
Content Has Format

Chapter 3
Content Has Structure

Chapter 4
Functionality Is Content, Too!

Chapter 5
But What Is Content Really?

◆ ◆ ◆ ◆

CHAPTER

1

Defining Data, Information, and Content

◆ ◆ ◆ ◆

In This Chapter

What data is and how it differs from content

How the purpose of the information turns data into content

How to turn information into content by adding data

◆ ◆ ◆ ◆

Computers were built to process data. Data consists of small snippets of information that have all the human meaning squeezed out of them. Today, people call on computers to process content. Like data, *content* is also information, but it retains its human meaning and context.

In this chapter, I lay out one of the basic challenges of content management: Computers are designed to deal with data that's stripped of any context and independent meaning. Users want computers to deal with content, however, which is rich in context and meaning. How can you use the data technologies to manage and deliver very nondatalike content? This challenge isn't easy. If you err toward making your information too much like data, it looks mechanical and uninteresting to consumers. If you make your information too rich, varied, and context-laden, then you can't get a computer to automate its management.

The compromise, as you see in this book, is to wrap your information in a data container (known as *metadata*). The computer manages the data and the interesting, meaningful information goes along for the ride.

Content Is Not Data

Computers were first conceived as a way to perform computations that were too time-consuming or complex for humans. The model was (and to a large extent still is) as follows: If you can reduce a problem to a series of simple mechanical operations on numbers and logical entities (entities that are either true or false), it is amenable to solution by a computer.

Computer professionals were either programmers or data input clerks. Programmers reduced problems to a series of mechanical operations according to the following simple maxim:

> You input data; the computer processes it and then outputs it in a more useful form.

Clerks took care of inputting the data. They sat in long rows and columns, typing long rows and columns of numbers as well as small phrases, such as first name/last name and street address. As time moved on, computer scientists invented databases (bases of data) to organize and hold vast quantities of these snippets.

As you may expect, some problems were better solved this way than others were. Thus, as computer technology developed, the use of computers moved naturally from science to manufacturing and finance, where numbers were still the main event. Today, of course, computers are in everything. But the part of everything that computers are in is the reducible part. The reducible part is the part where a finite set of very specific rules operating on numbers and logical entities can yield a useful result.

The idea of computers as data-processing machines runs deep and continues to this day as the main thing that computers do. On the other hand, everyone knows that most users want computers to do more than grind finely through mountains of snippets. Today, people want computers to sift through mountains of large, complete chunks (not snippets) of information and deliver the ones that they want most at that moment. In addition, people want computers to deliver information of the quality that they expect from more familiar sources of information, such as books, radio, TV, and film.

From manufacturing and finance, computers moved to the business desktop as the replacement for the typewriter and the paper-based spreadsheet. Then, three related developments occurred in the personal-computer industry. Together, the following breakthroughs set the stage for a major change in our expectations of what computers can, and are, to do:

- Digital media creation (images, sounds and video) became possible.
- Digital media output (color displays, sound cards, and video accelerators) became available.
- Consumer-oriented mass removable storage (CD-ROMs, in other words) became available and cheap.

These developments led to the meteoric rise of the multimedia industry. For the first time, you could create and cheaply deliver actual information and not just data. Soon, multimedia CD-ROMs proliferated, with everything from encyclopedias to full-motion games. You can now consider your computer an actual replacement for familiar information channels such as books, TV, and radio. What these traditional channels deliver is *content* and not data.

What the multimedia industry began, the Web is in the process of completing. Today, getting your content online isn't only possible, but it's often preferable to obtaining it in any other way. I now listen to more music and talk on my laptop computer and Personal Digital Assistant (PDA) than I do on my radio or stereo. Still, I'm usually frustrated whenever I go to the Web because I expect to quickly locate the content that I want and see it presented at least as well as - in the traditional channels. Unfortunately, that's not always what happens.

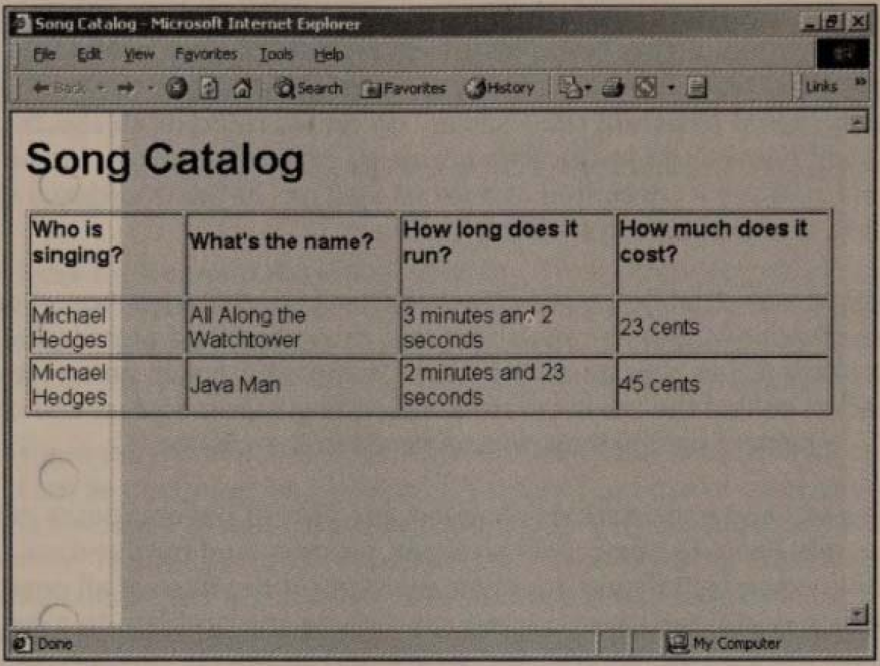
Note If you've worked with digital content for a while, then you realize just how sticky a paradox this situation is. People expect access to prove easy and presentation to seem compelling. If they are, it's only because someone's put in an enormous amount of effort behind the scenes to make everything appear so easy and compelling. Making content natural is an unnaturally difficult endeavor.

Although users' needs and expectations changed, the guts of the computer didn't. Ten years ago, people came to computers to input, process, and output data. Today, most people come to find and consume content. At the base of all computer technology, however, is still the idea that you can reduce any problem to a set of simple instructions working on discrete and structured snippets.

Data and content are different, certainly, but that difference doesn't mean that they don't interact. In fact, innumerable transitions from one to the other occur every day. Moreover, from the standpoint of the computer system, content doesn't exist — only data exists. Today, users have few tools for dealing with content as content. Instead, you must treat it as data so that the computer can store, retrieve, and display it.

Consider, for example, a typical Web interaction where you may go to a music site. You browse a page that features a music CD that you like; you add it to your shopping cart and then pay for it. What you experience is a series of composed Web pages with information about music as well as some buttons and other controls that you use to buy it. All in all, the experience feels like a content-rich interaction. What happens behind the scenes, however, is a set of data-oriented computer programs exchanging data with a database.

Some of the data behind the scenes is very contentlike. A database stores, for example, the feature article with the artist's picture. The artist's name is in one field, and the text of the article is in another. A third field contains the picture of the artist. Some of the data is very datalike. It consists of numbers and other snippets that create an economic transaction between you and the record company. A database stores your credit card number, your order number, the quantity that you order, and the order price, for example, and uses them in calculations and other algorithms. Some of the data is in-between data and content. The song catalog contains song names, running time, price, and availability, which are all snippets of information that can look a whole lot like data as the site's database stores them, but they appear more like content as the site displays them (as shown in Figure 1-1).


Song Catalog - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Links
Done
My Computer

Song Catalog

Who is singing?	What's the name?	How long does it run?	How much does it cost?
Michael Hedges	All Along the Watchtower	3 minutes and 2 seconds	23 cents
Michael Hedges	Java Man	2 minutes and 23 seconds	45 cents

Figure 1-1: As a site stores content, it can look a lot like any other data. In displaying that content, however, it can't look like data if you want to hold your audience.

In the database, it's all just data. On the page, the transaction data still looks a little like data; while the feature-article data looks nothing like data, and the catalog data can retain or lose as much of its data appearance as you want. On a well-designed page, however, visitors perceive it all as content.

So, from the user's perspective, information is all content, while from the computer programmer's perspective, it's all data. The trick to content management, in an age when the technology is still data-driven, is to use the data technologies to store and display content.

Content Is Information Put to Use

People in the computer world more or less agree to the definition of *data*. Data are the small snippets of information that people collect, join together in data records, and store in databases. The word *information*, on the other hand, contains all meanings, and no meaning, at the same time. You can rightly call anything, including data, *information*.

The word *information* holds a lot of meanings. For the purpose of this book, I use a pretty mundane definition. I take *information* to mean all the common forms of recorded communication, including the following:

- Text, such as articles, books, and news.
- Sound, such as music, conversations, and reading.

- Images, such as photographs and illustrations.
- Motion, such as video and animations.
- Computer files, such as spreadsheets, slide shows, and other proprietary files that you may want to find and use.

Before you ever see a piece of information, someone else has done a lot of work. That someone else has formed a mental image of a concept to communicate, and used creativity and intellect to craft words, sounds or images to suit the concept (thus crystallizing the concept). The person has then recorded the information in some presentable way. The author of the information pours a lot of personality and context into the information before anyone else ever sees it. So, unlike data, information doesn't naturally come in distinct little buckets, all displaying the same structure and behaving the same way. Information tends to flow continuously, as a conversation does, with no standard start, end, or attributes. You disrupt this continuity at your own peril. If you break up information, then you run the risk of changing or losing the original intelligence and creativity that the author meant the information to express. If you break up information, then you run the risk, too, of losing track of, or disregarding, the assumptions the author made about the audience and the purpose of the information.

The now defunct ContentWatch organization (`http://www.contentwatch.com/what.html`) gave the following definition of content:

> "What is 'content'? Raw information becomes content when it is given a usable form intended for one or more purposes. Increasingly, the value of content is based upon the combination of its primary usable form, along with its application, accessibility, usage, usefulness, brand recognition, and uniqueness."

Information that passes casually around in the world isn't content. It becomes content after someone grabs it and tries to make some use of it. You grab and make use of information by adding a layer of data around it.

The step of adding data may seem like a small step — from information to its use — but it's not. By refocusing from the nature of information to its practical application, you open up a world of possibilities for applying the data perspective to information. The crux is that, although you can't treat information itself as data, you can treat information use that way. As you begin to use information, you wrap it in a set of simplifying assumptions (metadata) that enable you to use computers to manage that use. Humans are mandatory in creating the information and figuring out the simplifying assumptions that wrap the information, but after that, the computer is fine by itself in doling information out in a way that usually makes sense.

By wrapping information in data, a small action by a person can trigger a lot of work by the computer. Suppose, for example, that to simplify information management, you decide that you need to consider a piece of information as 1) new; 2) ready to publish; or 3) ready for deletion. By itself, no computer can decide which of these statuses to apply to a piece of information. By wrapping your information with a piece of metadata known as *status*, however, and by having a person set the status metadata, you can use a computer to perform a lot of work based on the status. The computer doesn't need to know anything about the information itself; it just needs to know what status a human is applying to the information, as the following list outlines:

- If you give a piece of content the status "new," the computer sends a standard e-mail message to a designated reviewer.
- If you give a piece of content the status "ready-to-publish," the computer outputs it to a designated Web page.
- If you give a piece of content the status "ready-for-deletion," the computer removes it from the Web page and deletes it from the system.

In this way, the computer can accomplish a lot of work as the result of a small amount of work by people.

The Web version of the Merriam-Webster Dictionary (`www.m-w.com`) defines content in part as follows:

> "1 a: something contained — usually used in plural [the jar's *contents*] [the drawer's *contents]* b: the topics or matter treated in a written work [table of *contents*] . . ."

This definition provides a nice angle on content (something that *something else* contains. By switching from information to content, you're switching from a consideration of a thing to its container. You're shifting the focus from the information itself to the metadata that surrounds it. The container for information is a set of categories and metadata that . . . well . . . contain the information. This additional data corrals and confines that information and packages it for use, reuse, repurposing, and redistribution.

If content is information that you put to use, the first question to ask is, "What use?" What is the purpose behind marshalling all this information? For such a simple question, an astounding number of content management projects go forward without an answer. Or, to state the situation more precisely, all projects have some purpose, but the purpose may have little to do with the content that the project involves. A question such as "Why are we creating this Web site?" all too often receives one of the following answers:

- Because we need to.
- Because our competitors have one.
- Because our CEO thinks that we need one.
- Because everyone's clamoring for one.

These answers initiate a project, but they don't *define* it. To define the project, you need to answer the following question, "What is the *purpose* of the content we're about to put together?" If you provide a solid and well-stated answer, it then leads naturally to how you need to organize the content to meet your goal. The key to a good purpose is that it's *specific* and *measurable*. Following are some examples of good purposes for an intranet:

- To provide a 24-hour turnaround in getting any new product data from the product groups to the field sales representatives.
- To provide a site with all articles from the identified sources that mention the identified competitive products.
- To show all the data on a pay-stub with a complete history for every employee.

Notice that these goals are pretty specific, so quite a few of them may prove necessary to motivate a large intranet. As you organize your goals, you organize the content behind them. Your process is complete after all your individual goals fit together into a coherent whole and you can ultimately summarize them under a simple, single statement such as "Full support for the field" or "Zero unanswered employee questions."

Cross-Reference

In Chapter 14, "Securing a Project Mandate," in the section "The project mandate," I discuss the idea of a project mandate, which incorporates the organization's goals into a statement of purpose for the project.

Content Is Information Plus Data

There is something human and intuitive about information that makes treating it simply as data quite impossible. With data, what you see is what you get. With information, much of what it conveys isn't actually in the information but, rather, in the mind of the person creating or consuming it. Information lives within a wider world of connotation, context, and interpretation that make it fundamentally not amenable to the data-processing model. In fact, the concept of data was created specifically to remove these subjective qualities from information so that computers could manage it with strict, logical precision.

So, can computers ever really manage information? I believe that they can — albeit poorly by human standards. Until a new computer processing model appears, you can put methodologies, processes, and procedures in place to "handle" information by using data-age tools. Rather than reducing the information to mere data, you can capture whole, meaningful chunks of information and wrap it in descriptive data that computers can then read and act on.

Metadata makes the context and meaning of information explicit enough that a computer can handle it. Adding data to information (to metadata, that is) helps split the difference between keeping the information whole and enabling data techniques to effectively manage it. The data that you add to information is a way of making the context, connotation, and interpretation explicit. More important, metadata can make explicit the kind of mind that you expect to interpret the information. By adding a piece of metadata known as *audience type* to each chunk of information that you produce, for example, you can make explicit any implicit assumption about toward whom you're directing the information. Then, a computer can perform the simple task of finding information based on who's on the other side of the terminal. Of course, not all tasks are this simple, but the concept is always the same. You tag a large chunk of information with the data that the computer needs to access so that it can figure out what to do with that information.

Content, therefore, is information that you tag with data so that a computer can organize and systematize its collection, management, and publishing. Such a system, a *content management system*, is successful if it can apply the data methodologies without squashing the interest and meaning of the information along the way.

Until computers (or some newer technology) can handle content directly, you must figure out how to use the data technologies to collect and deliver content. Using data technologies to handle content is a central theme of this book.

From Data to Content and Back

From the perspective of this book, which attempts to reconcile the data and content perspectives, what is data and what is content depends mainly on how you create, manage, and bring each type of information onto a page, as the following list describes:

- **Transaction information** are datalike snippets that you typically use to track the processes of buying and selling goods. For transaction information and other very datalike sources, you don't generally use a content management system (CMS). Usually, they are managed as part of a traditional data-processing system. The templating system of a CMS, however, often does manage the displaying of such information on the -publication page.
- **Article information** is the sort of information that is text-heavy and has some sort of editorial process behind it. You create and manage article information, and other very contentlike information, most often using a CMS. In fact, without this sort of information, the CMS has little reason to exist. The CMS is also generally responsible for displaying this sort of information.
- **Catalog information,** which is information that might appear in a directory or product listing, can go either way. You sometimes create and manage it by using a CMS, and sometimes a separate data-processing application handles it. Obviously, if an organization already has a full-featured catalog system, tying your CMS to it is the easiest way to go. On the other hand, if you need rich media and lots of text, along with the part numbers and prices in your catalog, then including the catalog as part of the overall CMS may make the most sense.

The purpose of content management isn't to turn all data into content. The purpose of content management is to oversee the creation and management of rich, editorially intensive information and to manage the integration of this information with existing data systems. The CMS must, in all cases, carry the responsibility for the creation of the final publication. If this publication includes both data and content, then the CMS is there to ensure that the right data and content appear in the right places and that the publication appears unified and coherent to the end consumer.

Summary

Data was invented because it was much easier to deal with than information. It's small, simple, and all its relationships are clearly known (or else ignored). Data makes writing computer programs possible. Information is large, complex, and rife with relationships that are important to its meaning but impossible for a computer to decipher.

Here are some points to keep in mind as you hear and discuss the terms data, information, and content:

- Content (at least for the purposes of this book) is a compromise between the usefulness of data and the richness of information. Content is rich information that you wrap in simple data. The data that surround the information (metadata) is a simplified version of the context and meaning of the information.
- In a content management system, the computer manages information indirectly through data. The content compromise says, "I can't get the computer to understand and manage information, so I must simplify the problem. I must create a set of data that represents my best guess of the important aspects of this information. Then, I can use the computer's data capabilities to manage my information via the simplified data."
- Someday, someone may invent computers that can deal directly with information. I'm not holding my breath, however, because to do so requires cracking some of computer science's hardest problems, such as artificial intelligence and natural language processing. In the meantime, everyone must make do with the current dumb but very strong beasts. (If someone does finally invent these new machines, they may need to change their name from computers to something less mechanical.)

In the following two chapters, I describe the two basic qualities of content that a CMS must account for: *format* and *structure*.

✦ ✦ ✦

CHAPTER

2

Content Has Format

◆ ◆ ◆ ◆

In This Chapter

Why the binary (or file) format is important

How rendering (or visual format) must be separated from its content

Why formatting can prove such a pain for the content manager

The kinds of formatting you're likely to encounter and how to categorize them

◆ ◆ ◆ ◆

To communicate information, you must encode it. That encoding is known as *format*. In the computer world, format covers two broad, but related, concepts: *binary format* (also known as *file format*) and *rendering format* (sometimes referred to as *display format*). For a CMS to function correctly, it must offer the capabilities to receive and create a variety of binary formats and to separate rendering format from the content it surrounds.

In this chapter, I focus on format and discuss why it's so important but often so hard to deal with.

Binary Format: Storing Information

For a computer to use information, it must convert it to binary code. If you digitize a printed picture, for example, then the scanner must convert the picture from tiny blobs of color on paper to a structured set of ones (1) and zeros (0). Over time, the various kinds of structures that programmers have devised have become generally accepted. In images, formats such as JPEG and GIF are now the standards for Web publication, and formats such as EPS and TIFF are the standards in print publication.

Computer files are the traditional way to store binary information, so binary format is also known as *file format*. Each binary format has its own way of representing the information that it encodes. Most image files, for example, feature code at the top of the file that tells what kind of file it is, and how big it is, and gives a wealth of other format-specific data. Vector graphics are encoded as a set of equation values that, after calculated, create the shapes that you see in the image.

In text, the characters you type are generally encoded using *American Standard Code for Information Interchange* (*ASCII*) or its more modern equivalent — *Unicode*. This definition is the most fundamental one of format — the way information is encoded so that computer applications can use it.

Binary format is important because you often must move between one binary format and another. You may receive images in EPS format, for example, and must convert them to JPEG format for a Web publication. Or, you may need to keep both versions because you need to use the same images in print and on the Web. Although the various formats all represent the same sort of information, moving from one to the other isn't always easy.

Microsoft Word (DOC format) files, for example, are notoriously hard to convert to Web files (HTML format). Although they're both text formats, any number of conventions that exist in a DOC file don't exist in an HTML file. Thus, you get inconsistent, poor results if you try to convert between one and the other without a lot of intelligent human oversight.

Rendering Format: Presenting Information

In addition to being the way that you encode a file, format is more commonly known as the qualities that you use to visually render content. If I use the word *format* without qualifiers in this book, I mean this sort of format. Typographic qualities such as **bold**, *italic*, and underline, as well as layout qualities such as tables, right alignment, and margins, are all part of this definition of *format*. Although I dwell most on text formats in this work, the other media types (images, sound, and motion formats) all have unwritten equivalents of the notion of format.

Rendering format is important because you must manage it across all of the content that you intend to handle. The following list describes two ways in which you must manage your format:

- **You must make format consistent** across content categories, as well as across all content, in a single publication. To ensure readability, for example, you want to make certain that all of your news releases feature the title centered and boldfaced at the top of the page. You can also format a cross-reference link the same way, across all of the content in your publications, to ensure that the user recognizes it as a link.
- **You must separate format from content** so that you can reuse the content in a variety of outputs. Although making your titles centered and boldfaced may be appropriate in your Web design, for example, you may prefer to make them left-justified and italicized in print. In this case, the title itself doesn't have any set format associated with it; rather the publishing system enables the title to adopt the title format that the specific publication uses.

The Language of Formatting

Print editors and publishers have long understood the power of formatting and have developed strong formatting conventions to guide the user. Few such conventions exist yet for the computer screen. Moreover, whenever the computer industry borrows too heavily from the print world, it fails because many conventions require specific qualities of the physical page. On the other hand, whenever the computer industry disregards convention and invents a new language of format, it also fails because people simply don't understand it. *Wired Magazine*, for example, features a great, innovative layout and design. But, it defies so many conventions that, for some people, it's just plain hard to read.

My generation spent its youth learning the language of print formatting. It's familiar and subconscious. My children still learn the language of print, but now it has a counterpoint. My children also learn the language of the screen "on the street," as it were, as well as through games and on the Web. The language of the screen is a "pigeon" language. Like the pigeon English spoken on merchant marine vessels by natives of a hundred nations, it borrows from everywhere, messes with established meanings, and creates new ones at will. Like any pigeon language, the language of the screen is more fun, but far less agreed-upon and stable than its parent tongues.

In the end, we will have a universal set of conventions for computer presentation—a real language of the screen. But, we will arrive at it in fits and starts as we find techniques that work. I can say from grueling experience that, although we have a long way to go in creating conventions, we are certainly ahead of where we were 20 years ago when I started putting words on-screen.

For the content manager, rendering format is best thought of as separate from content. For the consumer of content, the two are inseparable. The typography and layout of the page tell you much of what you need to know about the page. Format is a kind of metadata. It's information above and around the language on the page that tells your brain what to do with the language on the page. It tells the reader things such as "This part is important," "Read this section first," or "This text is a link." It guides your eye and your emotions around the language, leading you, if done well, to a much faster and fuller understanding of the information on that page.

Dealing with Formatting

Format is the part of content that's often the hardest to deal with. The reason is that authors don't apply formatting consistently. If formatting is inconsistent, however, automating the process of separating format from content is almost impossible. In other words, you need to indulge in a lot of manual labor to make content ready for your system.

To ensure consistent formatting, professional writers use a *style guide* that specifies very precisely how to format each element of the manuscript. Unlike the style guides you might buy in a bookstore, these documents go paragraph by paragraph through a model of the publication to be produced, telling you exactly how to do each kind of formatting the publication requires. Word processing programs help enforce consistency by providing a way to name and apply a style to a block of text. Even among professionals, however, you see a strong tendency to apply formatting directly rather than go through the extra trouble of using word processing styles, often for the following reasons:

- Without a strict style guide, keeping consistent in the way that you apply formatting is really difficult. Creating good, complete style guidelines requires a lot of experience. Worse, adhering to style guidelines takes incredible patience and fastidiousness.
- Most word processors enable you to easily circumvent style constraints. The commands to apply formatting "correctly" (by using style templates, for example) are generally harder to use than the ones that apply it wrongly by clicking (the little bullets button in Microsoft Word, for example).

For the person who must work with preformatted content, the problems compound. For the following reasons, people who aggregate content are often very frustrated:

The Inner Struggle of a Lazy Writer

I present the following brief inner dialog for the gratification of those of you who, like me, wish that they didn't know the "right" way to format text:

> "Okay, I need to put in a bulleted list." My pointer moves effortlessly toward that little button at the top of the Microsoft Word window to make the bullets. Then I stop.
>
> "No," I tell myself, "that's the coward's way to format a bulleted list. If I do it that way, it's sure to all turn out inconsistent, and who knows how it may translate to HTML. I need to use the style that I defined for bullets." So, I open the style drop-down list to find the right bullet style.
>
> "Damn," I grumble. "Not there. I started a new document and didn't bring over the new styles." Finding my styles and bringing them into my document takes me ten minutes. I open the style list again.
>
> "Aaarrgg, which bullet style is it, again? I see 'Bull', 'Normal Bullet,' 'Bullet Indent,' and 'Body Bullet.'
>
> I can't, for the life of me, remember which one I'm supposed to use or even where some of these styles came from. I choose "Bull" (because it sounds like the kind of styles that I usually create) and hope for the best, even though it doesn't look quite right. I can change the definition of the "Bull" style to fit what I expect to see, but who knows what this change may do to the other lists that use "Bull."
>
> "Forget it," I say to myself. "Let the editor deal with it."

- Because formatting is hard to apply consistently, content often arrives poorly formatted.
- Because style guidelines vary from group to group, if you aggregate content from many sources, it can become a tangle of conflicting formatting.
- Because the way in which rendering format is encoded as binary format varies widely from tool to tool, you often have no way to translate one format to another. Each tool, for example, offers its own way to represent tables. You often find no way to represent the table attributes possible in one tool in the binary format that another tool uses.

For the content manager, who must ensure that the same content can appear in various publications, each with its own binary as well as rendering-format conventions, the problems are also severe. Very few binary formats really enable you to separate formatting from content. XML and its parent SGML are among the few that do offer this type of separation. The manager faces the difficult choice of either converting all of her content into a format such as XML or storing it in a binary format where rendering format and content are inseparable (such as HTML). If the manager chooses to convert to XML, then she incurs a big up-front conversion cost, and an ongoing cost, to reconvert it back to standard print and electronic binary formats for each publication. If she chooses to use a format such as HTML, then producing Web publications is easy, but other formats may require extensive manual effort because of poor convertibility between text formats.

Categorizing Formatting

You can categorize formatting in three different ways: by emotional effect, by visual effect, and by scope of application.

Formatting for effect

First, I divide formatting based on the desired effect of the formatting. The author has one of the following intents whenever she adds formatting to text:

- **Emphasis or importance:** Size, position, and color all direct your eye. They translate in your mind directly to the relative importance of the text to which you apply them. As everyone knows, but few people seem to remember, *importance* is a relative term. If everything on a site screams its importance at you, untangling exactly what is most important becomes impossible. You've no doubt seen the sort of Web site where everything screams out at you. Each new item on these sites ups the ante. Each competes for your attention in an ever-expanding battle of size, position, and (yes) blink rate.

- **Parsability:** You can take in only so much information in one eyeful. Well-used formatting breaks a mass of unparsable information into eye-sized chunks. White space (carriage returns, tabs, spaces, margins, and so on) is the main type of formatting that you use to segment information. A well-formatted presentation moves your eye quickly and naturally from area to area.
- **Interest:** Authors often use formatting to break the boredom of a flat presentation. The eye and brain respond to variation. By breaking up the otherwise smooth flow of text with various styles, you entice the reader to continue. The savvy author knows when to break cadence and offer some interest.
- **Cultural norms:** Compare the formatting that you see in the *Wall Street Journal* and *USA Today*, two popular U.S. newspapers. The styling and layout of the *Wall Street Journal* reeks of formality, institutionalism, and stability. The styling and layout of *USA Today* exudes a casual, friendly, easily accessible air. In both cases, the formatting does far more than draw your eye or generate interest. It links the publication to a cultural context. Formatting is a major key to where the publication sits in the vast web of cultural relationships. This format-based social positioning is serious business — so serious, in fact, that most companies create strict usage guidelines that specify in detail the fonts, colors, and positioning that employees may use for the identity-creating elements of their publications.

Formatting by type of effect

Next, I categorize formatting by the following types of visual effects that the formatting employs:

- **Typographic effects:** These effects apply to the qualities and look of text. Font face, font size, kerning (the space between letters), text color, and effects such as underlining, bold, and italics are all examples of typographic effects. These qualities are the ones that most people think of if you talk about formatting.
- **Layout effects:** These effects apply to the positioning of content areas. Margins, columns, tables, wrapping, and indentation are all examples of layout effects. Layout formatting tends to be the hardest type of formatting to translate from one type of publication to another (especially from print to the Web).
- **Background effects:** These effects happen behind and around the content. Cell colors and images, background colors and images, watermarks, and reverse effects are all examples of background effects. The rule of backgrounds, known well by print folk but still poorly understood on the Web, is that "If you even notice the background formatting, then it's too strong."

Formatting by scope

The final way that I catalog formatting is by the scope of its application, as follows:

- **Character formatting:** This type of formatting applies to one or more text characters. If changing between one character and the next in a particular sentence makes sense, then you can count it as character formatting. Character formatting includes most of the typographic effects. The familiar bold, italic, and underline are character formats, as are all of the font (or, more accurately, typeface) effects. Although character formatting is the most familiar, and easiest to understand, type of formatting, it generally has little to do with structuring or managing content, and often gets in the way of these important functions.
- **Paragraph formatting:** This formatting applies to blocks of text that paragraph breaks separate. Many of the layout effects such as indents and line spacing belong in this category. Unlike character formatting, paragraph formatting plays a major role in structuring and managing content. First, structural elements such as headings, summaries, and titles usually sit in their own paragraphs. Second, paragraph formatting is the most common application of the use of styles, where you roll a set of character and paragraph effects together under a single name and apply it to a set of text blocks. Style not only is a way to group formatting; it's also a way to give structural names to text blocks. Applying the style *title*, for example, to a paragraph doesn't just capture the fact that you want a big, bold, and centered paragraph. It can also serve as a marker to say that this paragraph contains the structural element, the part of the content that you call *title*.
- **Page formatting:** This type of formatting unifies a set of text (and media) blocks into a coherent presentation. As the name implies, it applies to a single (printed or online) page. Effects such as margins, columns, and backgrounds are what you commonly apply at this page level in HTML. In print, page formatting is usually applied for an entire publication. In each medium, page formatting is applied at the smallest complete unit of presentation. In print, you rarely take a single page away from its surrounding section (a chapter or maybe a subchapter). In HTML, viewing a single page apart from its neighbors is standard practice. Page formatting sets the layout and presentation of a standard block of a publication.
- **Subpublication formatting:** This formatting applies across pages to an entire section of a publication. As I mentioned earlier, you generally define layout effects at this level in print. On the Web, techniques such as the use of cascading style sheets and templates carry the burden of forming a single look across pages.

- **Publication formatting:** This type of formatting spans all characters, paragraphs, and pages in the publication. In most authoring environments, you can select an entire publication and make the text bold, but doing so is a truly vile thing to do from the standpoint of format or structure management. It means that someone will have to come along later and figure out how to interpret that global change and apply it in each separate element of the publication. How is publication-level formatting accomplished? In general, what happens is that you share subpublication formatting effects across all sections to form publication-wide formatting. Style sheets and templates are the tools that you use most commonly to extend formatting across sections and into the publication as a whole.
- **Suprapublication formatting:** This formatting applies across publications. Just as you share subpublication formatting to get publication-level formatting, you share publication-level formatting between publications to extend the same standards across publications. Again, style sheets and templates are the vehicles that you use most often for defining and enforcing formatting across publications. An example of suprapublication formatting is the branding guidelines of most companies. Every publication that the company produces must follow these rules.

Summary

As people read, they tend to take formatting for granted. Words are boldfaced or centered, text lies in columns or tables, and readers barely take notice. For the content manager, however, format warrants great attention for the following reasons:

- The system that she works with must offer the capabilities to accept the binary formats that she submits and must create the binary formats of the publications that she must produce.
- If she applies rendering format inconsistently to the text that she receives, then cleaning it up and separating the format from the content can require a lot of manual labor.
- She must recognize, analyze, and synthesize the wide variety of formatting that she receives into a single, coherent system. Whether it applies to a single character or a whole family of publications, all of the formatting must fit into a single system of emphasis, interest, and structure.

Formatting is the first basic quality of content that you must account for. In the next chapter, I detail the second basic quality: structure.

✦ ✦ ✦

CHAPTER

3

Content Has Structure

◆ ◆ ◆ ◆

In This Chapter

Understanding the importance of structure

Structuring your content (and inherent difficulties)

Categorizing different kinds of structure

◆ ◆ ◆ ◆

Structure is the way that you put information together. It encompasses the parts and pieces of a *base of content* and their relationships to each other. The title of a Web site, for example, relates to the rest of its pages in the same way that the name of a movie relates to its scenes. Along with format, structure is one of the basic qualities of content that you must understand and control to manage. Structure, however, goes far beyond format in importance. If you know and control the structure of a body of content, you can create whatever format suits your needs for a particular presentation. Format varies with the way that you present a body of content, but its structure remains the same.

In this chapter, I detail the structure of structure. Regardless of how important structure is, it can still be a pain to work with.

Structure Is Important

If format is about presentation, then structure is about management. Structure is the set of named relationships within, between, and beyond individual pieces of content. If a content base is well-structured, then it has the following features:

- Its content divides into a set of well-defined categories (which I call *component classes*).
- Within each category, the content segments into manageable chunks (which I call *component instances*).
- Each chunk divides further into a well-defined set of parts (which I call *elements*).
- Each chunk, or component instance, relates to other chunks, from the same or other categories, by way of outlines, indexes, cross-references, and sequences.

In short, a well-structured content base is well-organized. Just as in a well-organized file cabinet, where every sheet of paper goes into a specific folder, every folder goes in a certain drawer, and every drawer has a particular location, a well-organized content base is one that you divide and completely catalog. If you look closely at the organization of the content, then you can see that it is a set of structures laid in and around the content that you intend to collect, manage, and deliver.

Creating an overarching set of structures is *the* way to understand and control content. If you have well-defined categories, you can standardize the way that you create and evaluate them as well as use them. If you divide content into chunks, you can track them individually. If each chunk of a particular category has the same set of named elements, then you can count on a particular piece of information remaining available when you need to automatically lay out a page. Finally, if each chunk is stored within the same structures as all the others, you can create a fast and efficient storage and retrieval system that can automatically deliver targeted content into a variety of publications.

Traditionally, structure was solely the author's concern. She would compose, arrange, rearrange, and finally arrive at a structure that made most sense to her. From the beginning of composition to this day, authors have agonized (alone and with their editors) over the minute structure of what they were creating. Unfortunately, much of the intimate understanding an author achieves is lost to later users of her work. With the possible exception of academic writers, few authors take the time to know how their work fits into any wider framework. In academic writing, it is not only standard, it is mandatory to specifically cite the other works that relate to yours. Still, even academic writers create their own worlds of structure, which may or may not relate well to any other content in the world outside. Although working in isolation enhances creativity by not imposing any outside restrictions, it makes relating one work to another extremely hard. And, of course, relating one work to another is what content management is all about.

Structure is always key to managing content. But depending on the particular user, the kind of structure most-needed varies. A content creator, for example, may want to see how her content relates to the other content that is in the system. An administrator wants to see all content by type or by source. An end user cares most that she can find just the right chunk of content to answer the question she has in her mind. All three of these people want structure, but each wants a different kind of structure. The author wants to see her content chunk in the center of a set of relationships. The administrator wants to see a content outline categorized by content type and by author. The end user wants to see outlines that divide the content based on the issues that she wants to resolve. Thus, the same base of content needs a variety of structures to serve all constituents.

A good CMS enables you to design and apply a variety of overlapping structures on the same body of content.

Structure Is Like a Map

You can think of structures as maps that you lay over a set of content chunks to organize them and make them accessible to the people who need them. Just as many different kinds of geographic maps describe the same territory, many different kinds of structures describe the same content base.

A map of your neighborhood, for example, can show the area topographically, color-coded by household income, or by the distribution of electrical power. Each of these maps shows the same area, but organizes the information for different purposes. Similarly, you can "map" your content base by content type, audience type, or by the date that you create each chunk of information. Again, each of these structures shows the same content, but each organizes the content for a different purpose. The content-type view shows authors how much of their type of contribution is in the system. The audience-type view shows end users the information that may be of most interest to them. The create-date view shows administrators how much new content is flowing into the system and how often.

Structure Can Be Difficult to Create

Structure is the way that the chunks of content (components) fit together into a unified framework. That sounds great, but it assumes the following:

- ✦ That such a unified framework exists.
- ✦ That you can figure out how to relate your components to your structure.
- ✦ That you can muster the effort to apply the structures over the base of content that you must deal with.

Consider a typical business intranet. It starts with a group of representatives from the various parts of the organization in a room. Everyone states what they want on the intranet. The HR department wants all employee information to be there. The executives want to see business news and competitive analysis. Managers want policies and procedures. Each functional group wants its own set of specific documents and links. Sounds fine. But as they start brainstorming, the farther they get, the less they know about what each of these categories of content actually consists of. Which business news? Exactly what competitive analysis?

Someone says, "Why not just start by putting up what we have now?" They all agree that this idea could solve the problem and go away to collect key documents. The next day, they come back together, each with a large stack of papers. With all the stacks lying menacingly in front of them, they discuss how all this information fits together. Hours later, exhausted and disenchanted, someone sums up the meeting by saying, “I had no idea it was all this complex. How do we ever get anything done around here at all with this mess of information?” The fact is that an organization’s content is rarely organized. It’s a mess of loosely connected point-to-point communications, each with its own purpose, assumptions, and implicit structure. It’s not too hard to see what you need to do at the macro level. At the level of the whole intranet, you can easily see how it all fits together. However, at the micro level, the level at which the content was first created, it didn’t need to fit with anything else. As long as the intended audience gets the idea, who cares whether anyone else does? At the intermediate level, however, where the micro meets the macro, is where the problems occur. Not that someone can’t whip this rag-tag information into content. Doing so just requires a lot of effort.

So, the beleaguered group comes back with renewed purpose. “Yes it’s hard,” says the one. “But we can do it!” They agree, knowing that the alternative is to return defeated after just two meetings. The one who was an editor on the college newspaper rolls up her sleeves and leads the others in this new direction. They begin a major synthesis, painstakingly fitting each type of information into one big outline. They note any gaps in the material and how they need to classify and modify each kind of document to make them all fit together. In the end, they have a full-site outline and an idea of what making all these micro communications fit into the macro scheme that they came up with in their first hour together is going to take. They go home happy.

The final meeting is meant to clean up details and figure out how much budget to ask for to start the intranet. In the back of everyone’s mind (except the Web Guy), is the idea that the Web Guy receives content from lots of contributors around the organization and just puts it together into a nice set of pages. So, they ask him how much time he needs to build the site. The Web Guy’s jaw drops. “Me?” he stammers. “You think that I can do all this work alone? You all took hours just to figure out how one of each of these documents fits in. I don’t know about HR or product specs. How am I supposed to figure out how all this fits material together? And how many documents are we talking about? Sorry, but I have my hands full just keeping the servers running and designing pages for our Internet site.” Now some other jaws drop as others start to realize that he’s right. Few others in the organization besides the people at the meeting could successfully fit documents into the outline they created. And no way could any of them take on the extra task.

The woman from the organization’s consulting group starts to smell a familiar situation — a scene that she’s seen played out many times, as the fact that the task at hand far exceeds the available budget or staff becomes clear to everyone. “It’s a phased approach,” she says calmly. “We start with just a few of the most important ones of each type. Then, as the concept proves out, we can ask for more resources to grow the site.” They all agree and leave the last meeting feeling as if they’d taken a course at the school of hard knocks.

together. Hours later, exhausted and disenchanted, someone sums up the meeting by saying, "I had no idea it was all this complex. How do we ever get anything done around here at all with this mess of information?" The fact is that an organization's content is rarely organized. It's a mess of loosely connected point-to-point communications, each with its own purpose, assumptions, and implicit structure. It's not too hard to see what you need to do at the macro level. At the level of the whole intranet, you can easily see how it all fits together. However, at the micro level, the level at which the content was first created, it didn't need to fit with anything else. As long as the intended audience gets the idea, who cares whether anyone else does? At the intermediate level, however, where the micro meets the macro, is where the problems occur. Not that someone can't whip this rag-tag information into content. Doing so just requires a lot of effort.

So, the beleaguered group comes back with renewed purpose. "Yes it's hard," says the one. "But we can do it!" They agree, knowing that the alternative is to return defeated after just two meetings. The one who was an editor on the college newspaper rolls up her sleeves and leads the others in this new direction. They begin a major synthesis, painstakingly fitting each type of information into one big outline. They note any gaps in the material and how they need to classify and modify each kind of document to make them all fit together. In the end, they have a full-site outline and an idea of what making all these micro communications fit into the macro scheme that they came up with in their first hour together is going to take. They go home happy.

The final meeting is meant to clean up details and figure out how much budget to ask for to start the intranet. In the back of everyone's mind (except the Web Guy), is the idea that the Web Guy receives content from lots of contributors around the organization and just puts it together into a nice set of pages. So, they ask him how much time he needs to build the site. The Web Guy's jaw drops. "Me?" he stammers. "You think that I can do all this work alone? You all took hours just to figure out how one of each of these documents fits in. I don't know about HR or product specs. How am I supposed to figure out how all this fits material together? And how many documents are we talking about? Sorry, but I have my hands full just keeping the servers running and designing pages for our Internet site." Now some other jaws drop as others start to realize that he's right. Few others in the organization besides the people at the meeting could successfully fit documents into the outline they created. And no way could any of them take on the extra task.

The woman from the organization's consulting group starts to smell a familiar situation — a scene that she's seen played out many times, as the fact that the task at hand far exceeds the available budget or staff becomes clear to everyone. "It's a phased approach," she says calmly. "We start with just a few of the most important ones of each type. Then, as the concept proves out, we can ask for more resources to grow the site." They all agree and leave the last meeting feeling as if they'd taken a course at the school of hard knocks.

The way the team approached this problem involves a number of problems, but who can blame them? They had no idea of the magnitude of the task. Thinking it an easy problem, they approached it casually, looking for an easy solution. Structuring large quantities of information is similar to being hired to clean and maintain a large house where 16 children live. If you're lucky, it's merely a full-time job, because the house is basically organized but just a little messy. If you clean it every day and put things in their assigned places, you can maintain general order. You're more likely, however, to walk into a complete disaster on the first day, and just figuring out what the right place is for everything is likely to take you forever.

In summary, challenges you are likely to face when defining and applying structure include:

- **Understanding content in its entirety:** Creating structure requires an extensive understanding of the entire base of content. Doing a quick overview of a limited sample and arriving at structures that can persevere simply isn't possible.
- **Controlling all content:** To bring all content under control, you must subsume the structures already present in each type of content you receive. You must create a new structure that can encompass all existing content without changing it so drastically that it no longer meets the needs of the group that creates it. At the same time, the standards must make each group's content accessible and usable to other groups that use the same information for a different end goal.
- **Overcoming resistance to change:** Most groups that produce a lot of content arrive at some set of structure standards. The harder it is for the group to come to those standards and the more comprehensive the standards are, the harder that group is likely to fight to retain its standards (whether they serve the general good of the organization beyond the original group or not).
- **Providing a flexible solution:** Structuring is an iterative process. The structure that you originally define is very likely to change and deepen as you begin to fit the bulk of your content into it.
- **Enforcing your standards:** Unless you give everyone specific direction, the structure that one author creates is unlikely to easily mesh with the structure other authors create. Additionally, many authors often don't want to bother learning and obeying rules. Thus, you must make your standards enforceable but, at the same time, easy to understand and apply.
- **Understanding the mechanics of structure:** Generally, few individuals in the organization understand the mechanics of structure well enough to analyze and manage it. I coined a new term, "metator," to describe this poorly understood job function. A metator is someone who edits a body of content from the standpoint of its metadata structures.

- **Finding the right balance:** No single structure can cover all your content for all its users. You must create an overlapping family of structures that all describe the same content base in a different way so that authors, administrators, and end consumers all enjoy the sorts of structures that they need to complete their respective tasks.

A CMS provides the structure and process you need to overcome these structure hurdles. Still, groups venturing into the world of content management should recognize it for the huge beast that it is, and start by teaching their staff some more advanced content creation skills and getting them into the habit of adding some basic structures to their documents (for example, styles and metadata). Over time, they will build a large enough body of well-structured content that the task of implementing a CMS won't be so daunting.

How to Categorize Structure

If I scared you in the preceding section, then good. Information is a beast. Taming it requires determination, strength, and cunning. Your best weapon to tame the information beast is to understand how to put information together. So, in this section, I look at the structure of structure (the metastructure, if you like). You can categorize structure based on the purpose for which you're creating it, the type of structure you're creating, and the scope of the structure you're creating.

Structure by purpose

The following list describes how you might categorize structure based on your *purpose* for creating that structure:

- **Structuring for a single publication:** Outside a CMS, most people have a particular publication in mind as they endeavor to create structure. Whether that publication is as small as a memo or as large as an intranet, the author has a single unified target in mind. She creates the structure that directly aids the presentation of that publication.
- **Structuring for a content base:** In a CMS, your purpose for creating structure is to make content easier to create, easier to manage, and easier to automatically build into a variety of publications. Although a simple outline may suffice for a particular publication (a Web site, say), you need a much more robust structure to organize the creation and storage of the information, as well as its output into more than one publication.

Structure by type

If your purpose is to structure a large body of content for a range of contributors and publications, then you need most of, if not all, the following sorts of structures:

- **Divisional structure:** This type of structure slices content up into useful chunks. At the smallest level, by ending a sentence with a period, you divide words into sentences. At the largest level, by naming your Web site bologna.com, you divide the content there from the rest of what's on the Internet (at least you hope so). In a CMS, you focus mostly on the following two types of divisional structure:
    - *Named segments* of content, such as product reviews, tech support articles, and press releases.
    - *Named elements* within the segments, such as title, abstract, body paragraph, and emphasized text, footnote, and sidebar.
- **Access structure:** This type of structure presents the divisions that you and your audiences need to access your content. This type of structure includes the one most people think of first as structure—the outline. It goes far past the outline, however, to include a range of other navigational structures, including the following:
    - *Hierarchies* (also known as *outlines*, *taxonomies*, and *tables of contents*), which show parent-child relationships between different segments of content.
    - *Indexes*, which provide a map between text phrases and the content segments that relate to them.
    - *Cross references* (also known as *links*) from one content segment to another, either within the same work or to other works.
    - *Sequences* (also known as *browse sequences*) that specify which segment of content comes before or after the current one.
- **Management structure:** This type of structure specifies the attributes of a content component that enable you to track and manage it. Data such as author, creation date, version number, and review status are all forms of management structure. You can think of this sort of structure as an auxiliary map over the content, which gives the manager's cut on what's important.
- **Inclusional structure:** This structure type specifies which components include others. If you create an image reference in an HTML page, for example, you're putting a little piece of inclusional structure in the HTML that says, "Go find this picture and include it here."

Cross-Reference

In Chapter 20, "Working with Metadata," I revisit each of these types of structure as metadata because that's how you actually code them into content. In Chapter 26, "Designing Content Access Structures," I talk in detail about the concepts of hierarchy, index, cross-reference, and sequences.

Structure by scope

Structure, like format, also has scope, as the following list describes:

- **Structure below the radar:** You can certainly say that a period is a bit of divisional structure, but so what? Every content system that you encounter includes a level below which dealing with that structure is both infeasible and unnecessary. The level of granularity that you pay attention to depends on the system. In a public library, for example, which is a content system, the librarians don't care about the structure of each individual book. They neither notice nor encode this structure. Rather, they code the structure among books, put it on little cards (digital cards or paper ones), and make it available to visitors.
- **Inner structure:** The structure that you find represented within a content component is the inner structure. Following the library example in the preceding paragraph, you can say that each book is a component. The table of contents and index within each book are the inner structures within the components. Librarians today don't generally care about the table of contents within the books that they manage. But this fact doesn't mean that people never care about internal structure at all. Imagine a day when a librarian decides to make a master table of contents for the entire library. The book is still the basic component in the system; now, however, the tables of contents, which are still inner structures, are of interest, and the librarian must now access and, God forbid, combine them. The key point with inner structure isn't whether you need to notice it, but rather that it's a part of the component and always travels with it.
- **Outer structure:** This structure is what relates one component to another. In the library example, the card catalog is the outer structure that organizes the cards, and the books they represent, into a useful system. You always care about outer structure. Interestingly, you may store the outer structure inside or outside the component. In the library, in fact, you store it both within and outside the component: One paper card, or its equivalent, is inside every book, and one record is in the catalog database. In electronic systems, you often store outer structure along with components. You commonly store the name of a component's parent in the hierarchy, for example, as well as its links to other components.

Summary

Structure is the key to managing content. Control the structure and you control the content and its publication, which, of course, is easier said than done, as the following list verifies:

- Structure is often buried in the formatting of source content. Even if the formatting is consistent, which often isn't the case, extracting the structure and making it explicit in a set of tags or database fields may prove difficult.

- Organizations go to war over competing structures. "We should list jobs by department." "No, we should list them by type." Although a good CMS enables you to lay multiple simultaneous structures over the same content, reaching an agreement on exactly how to divide content may prove more daunting a task.
- The content architect (or Metator) has her hands full dividing undifferentiated information into chunks and tagging it for access, management, and inclusion. In addition, she must create hierarchies, indexes, cross-reference structures, and sequences that tie each chunk into a wider system. Although a level below which you can ignore structure does exist, that level is far lower than you might like it to be for your current workload.

In the next chapter, I expand past information and describe how you can manage computer functionality as just another form of content.

✦ ✦ ✦

CHAPTER 4

Functionality Is Content, Too!

◆ ◆ ◆ ◆

In This Chapter

The definition of functionality

How to produce functionality in small pieces that you can share

How the division between functionality and information blurs

How you can treat functionality as just another kind of content

◆ ◆ ◆ ◆

Since the time computers first entered organizations as accounting and manufacturing automation systems, the industry's seen a steady movement toward computer-based business processes. Today, a worker in an organization commonly spends her entire day "doing business" in front of a computer terminal. Inside organizations, business is largely electronic. Until recently, however, the movement of information between organizations was slow. Organizations and their partners were often on different internal systems and had different processes and standards that don't easily translate across company lines. Between organizations and their end users (customers, members, media contacts, and industry watchers), the basic connectivity that would enable the end-user computers to communicate with the organization's computers was largely absent — until the advent of the Web.

With the Web and the open standards it engenders, the way is now clear for making the transition to computer-based business processes among organizations and their partners and end users. The Web provides both the basic connectivity for end users and a "lingua franca" that enables different computer systems to communicate. Today, business functionality can extend far beyond the office to reach anyone with a computer and an Internet connection.

The way is clear, but the road is still rough. But as you see in this chapter, you can go a long way toward making all your business electronic by applying a content model to computer functionality.

What Is Functionality?

In the software world, a computer-based process is known as *functionality*. The functionality that a computer application offers casts a human process into a series of human-computer interactions using a user interface. From the standpoint of the organization, you can say that functionality is the way that you do business by using a computer.

Simply put, functionality is the stuff that you *do* on the computer. Unlike text, sound, pictures, and motion, which you experience (by reading, hearing, or seeing), you *do* functionality (by requesting or responding to something). A user interface represents functionality as a set of buttons, menus, dialog boxes, and other controls that give you a particular capability. Microsoft Word, for example, provides table functionality. The functionality is available from the Table menu at the top of the Word window. If you choose the Table ⇨ Insert Table command, Word displays a dialog box full of buttons and other controls that enable you to specify how you want the Insert Table functionality to perform. On the Web, functionality is programming code that lives in a Web page or on a Web server. It does useful things such as performing calculations, allowing you to submit data via a form, carrying out financial transactions, verifying your credit card information, and so on.

Functionality's taken the following two major steps over the course of computing:

- It's moved away from monolithic programs, where all the functionality is built-in, toward mix-and-match programs, where the functionality exists in small chunks known as *objects*.
- It's moved away from applications with the single purpose of presenting functionality toward Web sites in which functionality and content intermingle and become hard to tell apart.

Taken together, these two steps enable you to treat functionality as a form of content.

Monolithic vs. Mix-and-Match Functionality

In the early days of computing, all the functionality of an application was built into the application. The application's programmers coded all their own functionality from scratch. Today, much of the functionality that an application provides comes from outside the application itself. It's created and packaged into functionality objects that programmers then glue together into an application.

Computers entered the organization one process at a time. First came accounting and manufacturing systems. Then came spreadsheet and writing applications. Today hundreds of different types of computer applications are used in organizations. Because of the way they entered organizations, however, these applications were often unable to communicate with other applications, they duplicated the functionality of other applications, and couldn't easily integrate with each other. Because of this incompatibility, much of an organization's functionality is today locked in silos of distinct applications that don't easily operate with one another.

Most newer applications take solid steps to break down the borders of different applications. Communication standards (such as XML) provide a standard accepted map among the different vocabularies and methods of various applications. Open programming standards such as the Application Programming Interface (API), object-oriented programming, and Remote Procedure Calls (RPC) ensure that interoperability is built directly into the core of modern applications.

The result of the newer application models is that knowing or caring about the application that hosts a particular functionality becomes less and less important to you. You merely need to know how to connect to, and use, the functionality that you need. Conversely, building complex applications that combine the best functionality and data from a variety of sources is becoming easier and faster than ever.

Today, programmers pack pieces of functionality into programming objects. Objects serve as a standard way to package and transfer functionality. Every object programmer knows exactly how to access the functionality in an object and to include the object in her own programs. In fact, many programs (especially those on the Web) do nothing more than draw together a set of previously created objects into a customized look-and-feel.

My notion of content components shares a lot with programming objects. In fact, content components serve the same basic function as programming objects. They both provide a level of isolation between the person who creates them and the person who uses them. If I create a content component or a programming object, for example, I give it a set of attributes that enable it to take on a life of its own. Later users of the component or object find it and use it based on these attributes (that is, the metadata for components and properties and the methods for objects). They know that the component or object serves the purpose they intend for it and don't need to look inside the component or object to see what it contains or how it works.

Cross-Reference

In Chapter 23, "Designing Content Components," in the section "Components are like objects," I provide more information on content components and their relation to objects.

Functionality that you package in objects can have the following characteristics:

✦ **Segmented:** Functionality can come in chunks as small as you like. (*Small* in the computer world means not taking up many bytes, not requiring much of the computer's processing power to run, and not requiring a lot of supporting hardware or software to run.)

✦ **Shared:** You can easily locate and use the functionality that someone else created apart from the application that they originally intended it for.

- **Encapsulated:** To use the functionality in your application or Web page, you don't need to know how the functionality works; you need only know how to invoke it and what sort of results it delivers back to you.

These three attributes of modern computer functionality pave the way for treating functionality as content.

Functionality Interspersed with Information

Before the multimedia age, the state-of-the-art computer application used a graphical user interface (GUI) consisting of buttons, menus, and a small set of other program controls that gave the user the capability to interact with a given program. The application's aesthetic was efficiency of space and parsimony of language. Programmers prided themselves on their ability to use as few pixels as possible to convey the meaning of each action. Long explanations and anything else that you might consider content was relegated to the system documentation.

Multimedia applications began to overturn this aesthetic, and the Web destroyed it. After users stopped working with data and started working with content, the assumptions of how to create an application shifted, too. Interspersing text, pictures, and sounds with application functionality became not only acceptable, but also standard. And although the small set of standard program controls never disappeared, a whole host of other controls appeared to deal with the interesting new needs of content applications. The text hyperlink and the hyperpicture, for example, weren't part of the classic GUI, but were added later to deal with the need to link from one piece of information to another.

In the old world, the premium was on parsimony. Applications had terse menus, well-ordered and trim dialog boxes, and buttons with one word or less (such as Del instead of Delete) on them. On the Web, a new aesthetic has taken hold. If the old style was Spartan, then the new one is Attic. On the Web, the menus can come with a paragraph of text. Dialog boxes are interspersed with pictures, ads, text, and links to other sites and applications all across the Web. The "buttons," in the form of linked images, stretch all the way across the screen. The result is a tremendous amount of mixing between content and functionality, and a blurring of which is which.

In the older software applications, the programming code that creates functionality was packed deep inside executable files (EXEs). On the Web, you can type computer code directly into HTML (today's basic language that you use to create Web pages) just as you can any other content. Although this mixing of content and functionality has caused untold thousands of all-nighters by frustrated programmers who pine for the good old days of neatly encapsulated programs, it's also provided the capability to create an entirely new way of thinking about an application. In the past, an application was a single, defined thing. Applications changed interface or functionality only once every year or two, when the creators made another product release. On the Web, not only can you change interface and functionality, but you're fully expected to do so. A good site delivers personalized functionality based on what it can infer about the user. The same functionality may show up in different contexts, and a particular function may even be represented with different controls, depending on the situation.

On the Web, functionality and content are mixed and matched to whatever extent the site owners can manage. It's a management issue. To mix and match content and functionality, you must be well organized. You must have a lot of structure in place that enables you to efficiently store and deliver your functionality where and when you need it. In short, you need a CMS.

Functionality objects enable you to treat functionality more as content. The new aesthetic, of content-rich applications that provide personalized functionality to the user, demands the use of functionality as content objects. Luckily, you can use the same tools and processes to manage functionality as you do content.

Managing Functionality Like Information

Content management consists of collecting, managing, and publishing content. I define *content* as information that you enhance with data and put to a particular use. I can now extend the definition of *content* to include functionality that you've packaged for reuse in objects or in blocks of programming code.

Cross-Reference

You can find more information about the definition of content management in Chapter 6, "Understanding Content Management." In Chapter 1, "Defining Data, Information, and Content," you can find more information about the definition of content.

To illustrate this notion, look at a piece of information and a piece of functionality that you must manage:

Suppose that Wingnuts, Inc., wants to make people aware of upcoming marketing events. *People*, in this case, means staff, partners, and customers. *Making aware* means listing the appropriate events on the company's intranet, on the Internet, and in e-mail messages.

The following *events* functionality is available to Wingnuts in a programming object:

- Listing event descriptions.
- Showing the details of a single event.
- Creating a new event.
- Deleting an event.

The information that the object can provide for an event is as follows:

- A name.
- A location.
- A date.
- A description.
- A picture.
- An owner.
- Target customer type.
- Marketing purpose.

You can imagine that, for the different combinations of person and outlet (which refers to audience and publication in CMS-speak), you must use the appropriate set of functionality and information, as follows:

- **Staff** should be able to list, show detail, and create events on the Internet or intranet. If any staffers are the event owners, then they should be able to delete them, too. Staff shouldn't be bothered by e-mail and should see all the information associated with an event.
- **Partners** see events only on the Internet site and in e-mail messages. On the Internet, they should be able only to list and show details of events. The information that they see on the Internet shouldn't include the marketing purpose of the event. In e-mail, they should see only a list of events, but each list item should contain a link to a page on the Internet site that shows the event detail.
- **Customers** (like partners) see events only on the Internet site and in e-mail messages and should have only two options on the Web, list events and show event detail functions.

How much difference exists between the way that you must manage functionality and the way that you manage the information? First, notice that you do need to manage both the information and the functionality. You need to publish both in the same three publications (the intranet site, the Internet site, and e-mail messages). You need to store both in a way that you can access a subset of their parts. Finally, you need to subject both to the same rules that determine which parts are available to whom and in what publication.

Both functionality and information are kinds of content that you need to create, store, and access in such a way that you can target and format them appropriately for publication. With functionality, as with information, you can tackle the problem of managing multiple publications by hand or by using a CMS.

Publishing functionality on the Web

Because you bring functionality onto a Web page via blocks of text, you can treat that functionality as just another type of content to publish.

If you look inside a Web page, then you see that everything in it is text. In HTML, for example, the following text block causes a button to appear on the page. The button, after someone clicks it, initiates a search:

```
<input type=submit value=Search>
```

If you want the same button on ten pages, then you simply include the same text block on each page.

The next text block calls a Java applet onto the page. An *applet* is a “miniapplication,” or, as I call it in this book, a functionality object. The following code passes the applet all the information that it needs to do its job:

```
<applet code="Events.class" width="468" height="234">
<param name="User Type" value="Staff">
<param name="Pub" value="Intranet">
</applet>
```

The `<applet>` tag tells you which Java applet you want on the page, and the `<param>` tags pass the applet all the information that it needs to perform its functions correctly. In this case, you can probably guess that this applet lists events and can behave differently based on the user type and publication. You really see no difference between storing the block of text that invokes this applet and storing a block of text that describes an event. Moreover, no extra process or software is necessary to embed this block of functionality in a container that tells a CMS what the functionality is for, to whom it applies, its status, and on what pages to include it. The same process that you use to create a content component from a block of information you can also use to create a component from a block of functionality definition. Whether it’s delivering functionality or information onto a page matters not at all to a CMS.

Of course, I don’t discuss a lot of details here, such as how you package and deploy the events applet and exactly how you go about storing complete code blocks versus dynamically filling in the correct parameters. But that’s not my task at the moment. The point that I hope I’m making here is that if you store a functionality generating text block and call it into publications, then you’re making it into just another kind of content component.

Even at the highest level, the similarities are solid. Whether you’re managing information or functionality, you make an agreement with the creator or owner that enables you to capture her valuable creation; you devise a way to store what you collect; and you devise a way to deliver the creation electronically to the people who need it.

As you see in more detail throughout this book, a CMS provides the following capabilities — and more — for effectively managing and delivering functionality:

- **The CMS repository** provides a place to store the code chunks that provide the functionality.
- **The CMS metadata** facilities provide a way to tag functionality so that you can effectively store, locate, and select it for use in a particular page.
- **The CMS workflow** system provides a way to wrap the functionality in a process for reviewing and approving it. Many CMS systems actually supplant the source code management systems that programming organizations traditionally use.
- **The CMS templating system** provides the capability to locate a particular chunk of functionality on a particular page based on the user, or any other rules that you may use to place information on a page. In addition, the templating system enables you to customize the user interface of the functionality based on rules that you design. You may present the beginner's interface or the advanced interface, for example, based on the perceived computer savvy of the user.

Cross-Reference

In Chapter 12, "Working within the Organization," in the section "Tracking Information Flow in the Organization," I talk more about the way you can go about collecting functionality. In Chapter 10, "The Branches of Content Management," you find a number of examples of how you can use a CMS to manage e-business functionality.

Where information and functionality diverge

Where information and functionality management diverge is in the details, as the following list describes:

- **To collect content,** you work with authors or bulk content sources. To collect functionality, you work with application developers. Instead of making an agreement with a person to deliver a certain type of information, you make an agreement with a programmer to supply software code segments that create the functionality. You also make an agreement with a system administrator, who assures that, when the code runs someday, the database or other system that the code works with is available and ready to respond.
- **To manage content,** you send it through a variety of workflows that ensure its accuracy, connection to other content (metadata), and relevance. To manage functionality, you send it through a variety of workflows to ensure that it works as you want (that is, with no bugs), is the most current version available, and can connect to its data sources.

- **To publish content,** you create templates that draw in relevant content chunks and navigation. You ensure that the content always makes sense relative to the other content that surrounds it. To publish functionality, you create templates that draw in and integrate code chunks with the surrounding code in the template. You ensure that any supporting code registers on the Web server and is ready for the code that you draw into templates to call it. You also ensure that all connections to other servers or data sources are enabled on the server.

By and large, these details fall below the radar of a CMS. The biggest hurdle to overcome is the mindset change that your developers and organization may need to undergo. They, too, must come to believe that functionality is a kind of content to collect, manage, and publish just as information is.

Tip

People frequently ask when you would put your functionality code into the CMS repository and when you would not. One reason not to put code in the repository is if programmers are used to a different, or more sophisticated, version control system, but the CMS that you choose or build doesn't offer that level of sophistication.

Summary

Functionality is the capability to interact with a computer to perform a particular task. You create functionality by programming it using computer code and by presenting it in a user interface. Here are some things to keep in mind about modern functionality and how it relates to content:

- Where previously functionality was trapped within massive programs, today it's segmented into lightweight objects and code blocks that you can share and reuse in a variety of contexts.
- Where previously functionality stood separate from information, today it so intermixes with text, pictures, and other types of information that determining just where the information ends, and the functionality begins, is often difficult.
- In construction and in presentation, functionality shares so many of the attributes of content that you have no reason not to treat it as just another type of content.

In the next chapter, I end the discussion of what content is by handling a few of the more philosophical aspects of the nature of content.

✦ ✦ ✦

CHAPTER

5

But What Is Content Really?

◆ ◆ ◆ ◆

In This Chapter

A CMS manages context as well as content

How content purpose leads to organization

How naming is the basis of content management

A trip from data all the way to wisdom

◆ ◆ ◆ ◆

I've spoken with any number of people who believe that the basic agenda for content management as I lay it out in this book is flawed. You can't, they often say, separate content from its context without destroying its meaning. Furthermore, you can't, they contend, just create information in little chunks and move it all around and expect it to still communicate in the way the author intends it to! In addition, they say, the job of the author is never to create neutral information that can serve anyone at anytime. Authors need to know with whom they're communicating and why. Finally, they contend, each kind of publication is different, and each demands its very own, very different approach to content. How can you possibly create a content chunk that can work as well in any publication?

To all these objections, I say, "You're absolutely right! Luckily, that's not what a content management system is trying to do." If you expect your CMS to enable you to ignore context, communicate to everyone regardless of their perspective, or ignore the differences between publications, then you can stop reading now. If, however, you expect your CMS to help you understand and manage the various contexts that you expect to create, to communicate with the range of people whose perspectives you understand, and to create the best fit between your content and the particular publications that you want to produce, then read on.

In this chapter, I deal with the relationship between content and context and discuss some of the other deeper issues behind my notion of content, such as naming and knowledge.

Content, Context, and Meaning

Content, context, and meaning are tightly intertwined. Still, you can separate content and context and then automatically mix and match them later. Content and its context so intertwine, however, that you could just as well call content management *context management*.

I'm not, however, creating only a book. As I must (given my subject matter), I'm actually creating a CMS that can publish all or parts of this same body of content into a book, on a Web site, as e-mail messages, training materials, white papers, and so on. To take this CMS approach, I can't afford to lapse into unconsciousness about the context of my various publications. Rather, I must remain very conscious of, and explicitly account for, the needs of the various contexts in which my content may find itself. As one example among many, consider the name of what you're reading right now. Are you reading a book, a Web page, a white paper, or an e-mail? How am I to account for your context as I write? Here's how I do it: As I name the work, instead of typing one word or another, I enter the following tag where the name of this work should be

```
<Variable Name="TheWork"/>
```

This tag says, "The name of the work goes in this spot." Later, as I create the site, or the chapter, or the course, the appropriate name is filled in programmatically by my CMS publishing templates. So, far from ignoring context, I must pay close attention to it and explicitly account for it.

Make no mistake, accounting for context is much harder than ignoring it. On the other hand, at a certain volume of information and publications, recasting your content by hand is more work than setting up a CMS that ensures that your various contexts (publications, that is) are all as well served as possible.

Cross-Reference

In Chapter 8, "Knowing When You Need a CMS," section "Estimating Complexity," I provide a way to make a ballpark estimate of when the trouble of content management becomes worth the effort.

To mix and match content automatically, you must somehow reduce the complex relationship of content and context to a simple set of rules. You can then turn the rules into a computer program that a computer can understand and run.

Note

Programming rules about context is easy, compared to the effort it takes to understand context well enough to think up the rules.

I don't believe that anyone can create a set of rules that state how context and content interact *in general*. Fortunately, you don't normally write general rules. Typically, in a content management project, you must write rules governing the specific and limited relationships among the following:

- A relatively small set of context elements.
- A small set of content elements.
- A small set of audience types.

This task is hard, but not impossible. And you don't need to get it right the first time. In the electronic channels, you can try a rule and see how people respond.

The rules behind the naming of this work, for example, are fairly simple. Each publication type bears a different name. In the XSL programs that I wrote to transform the content into various publications, I inserted the appropriate rule for whether to include — or how to include — certain types of content based on the type of publication I'm creating.

A more complex set of rules may create changes in the layout and content on a Web page based on the user. For such rules, you begin by stating in plain English what you want to happen for each kind of user. A social services Web site, for example, may follow these rules:

- For contributors, I want the pages to feel solid, professional, and contain a medium amount of content.
- For the staff, I want the pages to look fun and intriguing but contain the full content.
- For clients in need of service, the pages must look accessible and easy to use; they should contain a minimum of information.

Next, you may want to specify the elements of the pages that the rules affect, driving the rules down a level of specificity, as follows:

- For contributors, use banner 1, navigation scheme 1, background 1, CSS style sheet 1, and the content elements 1, 2, 3, and 4.
- For staff, use banner 2, navigation scheme 2, background 2, CSS style sheet 2, and all content elements.
- For clients, use banner 3, navigation scheme 3, background 3, CSS style sheet 3, and the content elements 1 and 2 only.

Finally, you specify the programming-level rules that implement the intention of the plain English and the element level rules, as follows:

- Detect the referring domain and login status to determine the user type (1, 2, or 3).
- Redirect to the appropriate template (Tpl-1 for user type 1, Tpl-2 for type 2, and so on).
- In the templates, include banners, navigation areas, backgrounds, and style sheets by user type. Select content elements as specified in the user configuration database.

This task is hard, but not impossible. And you don't need to get it right the first time. In the electronic channels, you can try a rule and see how people respond.

The rules behind the naming of this work, for example, are fairly simple. Each publication type bears a different name. In the XSL programs that I wrote to transform the content into various publications, I inserted the appropriate rule for whether to include — or how to include — certain types of content based on the type of publication I'm creating.

A more complex set of rules may create changes in the layout and content on a Web page based on the user. For such rules, you begin by stating in plain English what you want to happen for each kind of user. A social services Web site, for example, may follow these rules:

- For contributors, I want the pages to feel solid, professional, and contain a medium amount of content.
- For the staff, I want the pages to look fun and intriguing but contain the full content.
- For clients in need of service, the pages must look accessible and easy to use; they should contain a minimum of information.

Next, you may want to specify the elements of the pages that the rules affect, driving the rules down a level of specificity, as follows:

- For contributors, use banner 1, navigation scheme 1, background 1, CSS style sheet 1, and the content elements 1, 2, 3, and 4.
- For staff, use banner 2, navigation scheme 2, background 2, CSS style sheet 2, and all content elements.
- For clients, use banner 3, navigation scheme 3, background 3, CSS style sheet 3, and the content elements 1 and 2 only.

Finally, you specify the programming-level rules that implement the intention of the plain English and the element level rules, as follows:

- Detect the referring domain and login status to determine the user type (1, 2, or 3).
- Redirect to the appropriate template (Tpl-1 for user type 1, Tpl-2 for type 2, and so on).
- In the templates, include banners, navigation areas, backgrounds, and style sheets by user type. Select content elements as specified in the user configuration database.

Although intentionally simple, this example indicates that you must first understand the contexts that you want to create for your content. You can then begin to reduce that understanding to a set of rules that programmers can implement to produce the appropriate context for your content and audiences.

You could just as well rename content management as context management. To be successful, a CMS must manage the surrounding context of the content as well as manage the content itself. The context must always support and enhance the content that it surrounds. The trick to content management is to really know your content, its contexts (or publications), and its users (or audiences). If you know these three very well, the rules for how they relate follow naturally. If you prepare yourself to try, evaluate, and try again, then your rules will continue to get better.

Content Organization Starts with Purpose

To turn information into content, you organize it around a *purpose*. That purpose leads to the schemes and categories into which you can fit the information. The purpose is the start of the organization process. The process is complete after you construct a full system of names and categories into which the content can fit.

To make it useful, you first must get your system organized. Every large city has a day-labor corner where people stand around at dawn looking for a day or two of work. Suppose that you go down to the day-labor corner with a bus and pick up 50 workers. As they ride together, they begin discussing what you may have in mind for them. Rumors and opinions fly through the uncoordinated mass of workers, but no one really knows. After you finally arrive at a cleared-off piece of land, you gather them together and give them a purpose: "We're going to build a house!" The unconnected mass of workers is now connected — they all know what their purpose is. But this knowledge is the beginning, not the end of the story. If you were to stop there, then you wouldn't end up with much of a house.

So, you bring out a plan that shows each person what to do and how. Some of the workers are plan readers, some are nailers, some are cutters, and some are measurers. All receive detailed instructions about how to carry out a given job. In addition to the overall purpose, each worker now has a specific purpose that feeds naturally into the overall purpose. Finally, you instruct the workers on how to interact. You give them instructions on exactly how to build each part of the house as a combination of lumber, saws, nails, plan readers, measurers, cutters, and nailers. Now, you can build a house!

You can think of the lumber in the previous story as information. Although a lot of effort's already gone into its creation (felling, milling, and so on), if it's just lying in a pile on the ground, it's of no use. Similarly, you must put a lot of effort into information just for it to exist. But information in a pile on someone's hard drive is of no use. To put it to use, you start with a purpose for it. That purpose begins to transform the way that you see the information. It's no longer just "stuff." It's a means to the end that your purpose intends. You can measure each piece of information against that purpose. But that's just the beginning. You can compare the construction workers in the preceding story to information workers who measure, cut, and nail the information into a set of content components that they all target toward your purpose. Without that purpose, your workers have no target, and without an organized approach, they can never reach the target.

Organizing your information into content involves the following activities:

- **Deciding on the right types of information** for your particular purpose. Just as the type of building influences the types of lumber that you need, the purpose of your content system influences the kinds of information that you need.
- **Deciding on the right sorts of people** you need to process the information. Just as the types of workers vary by building, the type of staff that you need varies with the kinds of content you manage.
- **Understanding the information** you have, can buy, or can create. You don't build a house only from the lumber that you have lying around. Similarly, simply taking what information you already have and putting it into a system isn't enough. You also need to ask, "What information do I need that I don't have yet?"
- **Finding ways to standardize your information.** Standardization ensures that any two items of the same class are of the same basic structure. Just as a lumber mill cuts wood into standard shapes and sizes, so, too, do you need to mill information to standard types.
- **Finding ways to track and easily assemble your information.** Building a house involves plans and materials lists that specify what goes where and when. A CMS needs metadata and templates to help you manage information. You may, for example, need to add a status field to a video review to indicate when it's ready for display. A template reads the status field to decide whether to put the video review on a page.

Much of this book concerns the details of creating an organizational scheme for your content. Not the actual scheme to use — only you can figure that one out — but the principles and elements of schemes that you must pay attention to.

Content Is Named Information

Names provide a simple, memorable, useful container in which to collect and unify otherwise disparate pieces of information. You can look at content management as the art of adding names to information.

Content is information that you organize for a specific use. The key to the organization and use of content is *naming*. I had this concept pounded into my head again recently. It was me, my team, and our clients (a major .COM company). We were preparing for the first big presentation to the executive staff on "What we'd been doing." Lots of good ideas and bits of presentation were floating around, and as the presentation approached, we began to feel pretty good that we'd actually been doing something. Still, we had no unity to our approach. Different subgroups had simply merged their slides into a single PowerPoint file. After the last run-through, a half-hour before the big event, someone finally nailed it. "We need a name for this thing!" she said. A dawn-breaking eyebrow raise spread through the room. Within minutes, we settled on Grace, one of the characters in our user scenarios, as the name of our presentation. Except that it wasn't just Grace. It was GRACE for *G*reat *R*elationships *A*nd *C*ustomer *E*-tailing. The entire process crystallized, and we all suddenly knew how our individual parts related to the whole project. The name stuck instantly, and we went into that presentation not as a collection of separate projects stuck together by PowerPoint, but rather as a team all working together on GRACE.

Names provide a simple, memorable, *useful* container in which to collect and unify otherwise disparate pieces of information. I exploit this quality of names over and over again in this work. In fact, you can say that content and content management are no more than discovering and successfully applying names to information.

Names have always had a strange and powerful hold on us. To this day, many people refuse to speak the name of a disease lest they contract it. In the Jewish tradition, the name of God may not be spoken. In Native American cultures, a person's true name is crafted to speak a deep truth about them. This true name is never used lightly. In modern consumer culture, the power of the name runs through the phenomenon of brand identity, where a well-recognized name is valued above all else in a company's portfolio of assets. Finally, the mystique and power of law, medicine, and many other professions derives from the set of names (in the form of professional jargon) that each uses and that no one else understands.

Why are names so powerful in human life? It is because names capture and contain a potentially vast amount of information in a simple-to-use form. Those who "know" the name hold a lot of understanding that others lack. Why are names so important to the study of content? For the very same reason: Names help you capture and contain a vast amount of information and make it simpler to use.

In effect, names "datatize" information; that is, they provide a small snippet that can stand in for the larger and less manageable whole chunk of information.

After you gather a standard set of names for your information, you can start to create simple rules and procedures for selecting and displaying content. These rules are just the simple sort that a data-loving computer can handle.

Naming, in fact, lies behind all of the major topics that I discuss in this work, as the following list summarizes:

- I discuss the notion of breaking information into discrete *components*. The way that you do so is to invent names for the various types of content that you plan to manage and then create a set of individually named content components of that type. You can find more on components in Chapter 23, "Designing Content Components."
- I discuss *metadata*, which is no more than a set of names that you associate with components to help find, relate, and use them. You can find more on metadata in Chapter 20, "Working with Metadata."
- I discuss a *content domain*, which is simply a name for the universe of information that you're trying to control. You can find more on the subject of the content domain in Chapter 26, "Designing Content Access Structures."
- I discuss the *hierarchy*, which is no more than an outline of names on which to hang each of your components. You can find more on hierarchies in Chapter 26, "Designing Content Access Structures."
- I discuss *indexes*, which extract the significant names from your content and put them all in a well-organized list. You can find more on hierarchies in Chapter 26, "Designing Content Access Structures."
- I discuss *markup*, which is a standardized system of naming that you can apply to content to make it usable to anyone who knows the markup's naming system. You can find more on markup in Chapter 28, "What Are Content Markup Languages?"
- I discuss the *Metatorial guide,* where you invent the system of names that you use throughout your system.
- I discuss the *metatorial framework*, where you invent the system of names that you use throughout your system. You can find more on the metatorial framework in Chapter 20, "Working with Metadata."
- I discuss *content collection*, which is the process in which you assign names to previously unnamed information. You can find more on content collection throughout this book as well as in Chapter 31, "Building Collection Systems."
- I discuss *repositories*, in which you pool the names together and make them accessible in a database or other management structure. You can find more on repositories in Chapter 32, "Building Management Systems."

- I discuss *publications*, which are named subsets of the content that you control. You can find more on content collection throughout this book as well as in Chapter 22, "Designing Publications."
- I discuss *workflow*, which is a set of named processes for the flow of named content components between named preparation and delivery stages and named staff positions. You can find more on workflow in Chapter 27, "Designing Workflow and Staffing Models."

From Data to Wisdom

The contrast between data and information is all you really need to know to manage content. You take the methodologies of data processing and wrap them around human-created information to create information methodologies. Still, to put content in the context of the wider world of communication and meaning, I'd like to reach beyond the basics, moving from data, the most concrete communication, to wisdom, the most abstract.

Data is raw and discrete

Data is unprocessed and uninterpreted. Data is the starting place. All the interpretation and most of the meaning is yet to come. Think of the data behind a payroll system for a small company: 100 employee names, 1,000 deduction entries, 10,000 time entries and 1,000,000 calculations before 1,000 numbers can go on 100 checks. The data here is a very large number of small, uncooked ingredients. You don't need to know or understand much about any particular datum. Only after they're cooked together into a savory stew (read: processed) do they yield interest. Now, in the payroll system as a whole, you find plenty of processing and interpretation, but it's all in the software and accounting staff, not in the data.

Data comes in *snippets*. Each snippet is complete unto itself and doesn't rely on its neighbors for its meaning. Each datum stands alone as one whole integer, string, bit, date, or what have you. Because they're discrete, you can disassociate each datum from the others, use it interchangeably with any other of the same type, and create any sort of complex processing routine that works on the data.

Table 5-1 describes the more common types of data that you find in Microsoft's version 7.0 of its SQL Server database.

Table 5-1
Common Data Types in the Microsoft SQL Server Database

Data type	***Description***
Binary	Fixed length binary data (8,000 bytes max)
Bit	Either 0 or 1 (minimum 1 byte)
Char	Fixed length, nonunicode data (8,000 bytes max)
Datetime	Date and time data with an accuracy of 3.33 ms. (8 bytes)
Decimal	Fixed precision data with a range of –10^38 to 10^38 –1 (5–17 bytes depending on the precision)
Float	Approximate precision data with a range of –1.79E+308 to 1.79E+308 (4-8 bytes)
Int	Integer data with a range of –2^31 to 2^31 – 1 (4 bytes)
Real	Approximate precision data with a range of –3.4E+38 to 3.4E+38 (4 bytes)
Smalldatetime	Date and time data with an accuracy of one minute (4 bytes)
Smallint	Integer data with a range of –2^15 to 2^15 – 1 (2 bytes)
SmallMoney	Currency value with a fixed scale of four (4 bytes)
Text	Variable length, nonunicode data (2^31 - 1 bytes max)
Timestamp	A value unique to the database and updated whenever a column changes (8 bytes)
Varbinary	Variable length binary data (8,000 bytes max)
Varchar	Variable length, nonunicode data (8,000 bytes max)

You don't need to be a programmer to see that the form for each data type is strictly defined—so strict, in fact, that if you try to disobey and call a Char a Bit or a Real an Int, your program can go down in flames. These discrete info snippets that I call data are collected into named buckets called *database fields*, or *columns*. The assumption that programmers must make to ensure that their programs work is that any operation that you can perform on one snippet you can perform on any other in the same bucket—or, in any other bucket of the same type, for that matter.

On the other hand, look at the *Text* data type. It's not nearly as restrictive as the others are. It can contain anything up to about a couple billion text characters. The data type *Binary* is even less restrictive; if you have bits and less than 8,000 bytes, you can put it in that bucket. So just about anything that you can make into bits, including stories, sound, pictures, and motion, you can put in a bucket and treat as data. Well, then, isn't everything that you communicate just data?

If the answer to this question were yes, then you'd have no need for this book. The methods that are currently in use would prove adequate to deal with the richness of human communication. Obviously, I believe that the answer is no — everything is *not* just data. I believe that the confusion here is that, although you can represent any communication as data (a series of ones and zeros stored in a database), if that data is information, you can't assume that any operation you perform on one info snippet, you can also perform on any other.

From another standpoint, you can say that, although you can represent all communication as data, it isn't all *just* data. It's often much, much more.

Data is nondiscursive and out of context

By design, data is too raw and fragmented to ever form the basis of a conversation. In fact, the whole point of data is to make information so raw and discrete that no conversation is necessary. What this situation means is that a datum has no life of its own. It's always an object, never a subject. The same qualities that make the datum easy to work with preclude it from ever being interesting itself.

A short list of data types can explain most of what a programmer needs to know to work successfully with data. Contrast this list with what you need to know to work with the following sentence: "Pick me up at 8 and don't be late." You have no little table that you can look in to see what this sentence means. In fact, without a certain set of life experiences (and perhaps a love of the Big Bopper — the younger of you, read on for an explanation), you'd stand little chance of making use of the sentence. The difference is that data is stripped of any connotations or context. It is content reduced to so simple a form that a dumb program can safely and expediently handle it. The sentence, on the other hand, is an unsanitary rat's nest of connotation and context. The way to use a datum is closely prescribed and easily determined, simply by knowing the type of the data. The way to "use" a sentence is nowhere prescribed, and, at best, you can only guess at it.

The long and the short of data is that it's easy to work with, but it carries very little interpretation or meaning. Data reminds me of the story about the guy who lost his wallet on the north side of the street. As I came across him, it was already night, and he was looking carefully for his wallet — but on the south side of the street.

"Why are you looking over here?" I asked. "Didn't you lose your wallet on the other side?"

"I did," he replied. "But the light is better over here."

Can a Computer Ever Interpret Like a Human?

So, what do I mean by a computer interpreting the way that humans do? You need to contrast the brain and modern computers to see what I mean. First, I must discuss basic design. Computer hardware is designed to run through a sequence of simple instructions, each one chained to the next. Something known as the *central processing unit* (*CPU*) is the means through which a set of instructions run to create a particular result. Computers are just entering the age of parallel processing, where a small set of processing units divide the labor to speed things up. Generally, however, all instructions flow through a small number (usually one) of processors.

The brain, on the other hand, is so massively parallel that it's in a wholly different league. By parallel, I mean that there are a large number of processors simultaneously working on the same "computation." Each neuronal processing unit (and the brain has billions of them) performs a very simple function – it sums the activation that its input neurons provide, and if they add up to enough activation, it fires some activation off to its output neurons. What supplies the complexities in this simple system are the enormous number of input and output connections that each neuron makes and the effect of chemicals in the brain that subtly, but decidedly, change the way that the activation cycle proceeds. So, in the brain, a huge number of very simple processors intensely interact to achieve a result (as opposed to one or a few processors performing a set of instructions in a computer's CPU).

Enough of a difference exists between how computers work and how the brain works that many good thinkers conclude that computers can never hope to do what a brain can do. Maybe so, but I'd say that *what* computers do is more important than *how* they do it. To probe further, I must move on to software.

At first, monolithic software programs ran from start to finish in a fairly predictable way. Today, an event-driven model, where some user action invokes a large set of interacting modules, replaces this original approach. A button click, for example, triggers a module, which triggers other modules, which trigger still other modules in a complex hierarchy of delegation of responsibility. Each module performs some tasks, changes some data, and then passes results up to the module that calls it. This process, in the most rudimentary way, mirrors what the brain does. Regions of the brain specialize in particular processes, and an entire interpretive cycle involves the collaboration of a variety of regions. Activation in one region triggers activation in allied regions, which trigger still other regions. But here is where the comparison ends. Computers are missing a number of fundamentally important characteristics that define interpretation in the brain.

In computers, modules form a hierarchy, with the parents calling and controlling the children. In the brain, no such hierarchy exists. A very tangled and recursive relationship exists between regions. You find no parent-child and no caller-called relationship. Activation spreads. It's sometimes amplified, sometimes squelched, but it is never just on or off, as calling is in a computer program. Regions specialize, but no one can yet predict which regions participate in which events, and no one knows whether anyone can ever come to predict that participation. So much interconnection goes on that any region may end up recruited in the processing of any event, based on relationships between that event and

Although data is a lot easier to pin down and discuss, information is the normal mode of life. Data is the newcomer. The quintessential Western idea that you can quantify and categorize meaning isn't normal even for the most scientifically minded, northern European, white, middle-aged male. You must learn it. Computer applications that work with data are commonplace. But as anyone who's worked much with computers can tell you, you must change your mindset to match the computer's total ignorance of what you intend and the context in which you're performing an action (not to mention its lack of basic politeness). Why? The computer's lack of context is simply leaking through to the user interface. Computers are the ultimate low-context culture. Computers force you to shed your entire context and leave nothing unsaid.

Computers, perhaps, were invented to work with data, but they're now fully pointed at information. Why? Because users have collectively gotten to the point where they can begin to see the possibility of computers moving toward human mindsets rather than people always needing to go to theirs. Users want computers capable of working with information and not just data.

The information technology (IT) departments of the world, whose names by rights should be data technology, are staffed with people who know exceedingly well how to create and run data systems. As the specter of information rises before them, in all of its dirty complexity, human interference, and esthetic considerations, however, they shrink back. They respond by either mistreating information as data (thus denying or ignoring the problem) or by throwing their hands up in disgust and declaring the project impossible. And, from their perspective, it is. These people know how to reduce information to a strictly defined set of small, discrete buckets. To succeed in an information project, however, IT professionals need to change perspective and understand how computers can facilitate the creation and handling of continuous, context-rich, intensely human information. Having learned the data craft myself, I'm the last to condemn it. The reductionist methodology created all of science. In particular, it was responsible for the vast power of computers. Only now are people beginning to use that power to grasp and harness information (and not just data) by using computers.

Computers deal with facts very well because facts fit the basic way that computers work—put the world into discrete little buckets and work on these buckets by using long chains of very simple instructions. I suspect that a basic overhaul in this model is in the offing. (People seem to be culturally and technologically ready.) In fact, the very concerns that I'm discussing may drive the overhaul. Until then, you can expect that whatever people make computers to do, they're sure to do it with the same lack of subtlety that they now exhibit. In this book, I try to make the best fit between the constraints of information and the capabilities of computers.

In summary, information is fundamentally messier than data. It contains enough human qualities to make it hard to parse, obscure in its interpretation, and complex to handle and use. Successfully working with it requires a different mentality and skill set. But when has that sort of thing ever stopped anyone before?

Knowledge and wisdom can be information

Data is material facts; information is matter-of-fact; knowledge is a matter of dispute; and wisdom is nonmaterial. Although both data and information have a face value, you must synthesize and extract knowledge and wisdom from a wealth of communication and direct experience.

> "*Knowledge*," according to Merriam-Webster OnLine (`www.m-w.com`) is as follows: "(1) : the fact or condition of knowing something with familiarity gained through experience or association . . ."

Knowledge, by this definition, is inside a person. It is a mental state, not a communication. So, although today's computers can conceivably store information, they can't store knowledge. But why can't computers store knowledge if books can? Books can't have a mental state anymore than computers can, but aren't books where people currently store the world's knowledge?

Both books and computers store information. If information that the book presents to a person inspires a state of knowledge, then great — the storage and communication was successful. The communication from the book or computer screen doesn't cause the knowledge to happen; rather, the person in conversation with the communication causes knowledge to happen.

If you don't buy this argument, then I've proved my point. I'm trying my hardest to give you some knowledge that I have (or think I do). Despite my best efforts, however, I can't do it. To my mind, codified knowledge (that is, knowledge that's written or otherwise recorded) is simply information. This concept is a good thing. It means that codified knowledge is manageable by using the same techniques as you use for information. The act of knowledge, on the other hand, is an interpretive act and is, at least for the moment, reserved for humans and maybe a few animals.

Knowledge happens within a *domain*. I know about cars, information, human nature, and God. Wisdom, on the other hand, is knowledge that transcends domains.

Again, according to Merriam-Webster OnLine (`www.m-w.com`), wisdom is defined as follows:

> "a : accumulated philosophic or scientific learning : KNOWLEDGE b : ability to discern inner qualities and relationships : INSIGHT c : good sense : JUDGMENT d : generally accepted belief <challenges what has become accepted wisdom among many historians — Robert Darnton> . . ."

Wisdom encompasses discerning inner qualities, exceptional insight, and good sense. Wisdom is a highly synthetic process that's beyond simple knowledge. Consider any of the following pieces of folk wisdom:

The Wisdom Web

Wisdom is a funny thing. It's the highest human achievement in learning and interpretation. At the same time, it's so rarefied and abstract that it transcends the complexity of language and knowledge systems. Wisdom contains a distilled and simplified logic that's been codified over the ages in a variety of easy-to-learn and use systems. Take a look at two of these codified wisdom systems, for example: tarot and astrology. You find two schools of thought regarding these types of systems. The first says that they're baseless pseudostudies that have no part in a modern skeptic's life. The second says that they're mystical understandings, somehow tied to universal forces beyond human apprehension. I want to propose a third alternative: These systems represent true wisdom concerning the human condition that's been distilled and systematized over the ages and is now wrapped in an easily remembered, presented, and consumed form.

Look first at astrology. There needs to be no mysterious connection between the stars and your birthday for astrology to be called a wisdom system. Further, even if your birthday has nothing to do with your personality (which I believe) there is wisdom in astrology. The wisdom in astrology concerns human personality. You find in astrology elements of all the "types" that you meet in life. The basic personality traits combine in natural clusters. These clusters, which can make very subtle psychological distinctions in their combination, form a very comprehensive and precise profile of a person. A vast amount of interpretation is also captured and simply presented in the advice that astrology offers about the mixing of personality types in relationships. The cosmological nomenclature need not be literally true for the system to work. It need only serve as a system to organize and systematize, and access a set of personality profiles. The profiles need not be "True" characterizations of a person, to be of value. Rather their value comes from their ability to provide recognizable personality types that people can relate to and use to understand themselves and others. The heavens do not have to be the determinants of personality to still provide a subtle metaphor for understanding it.

Now take tarot. There does not need to be anything mystical about it to call it a wisdom system. Even if the particular cards you draw are random (which I happen to believe they are), there is wisdom here. The wisdom in tarot concerns the human condition in general. The cards of a tarot deck represent the archetypal sorts of situations, characters, and turns of events that one confronts in life. The user arranges a set of these cards in response to a question. The cards entreat you to compare life's archetypes against the situation you present. The cards that you draw and their relative positions are a heuristic (an aid to learning, discovery, or problem-solving), dredging up fears, possibilities, and deeper interpretations of a situation than you may arrive at alone. Tarot, viewed this way, isn't fortune telling, rather it's a simple way to help you to a fuller analysis of the possibilities in any unknown situation. The cards that you draw are not the answer; they're the cues that lead you to your own answer.

If you extract the knowledge of the human condition that is embedded in either of these two systems (and many others, to be sure), from the mystical claims, you have some very workable content. It is a matter of debate whether many wisdom systems are "True." That's a debate I don't even need to touch. Whether a particular wisdom system is true or not, it might still have a set of well-organized concepts that it can deliver in a way that provokes thought and change in a person's life.

Continued

Continued

Philosophy aside, there is a different kind of point that I would like to make. How do these statements about wisdom systems involve the Web? The distilled and simple logic of many wisdom systems make them easy targets for a Web site. And indeed, most already have sites that you can visit to use them. But what if you combined a number of these systems in a single site that could lead you through the right part of each system to guide you to a comprehensive examination of your issue? The secret to doing so is to create a metawisdom system that codifies how to incorporate and weight the various systems toward the solution to the issue that you present.

Say, for example, you're having a row with your spouse and want to know the cause and solution. After taking you through the requisite caveat page telling you that this process is a heuristic and not a predictive one, the site may help you characterize your issue as concerning two conflicting personalities put together in a tough life situation. You further define the personalities and the life situation, and the site then accesses the astrological wisdom base, to provide personality insights, and deals out a few cards from the tarot deck along with a suggested interpretation for the cards and their positions. The site tracks the tarot reading and the personality profiles you've generated and leads you through the process of sorting all this out toward a set of possible solutions to your problem.

A curious paradox is that you can unlink wisdom, in this form, from human interpretation and treat it in a way similar to what you normally do with data. Just as in data-processing applications, this wisdom application chunks wisdom into little buckets and automates their processing and presentation. Just as in data systems, the interpretation is completely in the hands of the operator, not the computer.

- A stitch in time saves nine.
- Necessity is the mother of invention.
- Every cloud has a silver lining.

What's interesting about these wise sayings is that, although their literal meanings may be true, their bigger truths transcend the bounds of their subject matter and provide insight into the universe at large.

Wisdom isn't just a statement of fact, as is data. It's not a matter-of-fact statement, as is information. It's not a mental state of understanding, as knowledge is. Rather, it's an ultimately synthetic act where one expresses deep understanding, simply and universally. Consider these other wise statements:

- The opposite of a correct statement is a false statement. But the opposite of a profound truth may well be another profound truth. (*Niels Bohr*)
- When we try to pick out anything by itself, we find it hitched to everything else in the universe. (*John Muir*)

- I do not know whether I was then a man dreaming I was a butterfly, or whether I am now a butterfly dreaming I am a man. (*Chang-tzu*)

Each distills and synthesizes vast realms of knowledge into simple statements. Often using contradiction to capture and coalesce worlds, they pack as much as is humanly possible into a few words.

Like codified knowledge, you can treat codified wisdom as information. After you record it, wisdom functions the same as any other information (even if it's pretty tinny if you compare it to live communication with a wise person).

The works of Lao Tzu, Socrates, or (in my personal opinion) Kurasawa (the Japanese filmmaker) are wise indeed, but you can still collect, store, and distribute them as if they were "just" information.

Why Does Text Get All of the Attention?

Did you ever notice that when people (including me) talk about the promise and future of computers they include sound, video, animation, and other fun stuff, but how they only actually go into detail about text? Why is that? Why is it that it is so much easier to talk about text as content — useful, organized information — and so much harder to talk about media as the same?

I'll never forget the time my wife and I visited the studio of a friend's husband. He had worked his way up through the ranks in Los Angeles and was now the main sound guy on a TV series I had never heard of about a car that could do just about anything. He showed us a bevy of late model, very expensive software and hardware for getting just about anything you might want out of a sound. He could decompose a sound into its constituent wave forms, pull it apart, mix it precisely with other sound components and create the exact sound of fist against flesh from noises that came from nothing remotely human.

He showed us the way he took the director's vague instructions for backgrounds and effects and turned them into an audio tapestry that was behind, and in support of, every movement in the show. It seemed that he had the art of sound down so well that nothing was beyond his reach.

Then, being who I am, I naturally asked him, "So how do you catalog and find the sounds that you start with?" His answer stood in the starkest contrast to the advanced technology he had been previously demonstrating. "Oh, I just know," he said. "I have a few collections of sounds that I know pretty well, and I just work from there." I was surprised. "No indexes, no searching for just the right effect through enormous catalogs. *No table of contents*?" I returned. "Not really," he said. "There are a few things like that, but most of us work from feel. We know sound and know where to look for it. There is no master directory of sound that you can just look in." "Besides," he said, "it's an art, not something you can just pick up from a book or something!"

Sound is possibly the most organized (in my sense of the word) of the media arts. Images (moving or still) are considered even less categorical. It would seem almost absurd to ask an animator or videographer what names and categories they are employing in their work. Most would scan you for a pocket protector, or a propeller on the top of your head and then dismiss you without comment. Some would undoubtedly attempt to school you on the nature of art — how it is intuitive and how any attempt to pin it down or box it in simply destroys it. And they would be right, sort of. The way most people have approached media to date has been from the intuitive, non-linguistic side. When it is right, it just feels right. Artists go only so far in explaining or categorizing their work. After that, it either works or it doesn't. Fortunately for some, and unfortunately for others, that is changing, and will continue to change. As more and more people are using computers to turn art into content, artists will have to face that there are ways to name and categorize art that will continue to advance against intuition.

Is this bad? I wouldn't be so bold as to say no. What I would contend, however, is that the two modes of being, the rational/linguistic and the intuitive/nonlinguistic are parts of a wider human unity that do not have to be pitted against each other. Just as the notes on a musical score (a naming and categorization system) help the musician achieve intuition into the art of music, other naming and categorization systems can help other artists achieve the same. And, of course, the reverse is true. As any good taxonomist will tell you, when you have gotten to just the right set of categories, it just feels right!

So, why then do most people focus on naming and categorization in text to the exclusion of other media?

- Text is codified language. Language is our main medium for organizing and communicating experience. We use language to name and categorize. No wonder it lends itself to this sort of analysis.
- The light is better on the text side of the street. With 1,500 years of experience, text categorizers are a bit ahead of the other communication disciplines.

Text is codified language

In his fascinating book *Breaking the Maya Code,* Michael Coe explains in detail how written language has been characterized in a variety of ways, but how it is now agreed to be simply codified language: "Writing is speech put in visible form, in such a way that any user instructed in its conventions can reconstruct the vocal message. All linguists are agreed on this, and have been for a long time, but it hasn't always been this way."

Of course, codifying language is not as simple as just writing down what you say. I was rudely reminded of this in writing this work. I had delivered much of this material as lectures. I figured that if I had the lectures transcribed, I would have a fair start on the writing — not.

Transcription problems aside, it is clear that text has the jump on other media types when it comes to naming and categorization. What are names and categories if they are not text? This is not to say that it is easy to categorize text, just that there is no big conceptual stretch to do it. There is a fair amount of evidence, in fact, to support the claim that the brain process behind the sort of naming and categorizing that we are aiming for is linguistically based.

To index a book, you scan it for important words and concepts (read names and categories). The words and concepts are usually explicit in the text — they are words and phrases that you can pull out and attach page numbers to. Of course, good indexers can pull out words and phrases that are implicit in the text, but even these terms are never far from the surface of the words. Now consider indexing a set of pictures. You would most likely invent a set of descriptive names and categories first, and then apply them to the pictures. None of the words would be explicit in the pictures, there would always be an abstraction and interpretation needed on your part to associate a picture with a word. Furthermore, you would have to educate each user ahead of time to show them how the words you chose relate to picture qualities. All in all, it is a tough process to apply even the most basic organizational methods to non-text information.

Text has the lead

Pictures and sound have existed a lot longer than text. They are embedded much more deeply into the fabric of humanity than is text. On the other hand, reason (another word for the process of naming and categorizing the universe) developed hand in hand with text. It's hard to know exactly how organized we were before writing because little record has been left. We do get a glimpse from the Greeks. When they first learned how to write, they spent a lot of time writing about what they did when they could not write. We know from their early writings, that for the Greeks, at least, reason was verbal before it was written. It was natural for the Greeks to extend their reasoning skills from verbal to written form. In these western roots as well as from other traditions, writing has been used to name and categorize the world. Confucius did nothing if he did not divide the known universe of social interactions into a rigorous set of categories that his followers could easily ingest and understand.

It is no surprise, then, that text has a well-established tradition of being named and categorized that the other media types do not share. Consider again the indexing of a book. There are professional indexers who do nothing but name and categorize text all day long. There are millions of librarians and archivists who spend all of their time making catalogs of written works. Published writing has rigorous, time-honed methodologies in place for naming and categorizing within and between works.

Now look again at the issue of indexing a set of pictures. It's hard to even know where to start. Do you index them by the colors used? By the subject matter? By the objects shown in the pictures? There are no standards at all to follow. No professions, no rules, no one to rely on for help. There is not even a received way to say where in a picture a particular index value appears. Do you give X,Y coordinates? Compass points?

Summary

Computers have no problem with data. I can add little to the codified knowledge or wisdom on how computers can better deal with these bucketized facts. On the other hand, I *do* try to add to our codified knowledge (not quite wisdom, I fear) about how computers can work with information. Keep these ideas in mind as you make your way through the less philosophical aspects of content management:

- Although the act of creation and interpretation must remain for the present in the hands of humans, after you create it, you can use the computer's considerable power to manage the information as well as its codified context and interpretation.
- Your job as a content manager is to organize the mass of information and functionality with which you're presented into a cohesive body of well-structured content.
- You are aided in your daunting task if you understand that you're in the business of choosing and applying names. These names reduce the complexities of context, interpretation, and meaning to the kind of data (in this case, *metadata*) that a computer can handle.

No one's asking you to make your computer become creative (well, hardly anyone). Everyone's asking you to get your computer to deal with vast quantities of information — to make that information accessible and to interweave it with other, "related" information. In the next chapter, I leave the subject of content and focus on the main event of this book: content management.

✦ ✦ ✦

P A R T

II

What Is Content Management?

✦ ✦ ✦ ✦

In This Part

Chapter 6
Understanding Content Management

Chapter 7
Introducing the Major Parts of a CMS

Chapter 8
Knowing When You Need a CMS

Chapter 9
The Roots of Content Management

Chapter 10
The Branches of Content Management

✦ ✦ ✦ ✦

CHAPTER 6

Understanding Content Management

◆ ◆ ◆ ◆

In This Chapter

What content management is and can become

Why I don't talk about particular CMS products

How content management relates to Web-authoring tools and dynamic Web sites

◆ ◆ ◆ ◆

I assume that most people come to this book because they want to know how to make large and well-managed Web sites. You can learn that here. In the process, I hope that you also find that content management isn't about Web sites, although that's where it's mostly practiced today.

Content management is about gaining control over the creation and distribution of information and functionality. It's about knowing what value you have to offer, who wants what parts of that value, and how they want you to deliver it. Knowing that, you can build a CMS machine to help you get the right stuff to the right people in the right way.

In this chapter, I put a definition around the phrase *content management*, relate it to the very young industry with the same name, and link it to some of the Web technologies that you perhaps are now using to deal with sites that have gotten out of control.

Defining Content Management

If you've previously heard about content management, it's most likely because you have connections to a large Web-development project. Today, that's where most of the interest and activity lies. As the Web moved past small, informally designed sites and into large, rapidly changing sites, the need for strong management tools became pressing. Product companies moved in to address this need and called their offerings *content management systems* (CMSs). If your only problem is to create and maintain a large Web site, you have reason enough to desire the strict structure and formal procedures of a CMS. Such a system helps you get and stay organized so that your site can grow and change quickly while maintaining high quality. The Web, however, is simply one of many outlets for information that organizations need to organize. And as the amount of information sharing between these outlets grows, the desire for an organized approach becomes an absolute need.

I've been giving talks and running seminars on content management for the last couple of years. I often ask my audiences what publications they're responsible for. The very large majority of the responses once were, "Web site only." An ever-rising number, however, now say, "Multiple Web sites and print publications and anything else that we can create from the same information."

At the highest level, content management is the process behind matching what *you* have to what *they* want. You're an organization with information and functionality of value. They're a set of definable audiences who want that value. This definition and the processes behind it work as well in other outlets as on the Web.

At first blush, content management may seem a way to create large Web sites, but on closer examination, it's in fact an overall process for collecting, managing, and publishing content to any outlet, as the following list describes:

- **In collection:** You either create or acquire information from an existing source. Depending on the source, you may or may not need to convert the information to a master format (such as XML). Finally, you aggregate the information into your system by editing it, segmenting it into chunks (or components), and adding appropriate metadata.
- **In management:** You create a repository that consists of database records and/or files containing content components and administrative data (data on the system's users, for example).
- **In publishing:** You make the content available by extracting components out of the repository and constructing targeted publications such as Web sites, printable documents, and e-mail newsletters. The publications consist of appropriately arranged components, functionality, standard surrounding information, and navigation.

If content management were only a process for creating large Web sites, using it for organizations with large Web sites would prove well worth the effort. For organizations with multiple large publications (even if they're multiple Web sites), content management becomes a necessity. A deeper reason exists, however, as to why content management is important. It's not just the common forms of information that I call content that you must collect, manage, and deliver. It's all business. As organizations begin conducting their business electronically, they open the same Pandora's box that they open as they begin delivering information electronically—namely, one that invokes the following questions:

- How do I break my business down into electronically deliverable parts?
- How do I make sure that I know what parts I have and that these parts are the right ones for my staff, partners, and customers?

✦ How do I ensure that the right part reaches the right person at the right time?

If content management is the process of collecting, managing, and publishing content, e-business is the process of collecting, managing, and publishing parts of your business. Of course, much of that business entails publishing information. So, with no stretch at all, you can see how content management may underlie some e-business. But organizations conduct business by providing for actions as well as for information. Actions, in an electronic world, take the form of computer functionality. Functionality, in turn, takes the form of objects and blocks of computer programming code, which you distribute in exactly the same way as you do run-of-the-mill content.

Cross-Reference In the Introduction to this book, you find more information about e-business. If you're unsure about functionality, check out Chapter 4, "Functionality Is Content, Too!"

The Content Management Industry

At the beginning of the year 2001, the industry of content management is still very new. No standards are universally accepted for what content management systems are or do. Based on my own definition, however, you'd find that the majority of content management systems in use today were created by programmers and Webmasters who were simply trying to keep up with their explosive sites. In the last few years, however, several commercial applications have been released that perform content management in one way or another.

As time goes on, the concepts around content management and the standard offerings of content management systems are sure to coalesce. But for the present, drawing one-to-one comparisons between products is difficult, and clearly stating the advantage of one of these products over a system that you create on your own is often hard. In this book, I don't look at particular content management products. If I did, the information that I'd present would be out of date before I could finish writing it. Rather, I build a set of concepts and the framework of the discipline of content management that you can use to decide what you need in your situation, independent of any particular product. Then you can go to the products that are available with an independent and thorough assessment of what they need to accomplish for you.

Cross-Reference In Chapter 16, "Selecting Hardware and Software," I detail a process that you can use to select the right hardware and software for your particular needs.

Today, you can't find a commercial system that does everything that I present in this book. Most major CMS products offer some capabilities in the three areas of collection, management, and publishing, but they concentrate most of their strengths in only one of these areas. In addition, because no center to the discipline of content management exists, any company offering a CMS product is free to define the term *content management* as it sees fit to best appeal to its customers. As time goes on, this situation is certain to change. What isn't likely to change, however, is your responsibility to understand what you're trying to accomplish with a CMS, and how to organize your content enough to benefit from one. That's what I hope that this book teaches you.

Note

In addition to providing solid concepts and advice to people working with content management "in the trenches," I also hope that this book helps move the content management discipline and industry toward a unified set of concepts and a clear target for the range and depth of functionality that it must provide to truly serve the needs of its clientele.

Do all these qualifiers mean that you should avoid buying a commercial product? That depends. With a little help and customization, the products available today are certainly up to the task of managing a large Web effort. In addition, the CMS products aren't the only ones that are immature and growing. That's the case with the entire digital-information industry. So if you can afford to wait for the entire "online information" phenomenon to settle down, you're sure to have an easier time getting a CMS going. But be willing to wait quite a while. If, like most of the organizations that I know, you've already waited far too long and the weight of your own information is crushing you, you may as well buy a product and start now. Simply understand that you're signing up for more than a single purchase. You are signing up for participation in the ongoing definition of digital publication technologies.

Systems for Managing Content

People use a very wide variety of tools and methods to process information and get it into publications. From using paper note cards to installing massive database applications, organizations find ways of organizing and managing information. My goal in this book isn't to say which of these systems are and aren't true content management. Rather, my goal is to describe the full range of possibilities at your disposal for constructing a system to most effectively handle the range of issues that you may confront.

I begin with the simplest system I've seen that still bears some of the earmarks of a full content management system. I then move up in complexity to describe other systems that embody more and more of the full picture. In Chapter 7, "Introducing the Major Parts of a CMS," I describe what I see as the full CMS.

The nominal Web CMS

As people started creating Web sites, they did so by typing HTML into plain text editors (such as Microsoft Notepad). As time moved on, the need for better tools grew (fueled by enterprising product companies, less technical users, and the need to automate tedious tasks). At first, these new HTML authoring tools did little more than help you remember the arcane syntax of HTML. Later, they began to be true WYSIWYG (what you see is what you get) environments for creating Web pages. Today, some of these tools add just enough management to serve as a nominal Web CMS for organizations with small sites and no additional publications.

Tools such as Macromedia Dreamweaver and Microsoft FrontPage intend to serve as the single tool from which you not only create pages, but also share resources between pages and manage the layout and organization of your site. The main functionality that they offer for these management tasks is as follows:

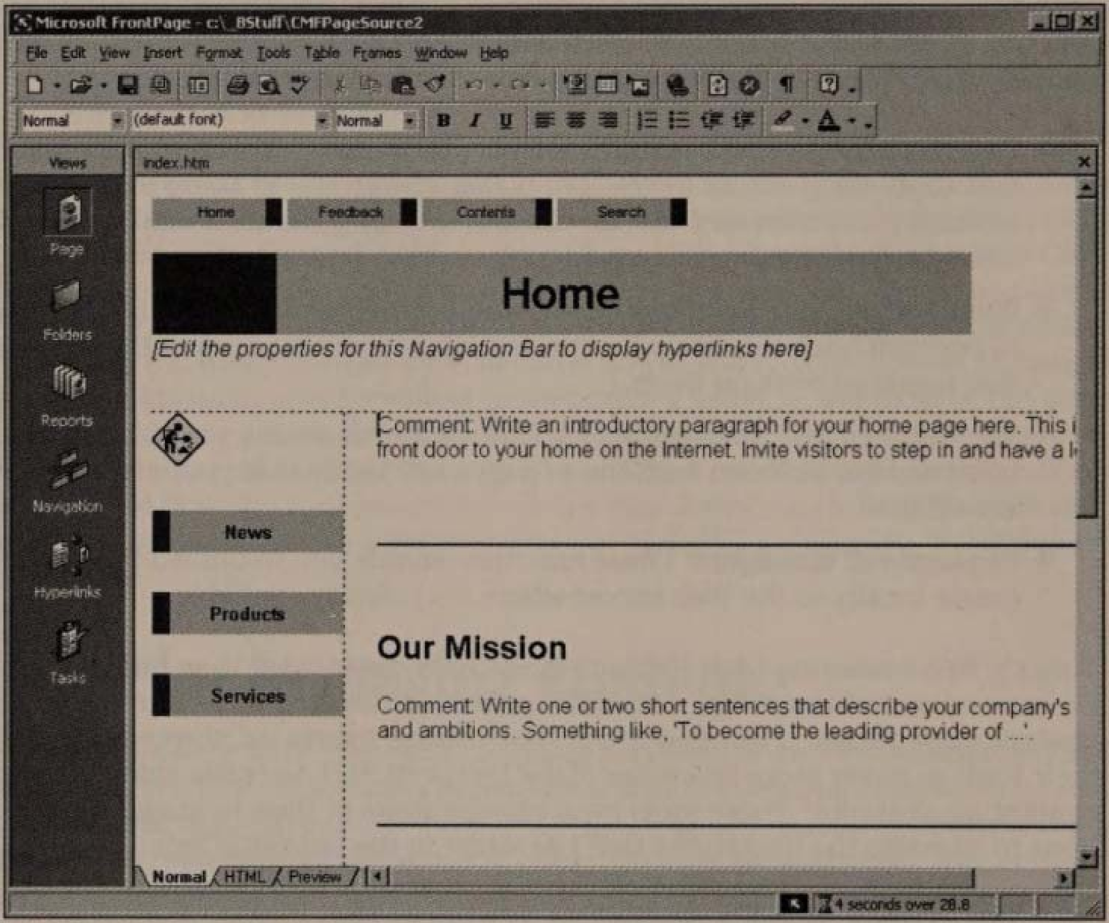
- **Page templates:** These features enable you to create standard page layouts and apply them across pages. Templates enable you to share resources such as images and standard text blocks and to auto-generate some basic navigation based on the pages you add to your site.
- **Basic status:** Such functions enable you to track where a page is located and to perform basic workflow functions such as routing pages to the next person that needs to work on them.
- **Site outlines and link managers:** These features enable you to see the general relationships between sections or pages and verify that your cross-references are valid.
- **Deployment managers:** These functions enable you to upload the site you create locally to the Web server where it's publicly available.

Today's Web-authoring tools (including certainly many more than just Dreamweaver and FrontPage) are making their first halting steps into the realm of content management. I have no doubt that, as time moves on, they will expand their tools to cover more and more of the territory. Still, to really enter the world of content management, these tools must change some of their fundamental assumptions to alleviate the limitations that I describe in the following list:

- **The existing tools are small-scale by design.** They assume a small and loosely organized team producing pages one at a time. A CMS assumes a large, very well organized team producing content that will be moved onto pages in bulk.
- **They're page oriented.** As I discuss elsewhere (see Chapter 23, "Designing Content Components," for more information), in a CMS, pages aren't the things that you manage; content chunks (or components) are. Managing pages is convenient with a single site at a small size. But after your site grows to a larger size — and especially if you must create more than just a single site — you need to move beyond managing pages to managing content components.
- **These tools assume that you're creating one Web site.** (Or, if you're creating multiple Web sites, they assume ones that are completely independent of each other.) They offer no capability to share the same content between pages of the same site, let alone between pages of two different sites. They're not even close to capable of creating any publication that's not in HTML.

Page templates

Microsoft FrontPage, for example, provides a page-templating function that illustrates a number of the basic ideas behind publication templates. Figure 6-1 shows a screenshot of a page that I created in a few clicks by using the site-creation wizards in FrontPage 2000.


Microsoft FrontPage - c:_BStuff\CMFPageSource2
File Edit View Insert Format Tools Table Frames Window Help
Normal
(default font)
Normal
Views
Page
Folders
Reports
Navigation
Hyperlinks
Tasks
index.htm
Home
Feedback
Contents
Search
News
Products
Services
Normal
HTML
Preview
4 seconds over 28.8

Home

[Edit the properties for this Navigation Bar to display hyperlinks here]

Comment: Write an introductory paragraph for your home page here. This i
front door to your home on the Internet. Invite visitors to step in and have a l

Our Mission

Comment: Write one or two short sentences that describe your company's
and ambitions. Something like, 'To become the leading provider of ...'.

Figure 6-1: Microsoft FrontPage 2000 page view

The page has everything on it that you may see on a published page except the actual content. In fact, it's a WYSIWYG template for creating pages for this site. The template includes the following features:

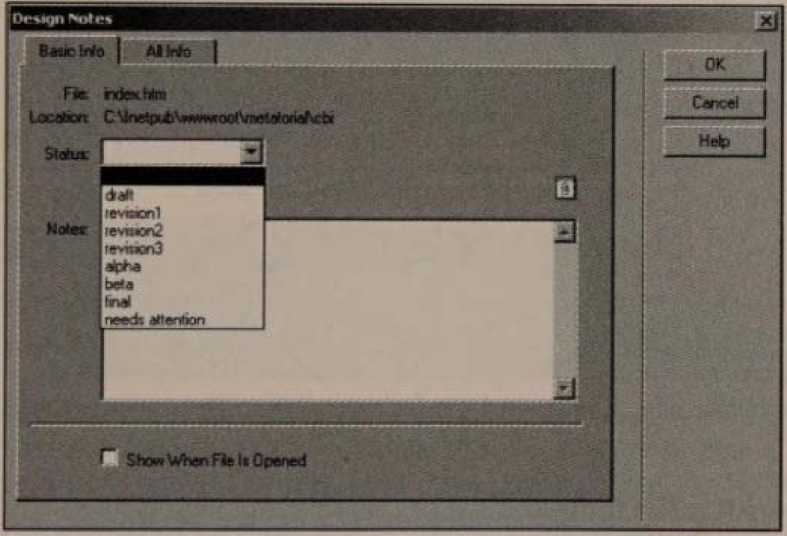
- A banner and global navigation buttons at the top, with names that the application changes as you change the name of the site or its pages.
- A local navigation bar on the left, with button names and order that the application changes as you add or modify pages in this section of the site.
- A footer at the bottom with standard text and links that you want to appear on each page.

Although this system is far from a full publication template system, it shows many of the basics. In particular, it shows how the system can specify page layout, page names, and standard content blocks once and then have the system automatically update and fill in the right stuff in the right place. Most important, if you change the design of the template, the system automatically propagates your change to all affected pages.

Cross-Reference In Chapter 22, "Designing Publications," I describe the qualities of a full publication template system.

Status functions

Macromedia Dreamweaver, for example, provides a function by the name of Design Notes that enables you to specify and review page status and notes (see Figure 6-2).



File: index.htm
Location: C:\Inetpub\wwwroot\metatoria\cbi

☐ Show When File Is Opened

Figure 6-2: The Design Notes dialog box in Macromedia Dreamweaver 3.0

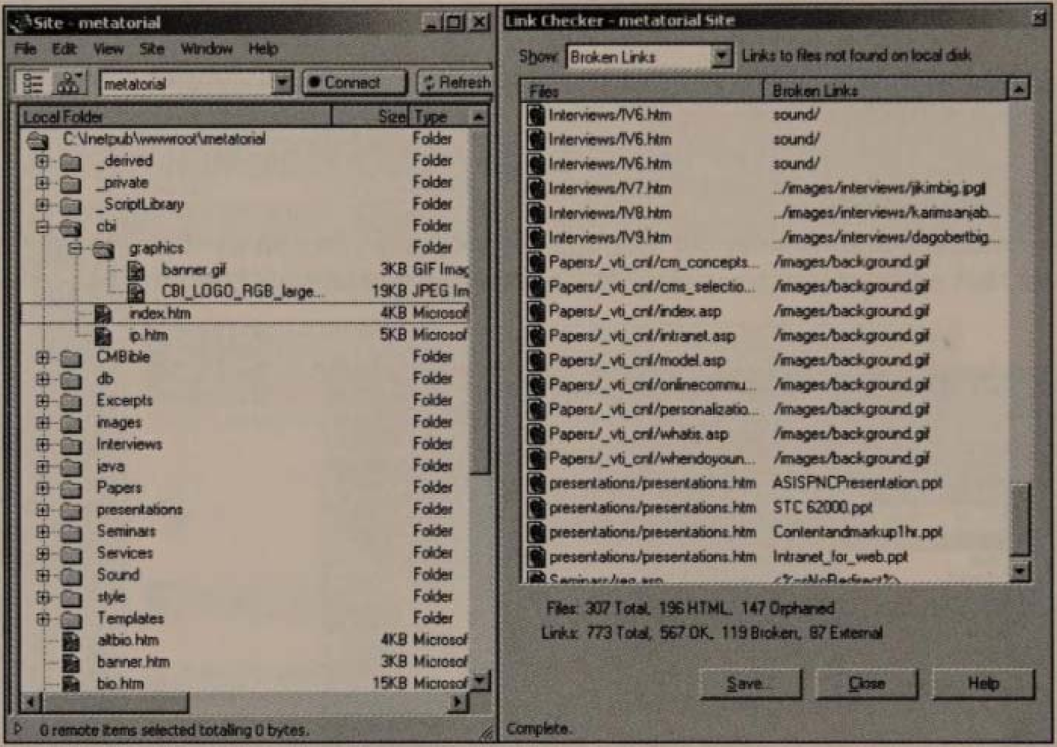
The box enables you to choose a status for the page you're working on and type whatever notes you want. To help you encourage people to update statuses, the application can display this dialog box each time that the page opens.

This functionality is the mere hint of a workflow and routing system, but it shows the basic idea of tagging pages with management information that you don't publish but that authors and administrators use.

Cross-Reference In Chapter 27, "Designing Workflow and Staffing Models," I provide a more detailed overview of project workflow and its attendant challenges. Look there for what a CMS ultimately has to accomplish.

Outlining and organizing functions

Macromedia Dreamweaver provides a site outline and link checker that help you organize the outline and cross-references on your site (see Figure 6-3).


Site - metatorial
File Edit View Site Window Help
metatorial
Connect
Refresh
Link Checker - metatorial Site
Show: Broken Links
Links to files not found on local disk
Save...
Close
Help

Local Folder	Size	Type
C:\Inetpub\wwwroot\metatorial		Folder
_derived		Folder
_private		Folder
_ScriptLibrary		Folder
cbi		Folder
graphics		Folder
banner.gif	3KB	GIF Imag
CBI_LOGO_RGB_large...	19KB	JPEG Im
index.htm	4KB	Microsof
ip.htm	5KB	Microsof
CMBible		Folder
db		Folder
Excerpts		Folder
images		Folder
Interviews		Folder
java		Folder
Papers		Folder
presentations		Folder
Seminars		Folder
Services		Folder
Sound		Folder
style		Folder
Templates		Folder
altbio.htm	4KB	Microsof
banner.htm	3KB	Microsof
bio.htm	15KB	Microsof

0 remote items selected totaling 0 bytes.

Files	Broken Links
Interviews/IV6.htm	sound/
Interviews/IV6.htm	sound/
Interviews/IV6.htm	sound/
Interviews/IV7.htm	../images/interviews/jkimbig.jpg
Interviews/IV8.htm	../images/interviews/karimsanjab...
Interviews/IV3.htm	../images/interviews/dagobertbig...
Papers/_vti_cnf/cm_concepts...	/images/background.gif
Papers/_vti_cnf/cms_selectio...	/images/background.gif
Papers/_vti_cnf/index.asp	/images/background.gif
Papers/_vti_cnf/intranet.asp	/images/background.gif
Papers/_vti_cnf/model.asp	/images/background.gif
Papers/_vti_cnf/onlinecommu...	/images/background.gif
Papers/_vti_cnf/personalizatio...	/images/background.gif
Papers/_vti_cnf/whatis.asp	/images/background.gif
Papers/_vti_cnf/whendoyoun...	/images/background.gif
presentations/presentations.htm	ASISPNCPresentation.ppt
presentations/presentations.htm	STC 62000.ppt
presentations/presentations.htm	Contentandmarkup1hr.ppt
presentations/presentations.htm	Intranet_for_web.ppt

Files: 307 Total, 196 HTML, 147 Orphaned
Links: 773 Total, 567 OK, 119 Broken, 87 External

Complete.

Figure 6-3: The file view of the site outline and the link checker in Dreamweaver 3.0

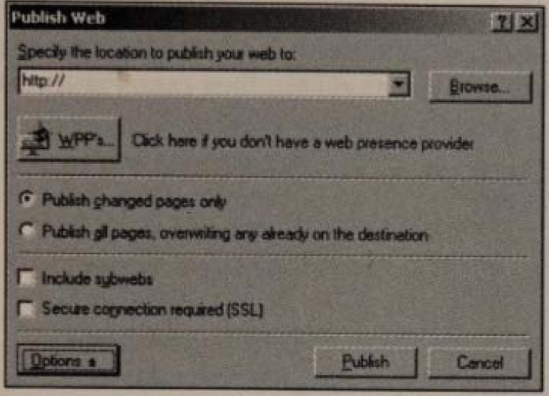
A full CMS provides a much more thorough set of tools for managing hierarchies and cross-references (as well as indexes and sequences). The capability to see the directories where your files are stored and to identify broken links is, however, a start.

Cross-Reference

In Chapter 26, "Designing Content Access Structures," you can find out more about access schemes such as hierarchies and indexes. There is a lot more to know than is shown in Dreamweaver.

Deployment tools

Microsoft FrontPage offers a function for deploying your site to a final Web server, where your site becomes "live" on the Web and viewable by the world. Figure 6-4 shows a screenshot of the Publish Web deployment tool in FrontPage.



Publish Web

Specify the location to publish your web to:

http://

Click here if you don't have a web presence provider

● Publish changed pages only

○ Publish all pages, overwriting any already on the destination

☐ Include subwebs

☐ Secure connection required (SSL)

Figure 6-4: The Publish Web dialog box in FrontPage 2000

This particular function enables you to communicate via FTP (File Transfer Protocol) to a server on the Internet and deploy all your pages or only those that have changed since the last time you deployed them. This method of file transfer is the most basic form of deployment. I illustrate more advanced forms of deployment in the sections that follow.

The dynamic Web site

A dynamic Web site, sometimes known as a database-driven site, is a system for producing Web pages "on the fly" as users request them.

A data source (a relational database or possibly an XML structure) on the Web server receives a query in response to a user clicking a link. The link activates a template page. The template page contains regular HTML as well as programming scripts, objects, and other programs that interpret the request, connect to the data source, retrieve the appropriate content, and do whatever processing is necessary to form an HTML page. After the template creates the appropriate HTML page, the Web server sends it back to the user's browser (see Figure 6-5).

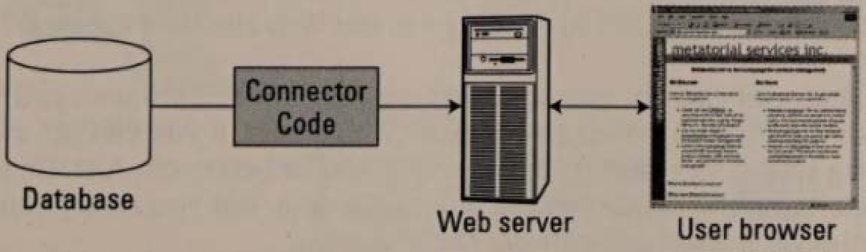


Figure 6-5: A dynamic Web site creates pages programmatically, sending them back to the inquiring browser.

In a purely dynamic site, no HTML files exist — only the capability to build them whenever someone wants them. Contrast this fluidity with a static site, where all the pages are pre-built and stored on a Web server as HTML files (see Figure 6-6).

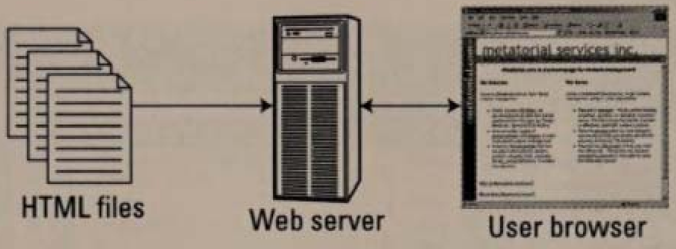


Figure 6-6: A static Web site delivers pages that are already built.

Given this definition of dynamic sites, you can easily see why people often confuse them with content management systems. For one thing, you can say that content management systems do essentially the same thing. They, too, maintain databases or XML structures, retrieve appropriate content, and return built pages. On the other hand, you have compelling reasons to distinguish between the two. Why? Because you can have a dynamic site that really isn't doing content management at all. In addition, a content management system can just as easily build a static site.

Suppose, for example, that you maintain a large, dynamic Web site that uses advanced scripts to put a user interface on your organization's financial system. The system responds to user requests and dishes out just the right HTML page in response. You'd be hard pressed to call this setup a content management system — it's really just a Web-based application.

You can, on the other hand, have a CMS that produces a static site. Suppose that I set up a very complex CMS that contains millions of components, offers a sophisticated workflow, and produces 100 distinct publications. One of those publications is a Web site. After I hit the right button, out flows a static Web site of one million HTML files. I put those files on a Web server and I'm done. If I want to change the site, I hit the button again, and out flows a new, static Web site that replaces the first one. In this case, I have a robust CMS producing a static Web site (see Figure 6-7).

So, dynamic Web sites and content management systems aren't the same, and you don't need to produce a dynamic Web site from a CMS. In fact, if you can get away with only producing static Web sites from your CMS you're better off. A static site is faster and much less prone to crashing than a dynamic one. But you rarely can get away with static sites or at least not entirely static ones.

You need a dynamic site if you don't know beforehand what may appear on a page. If you must assess user input or some other factors to figure out what belongs on a Web page, you need a dynamic system. For any realistic degree of personalization, for interaction with data systems (transactions, catalogs, and so on), or for "live" updates (such as changing news stories or stock quotes), you need the processing power on the Web server that a dynamic system gives you.

Although dynamic sites aren't content management systems, they illustrate a number of the qualities of a CMS, as follows:

- **The template pages** on a dynamic site are very similar in approach to the templates in a CMS. Unlike the templates system in some of today's Web-authoring tools, which may be isolated from you and offer a limited set of features, dynamic sites use generalized Web programming technologies such as Java Server Pages (JSP) and Active Server Pages (ASP). These programming languages are virtually unlimited in scope and can create any sort of page layout and logic that you may want. The templating system in many commercial CMS products is no more than an enhanced version of the kinds of template files that dynamic sites use.
- **The types of data sources** on a dynamic site are similar to and sometimes even the same as those you use in a CMS. In either a dynamic site or a CMS, you're likely to use the same database or XML products, use the same programming techniques, and write the same sorts of content selection, layout, and navigation-building code. A good CMS, however, provides you with enhancements that make these tasks much easier. CMS data sources also tend to store more management information (metadata) than the data sources for dynamic sites.

What's typically most often missing from a dynamic site system is the capability to create more than just a Web site and the collection system that a CMS includes. Of course, nothing is stopping the enterprising programmer from extending Web-site code to perform these functions, but then she's not creating a dynamic Web site; she's creating a CMS!

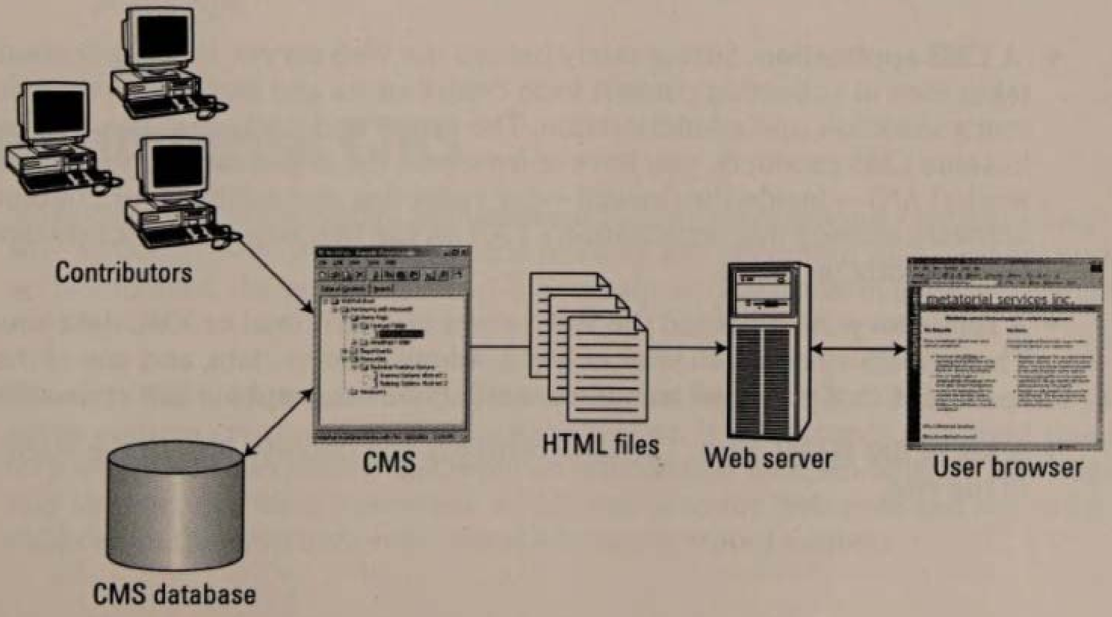


Figure 6-7: A CMS can, and often does, create static Web sites.

The full Web CMS

Sites don't need to be 100 percent static or 100 percent dynamic. In fact, the vast majority of large Web sites are a little of both. Parts of your site can consist of HTML files and other parts you can dish up dynamically out of a database. In addition, a variety of databases may provide different parts of your site (see Figure 6-8).

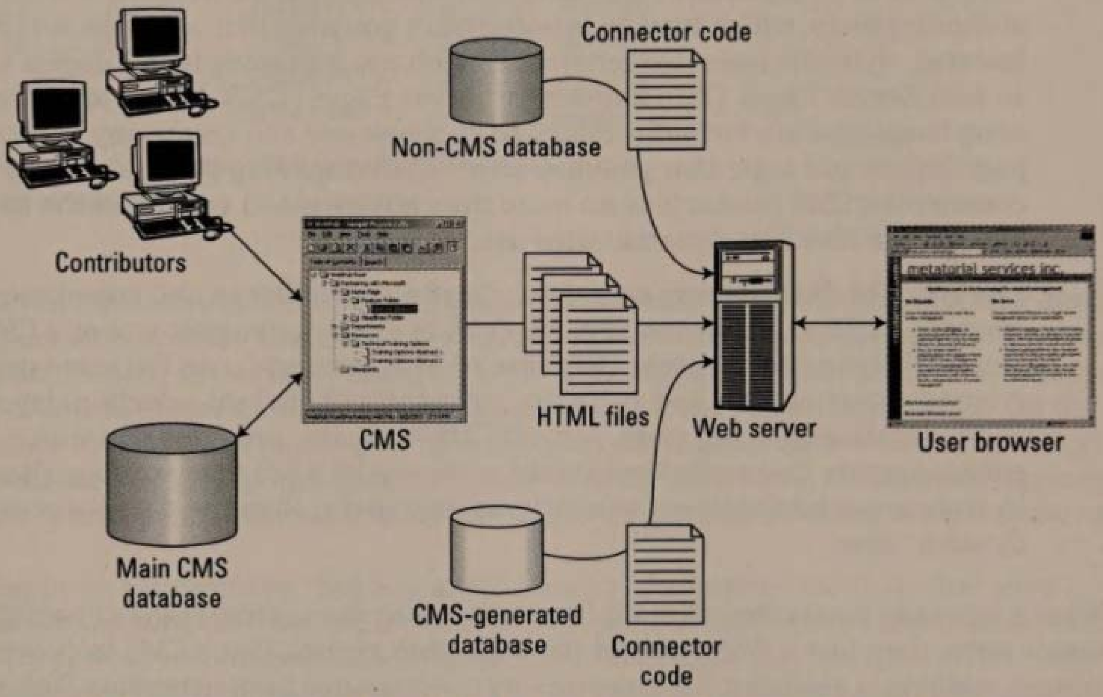

Connector code
Non-CMS database
Contributors
CMS
HTML files
Web server
metatorial services inc.
User browser
Main CMS database
CMS-generated database
Connector code

Figure 6-8: CMS can stand behind all the static and dynamic parts of your site.

In a full Web CMS, you can have any or all of the following:

- **A CMS application:** Sitting safely behind the Web server, this application takes care of collecting content from contributors and managing your content's workflow and administration. The actual architecture varies by product. In some CMS products, you have software on the organization's local area network (LAN) — inside the firewall — for collecting and testing your content, and software outside the organization's LAN on the live Web server for dynamically serving up pages.
- **A repository:** Also behind the Web server is a relational or XML data source. The repository holds all your content, administrative data, and any of the resources that you need to build the site (such as graphics and style sheets).
- **A set of flat HTML files:** The CMS manages and deploys files to the static part of the site.

- **A live data source (the CMS-generated database):** This one is located on the Web Server for the dynamic parts of the site. The CMS can deploy data and content from its repository to the CMS-generated database. In this way, even dynamic content can be managed behind the firewall and kept off the server if it is not ready to be seen publicly. In addition, the template pages that access the CMS-generated database can be pages that are created by the CMS.
- **Other data sources:** You can connect other sources of data to the Web site that you don't connect to the CMS. A transaction database for conducting sales on the site, for example, you may connect to the Web site but not to the CMS. The other data sources can run completely independently of the CMS (or the template pages) that accesses the sources.
- **Templates:** A set of publication templates moves data from any source to the state it needs to be in for the site. In some cases the data needs to be put into finished static HTML pages. In other cases, it needs to be put into databases that will live on the Web server.

As you can see, you can quickly get a Web site that is pretty complex. It is complex in two ways. First, it has a lot of software and hardware parts that must be coordinated. Second, it has complex information structures that must be stored and correctly delivered. You can manage the complexity of the information with a CMS. If you consider the various chunks of code in the template pages as chunks of functionality (which they are), you can treat them as just another kind of content to collect, manage, and publish onto the right pages.

Tip

In general, your best bet is to drive as much of your site as is feasible toward static pages. Even if content changes once a day, you're probably better off producing it as HTML files and posting the ones that change once a day to your server. Flat pages are tremendously faster and more reliable than dynamic pages. You need compelling reasons to dynamically generate pages—for example, the content is changing minute by minute or you need to personalize the content for each user or user type.

The Enterprise CMS

Most discussions of content management today center around creating a large Web site. Although large Web sites are the primary use of content management systems at the moment, the potential for a CMS to help an organization goes far beyond the Web.

Clearly, a CMS can do a lot more than produce a Web site. It can encompass your entire content creation and organization system. It can provide a content repository where you can review and work on information independently of any page it may land on, and most important, a CMS can produce Web sites and any other publication that you may care to make from the stored content.

If you look at your content collection, management, and publishing needs from the perspective of the whole organization, you quickly see that the potential for content management goes way beyond the Web. In my experience, this insight is dawning only slowly in organizations that are hard-pressed to even get their sites organized, let alone revamp the information processing in their entire enterprise.

Many organizations are accepting (at least for the present) that their print publications are quite separate from their Web publications (not that most would even call a Web site a publication) in the following ways:

- They're willing to spend the extra time and money duplicating effort between a print team and a Web team.
- They're willing to put up with the lack of synchronization between the two sources of information that they produce.
- They're willing to let one publication wait for content until the other goes to press. At first, the Web site was always what waited. More and more, however, the print publication must now wait for Web content.

I think this reticence to think globally is often driven by the entrenchment of print-based teams in their traditional tools and methodologies. For some print team members, the CMS attitude (away from a specific publication and toward creating good content that can serve multiple purposes) is a godsend that enables them to reach new heights. For others, it's no more than needing to learn a lot of new programs and unlearn a lot of old habits.

Not surprisingly, I've seen much more desire to think globally from those who must share content between Web sites or between the Web and other digital devices (wireless phones, for example). These teams already embrace the new authoring tools, content formats, database technologies, and templating models that dynamic Web sites have taught them. Considering going the final steps to a CMS isn't such a big stretch.

Interestingly, the divide between publications on handheld digital devices and the Web is much wider than the divide between print publications and the Web. Handheld appliances can show only a couple of sentences of text at a time and only very, very small pictures. They require a major rethinking of how you deliver anything you're normally used to seeing either in print or on the Web.

In this book, I talk most about enterprise content management. Although the Web still figures most prominently, nothing's exclusive about the Web's demand for well-managed content. If your sole concern is the Web, it likely won't stay that way for long.

Cross-Reference In Chapter 7, "Introducing the Major Parts of a CMS," I fully develop the definition of an enterprise CMS.

Summary

Content management systems have a simple purpose — to make it so that you can get content with the least hassle, so that you know what content you have, and so that you can get it out to a variety of places automatically. The details are less simple, as the following list illustrates:

- There is now only the beginning of a content management industry with only immature products currently available.
- Most content management systems in place today cover only the Web.
- Web-authoring tools are great for small sites, but they lack the functionality or scalability to serve as real systems.
- Today's most-sophisticated dynamically generated Web sites go a long way toward content management but are still much too narrowly focused to do the whole job.

While the technology matures, you're best off spending your time and energy taking what steps you can to move toward a content management approach. This strategy means first figuring out in detail what you want to happen and then seeing how far you can get with today's tools.

In the next chapter, I go in detail into the structure of an enterprise content management system.

✦ ✦ ✦

CHAPTER 7

Introducing the Major Parts of a CMS

◆ ◆ ◆ ◆

In This Chapter

A high-level view of CMS features

Collecting content

Managing content

Publishing content using templates

◆ ◆ ◆ ◆

A lawyer, a judge, and a psychologist were discussing a murder case. "It's about the law," said the lawyer. "Either she broke the law or she didn't." "You're wrong," said the judge. "This case is about justice. The moral order must be upheld." The psychologist shook his head and said, "This case is about love; she loved him, but he didn't love her."

You have a number of valid ways to look at crime and, in the same way, a number of ways to look at a CMS, all equally valid, as the following list describes:

- *A CMS is a hardware and software system* that consists of programs running on particular computers.
- *A CMS is a set of content types* and the processes that you use to manage them, from creation to distribution to final destruction.
- *A CMS is a vehicle* that enables an organization to connect its information sources and computer functionality to its constituents.

In this chapter, I present a perspective that contains bits of all the others: A CMS is a system that collects, manages, and publishes information and functionality. I present a top-level view of content management that's partly hardware and software, partly process, and partly an organizational vehicle.

Note By the way, the illustrations in this section are based on the CMS Possibilities poster my colleagues and I created at Chase Bobko Inc. For more information, go to `http://www.metatorial.com`.

A CMS Overview

A content management system (CMS) is responsible for the collection, management, and publishing of chunks of information known as *content components*. Schematically, you can envision the process as shown in Figure 7-1.

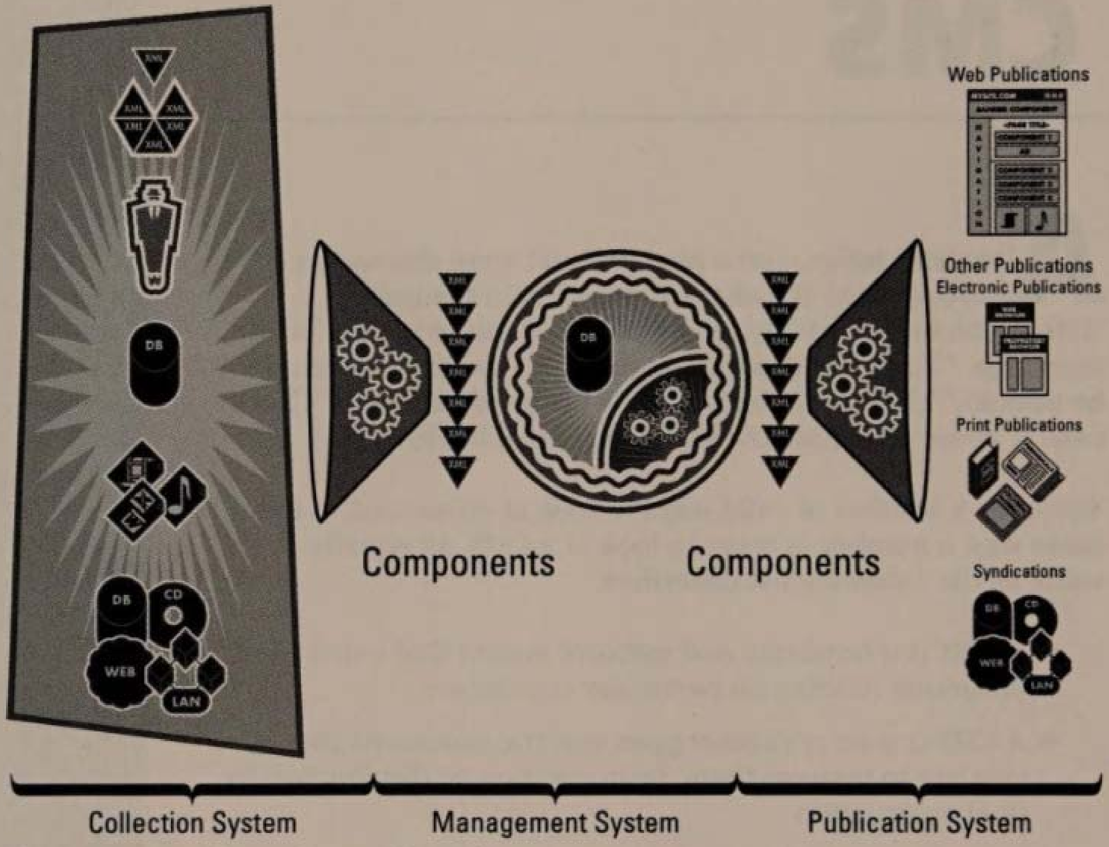

XML
DB
CD
WEB
LAN
Components
Components
Web Publications
Other Publications
Electronic Publications
Print Publications
Syndications
Collection System
Management System
Publication System

Figure 7-1: A schematic overview of a content management system

Flowing from left to right, the illustration shows how raw information runs through a collection system and turns into content components. A management system, which is a sort of database, stores these components. The publication system draws components out of the management system and turns them into publications. This illustration is the highest-level view of the content management model that I use throughout this book.

You need to notice that, although logically separate, the three parts of the system can involve large physical overlaps, as the following list describes:

- **The management system can serve as part of the collection system.** You very often deposit content in the content repository of the management system before it is fully processed (making it part of the collection system).

- **The management system can serve as part of the publication system.** The repository, for example, often sits on the Web site it creates, and you can't easily distinguish it from the system that publishes the site.
- **The publication system can serve as part of the collection system.** In a Web-forms collection system, the author types content into a Web form. The content that the author types is stored in the repository. Behind the scenes, the publication system creates the Web forms that the collection system uses.

The following section immediately moves down another layer and puts a bit of sinew on the CMS skeleton that I'm describing.

The Collection System

A CMS collection system is responsible for all the processes that happen before a piece of content is ready for publication. It turns raw information into a well-organized set of content components. Figure 7-2 presents an overview of the collection process, showing authoring, acquiring, converting, and aggregating as well as the collection services that a CMS provides.

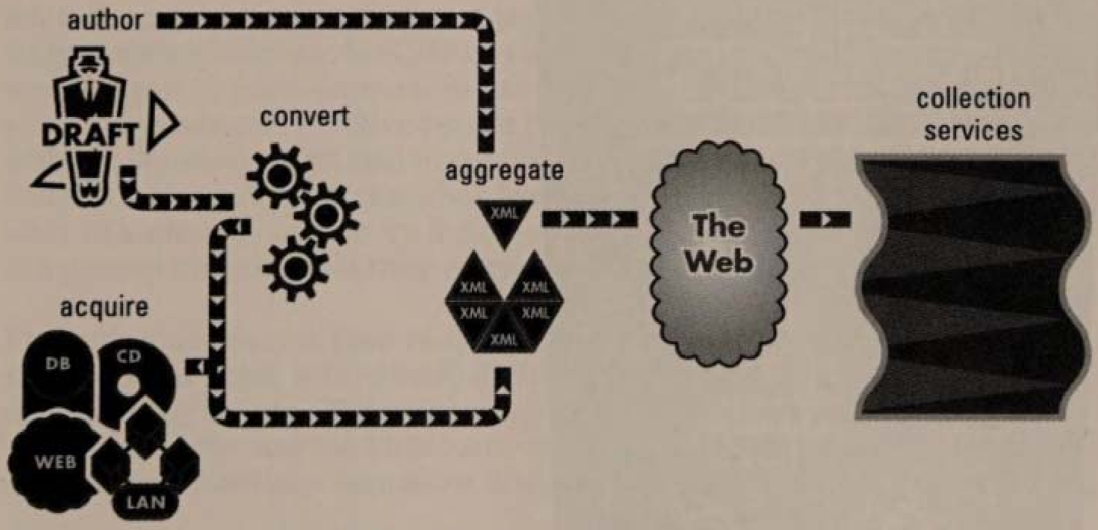


Figure 7-2: An overview of the collection process

These processes include the following:

- **Authoring:** Here someone creates the content from scratch.
- **Acquisition:** Here you gather the content from some existing source.

- **Conversion:** Here is where you strip unnecessary information from the content and change its markup language.
- **Aggregation:** You edit the content here, divide it into components, and augment it to fit within your desired metadata system.
- **Collection services:** These services are CMS programs and functions that aid the collection process. For example, the collection services produce the Web forms into which you enter the content for components.

Note

For clarity, Figure 7-2 implies that these processes happen serially and in a particular order. In reality, they may not happen in the order that the figure implies. In addition, the entire process is shown apart from the management system for simplicity only. In fact, much of the collection process can and usually does happen after content enters the repository.

Authoring

Authoring refers to the process of creating content from scratch (see Figure 7-3).

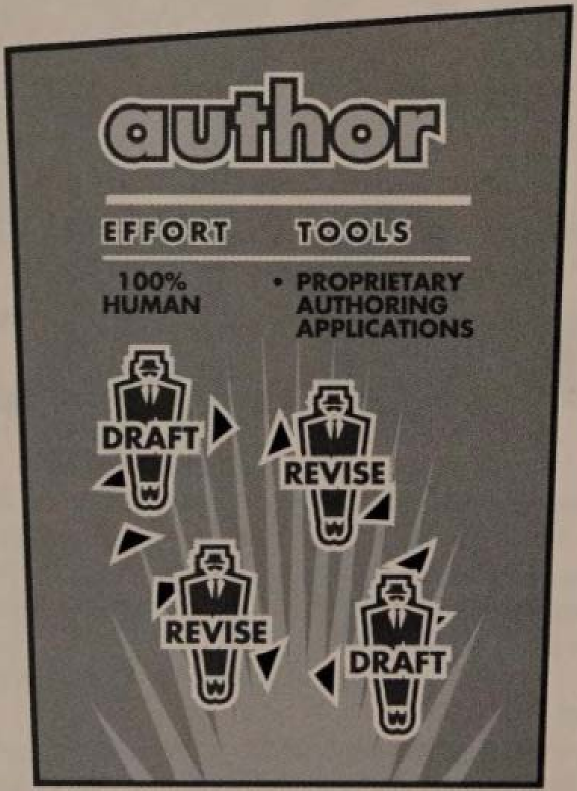


Figure 7-3: The authoring process is an essentially human process of creation and revision.

I define an *author* as someone you specifically commission to create content for you. If the content creator isn't under your jurisdiction or creates the content for any purpose other than for your CMS, I define her content as acquired, not authored.

A CMS can help the author work efficiently and effectively by doing the following:

- **Providing an authoring environment** (either a full application or extensions to the author's native environment).
- **Providing a clear purpose and audience** for the author's efforts. In a CMS, you direct authors to create particular content components that already are defined in terms of their basic purpose and audiences.
- **Providing aids** for including standard information. The CMS can easily fill in the create-date and author's name, for example, to save the author effort.
- **Providing templates** that break down the content the author creates into its constituent elements. You may provide authors with a Microsoft Word template (a DOT file), for example, that already includes places to type a title, summary, and body for the component's author to create.
- **Providing workflow, status, and version control** for content that's in process.

Regardless of how many tools and processes the CMS provides, authoring is essentially a manual process. No CMS can tell an author what to say or, more important, how to say it to best communicate an idea or goal. Because authoring is manual, it's slow and expensive. Creative people can create only so quickly. Creative people with the knowledge and skill to create good content are a limited resource and thus come at a high price. On the other hand, you can count on the quality and applicability of authored content. By definition, authors are under your direction, so you can control the quality of their output.

The authoring process itself is one of creation and revision, where the author (possibly in conjunction with others) drafts and revises her work until she is satisfied that it's ready for use. The author may keep her work outside the CMS until it's complete, or she may load her early drafts into the CMS to take advantage of its management functions (workflow, status tracking, version control, and so on).

Acquiring

Acquiring is the process of gathering information that wasn't originally created for your CMS (see Figure 7-4). This process might be partly manual or fully automated.

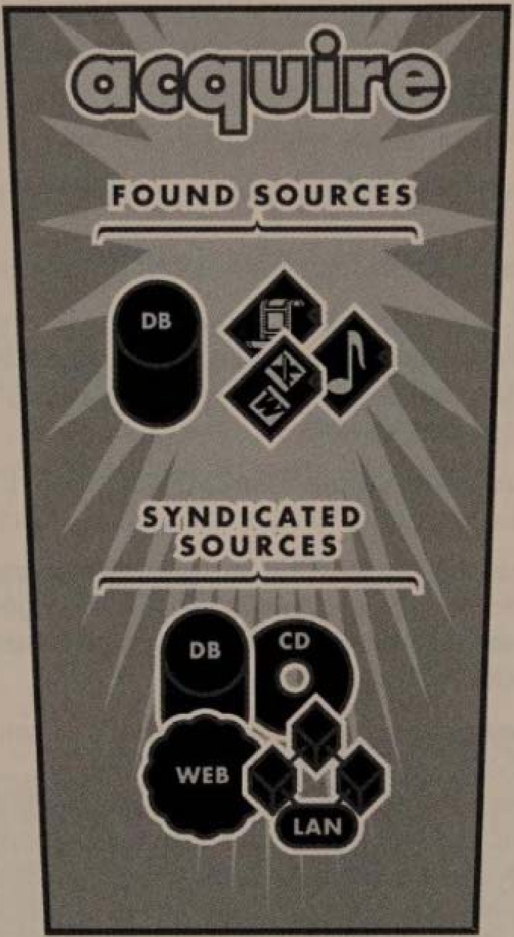


Figure 7-4: Acquiring is the process of gathering information that was created for some other purpose.

Unless you're extremely lucky (or have lax quality standards), you're going to have some work to do to get this information ready for the CMS. Acquired information comes from the following types of sources:

- **Syndications:** Syndications are sources that are designed for reuse. *Designed for reuse* means two things: First, it means that the information is delivered in a generally useful binary format (XML, for example). Second, it means that the information is already segmented and has metadata attached. Syndicated news, for example, generally comes in story segments that are tagged with at least a headline, byline, date, and story. Today a small but quickly growing commercial industry sells reusable information. Commercial syndicators can package and deliver a wide range of high-quality information (for the right price, of course). Not all syndications, however, are commercial. You can use your CMS to package and distribute your components in a reusable form to others in your organization, for example, and if you're lucky, others can do the same for you.

- **Source files:** These files are any sort of preexisting information that a computer stores. I include in this definition nonelectronic sources such as paper photographs, analog video, and printed text that, after you digitize them, end up in files as well. Source files generally aren't designed for reuse. They're often not in an easily reused binary format, are unsegmented, and have no metadata attachments. Except for databases, which can be (but certainly aren't always) well structured, source files require work on your part to transform into a usable form.

Authored information is generally low volume but high quality. Acquired information is generally high volume but low quality. You may have photographs of all your products available as JPEG images, for example, but they may need hand processing to crop them and adjust their contrast.

You may get lucky and find sources that are high quality (meaning that you have little to do to process them), but don't count on it. Most information people create is for a limited use. If you bring it into a system for unlimited use, you can reasonably expect to do additional work to make it usable.

Receiving acquired information may prove as simple as saving attachments sent to you in e-mail or as complex as setting up a sophisticated server that receives information feeds at regular intervals and automatically processes them. As often as not, it's a simple process of receiving files from a network location or on CD-ROM.

Tip

Make sure that, after you receive acquired content, you move it into your own storage area. Doing so prevents the contributor from inadvertently deleting or modifying the files after you start processing them.

Converting

If the information that you create or acquire isn't in the format or structure that your system requires, you must convert it to match the accepted standards of the content system (see Figure 7-5).

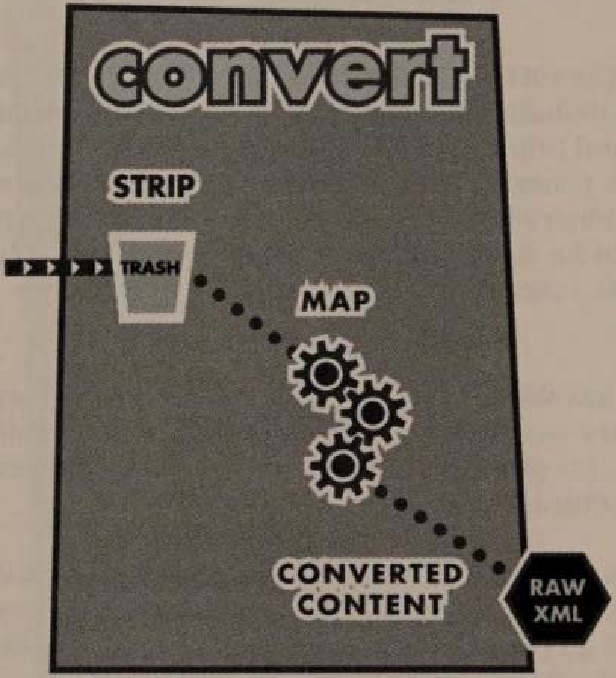


Figure 7-5: Conversion is the process of digitizing and then changing the binary format, and/or remapping the structure of information.

The conversion process consists of the following three logical steps (these steps usually mix together in an actual conversion process):

- **Stripping:** This step is where you remove and discard unneeded surrounding information such as page headers and footers, unnecessary content, and unwanted navigation.
- **Format mapping:** In this step, you change the information's binary format to a standard one that the CMS supports. In addition, you separate its rendering format from its structure.

✦ **Structure mapping:** This step is where you make the information's structure explicit or change it as necessary.

Information that you acquire from nonelectronic sources, such as printed text and analog sound, you must first digitize—that is, you must give it a binary format before you can change that format.

Cross-Reference

I define *format* as the binary file structure and rendering characteristics of a piece of content. In Chapter 2, "Content Has Format," you find more information about this definition. I define *structure* as the set of named relationships within, between, and beyond individual pieces of content. See Chapter 3, "Content Has Structure," for more information about structure.

Obviously, you need to convert information only if what you receive isn't what you need. Thus conversion usually ties in to acquired information, where you have no control over the format and structure of this type of information. But just because you receive HTML (for example) and you need HTML doesn't mean that you don't need to convert that information. If the target HTML (what you want) has a different structure or format from the source (what you have), you must still convert its structure.

The result of a conversion process is information that conforms as well as possible to the standards you develop for format and structural tagging. I sometimes use XML as shorthand for these qualities. If you convert your information to XML, you're (most probably) standardizing its tagging. On the other hand, you don't need to convert information to XML to get the same effect. Any format that can represent structure works.

Cross-Reference

In Chapter 30, "Processing Content," I discuss in depth the mechanics of conversion.

Aggregating

In addition to authoring or acquiring it—and converting it, if necessary—you must aggregate your information (see Figure 7-6).

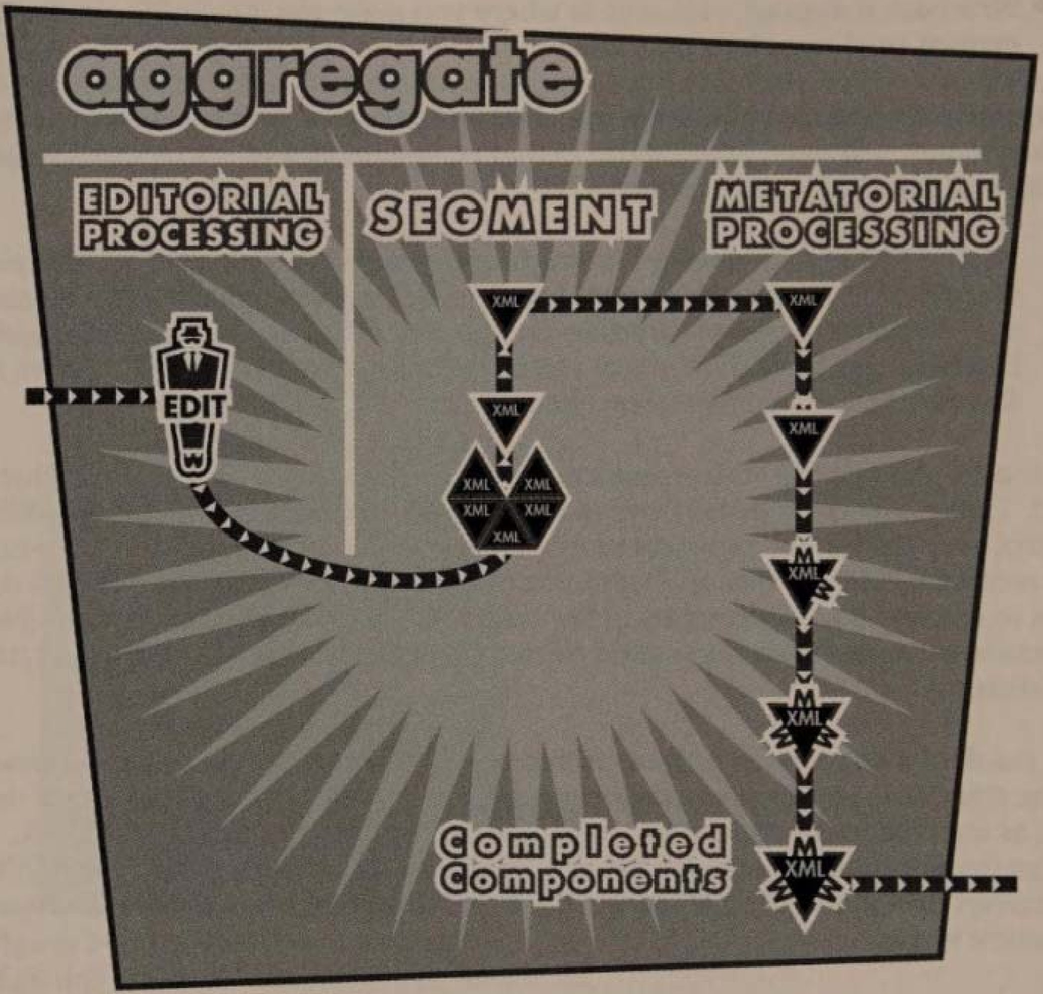

aggregate
EDITORIAL PROCESSING
SEGMENT
METATORIAL PROCESSING
EDIT
XML
M
Completed Components

Figure 7-6: Aggregation refers to bringing information into a CMS through editorial processing, segmentation, and metatorial processing.

I define *aggregation* as the process of bringing disparate information sources into one overall structure through:

- **Editorial processing:** This process performs the familiar functions of styling, consistency, and usage.
- **Segmentation processing:** This process is where you break the information into chunks that I call *content components*.
- **Metatorial processing:** This system performs the less familiar function of fitting new content into the prescribed metadata system. The metadata that you apply to the content enables the system to effectively store and retrieve it. This system consists of the rules that you create for how to supply metadata values for each new piece of content that you bring into the system.

Note Aggregation is often a part of the conversion process. As information goes through stripping, formatting, and structure mapping, you can accomplish much of the segmentation, editorial processing, and metatorial processing. I break the two apart because some information may not need converting, but you must still aggregate it all.

Metators, who are the editors of metadata, tag the segmented information with metadata to enable you to automatically store, find, and later combine that information into publications. Just as editors edit, metators "medit." A metator uses a Metatorial guide that details accepted metadata values and how to tag each kind of content.

Editors, segmentors, and metators aren't necessarily different people. In fact, in a small system, the same person (who may have other tasks as well) probably performs these three tasks. But as the size of the system grows, these three tasks present the natural way to divide the workload.

Additionally, you don't usually need to apply the same amount of effort to all information. Content that someone with good editorial and metatorial skills authors directly into the CMS as a component, for example, may need no work because the author aggregates as he authors it. On the other hand, information that you acquire from a source with little metadata, a divergent editorial framework, and no correct segmentation may need a lot of work to aggregate.

Editorial processing

All professional publishing groups use an editorial framework to guide their work. This framework consists of the following types of rules:

- ✦ **Correctness rules:** These rules ensure that you prepare content in accordance with generally accepted standards. Rules on punctuation, word usage, and grammar are all within this realm.
- ✦ **Communication rules:** These rules ensure that the content projects a specific image and targets a specific audience. Rules concerning the voice of the content (active, passive, first person, third person, and so on) as well as other stylistic rules are within this realm, as are rules concerning the correct way to communicate with the intended audience (such as the right vocabulary to use).
- ✦ **Consistency rules:** These rules ensure that you apply all the other rules evenly across the entire content base and that, after you define it, you always use a term the same way.

Without such a framework, publications can lack unity and seem disorganized and unprofessional. For static publications (such as books and magazines), which people manually produce and handcraft, the editorial framework is all that you need to ensure unity.

I don't suppose that I need to say too much more about this subject. Much is written elsewhere, and I'm a pretty lousy editor myself. I don't catch errors very easily, and unless I'm under the iron thumb of a DTD (a set of rules that you use to enforce consistency in documents that you code in XML), I tend to disobey the rules as often as I follow them.

What I can say about editing is that it's a very human effort. Tone, style, usage, and the like are certain to prove the last of the pillars of language to dissolve before the advancing tide of digitization.

Note Although editing is a human effort, you can still create good tools to help the editor in her very labor-intensive task. To work, though, the tools need to be very local in scope, and under the direct and immediate control of the editor. They're on the quick-return side of the automation ROI spectrum. You can learn more about these tools in Chapter 30, "Processing Content," in the section "Focusing on the long-term benefits."

One of the few additions that a consideration of content management offers to the art of the editor is the idea of loosening the constraints on the content so that you can use it in the widest variety of contexts. In the content management realm, the editor must constantly ask the following questions:

- What if I remove this piece of content from the text and media that I now see around it? I can't assume, for example, that what I see as I write this paragraph precedes or follows the words that you're reading now. Maybe you're receiving them in an e-mail message quite apart from the surroundings that I see. Maybe they're stripped of all imagery for display on a tiny screen. Maybe they're combined with sections widely dispersed in this work to form a custom white paper.
- What constraints does the content put on the frame that may surround it? Are the colors in the graphics, for example, likely to clash with the branding and advertisements that may surround them on a Web page? Can the line lengths cause a horizontal scroll bar to appear on-screen?
- Do links make sense in all media? Do the terms "earlier" and "later" or "above" and "below," for example, make sense in every publication in which the content is to appear? How can you refer to external information in a way that makes sense regardless of the publication platform?

All in all, content that you compose for componentization and multiple use must be far more open and forgiving than single-use continuous stream content. Finding just the right language devices that work well enough in every medium isn't easy. Of course, you can always fall back to writing multiple versions for multiple outlets, but the return isn't high on such a labor-intensive effort. In the long run, helping now to create the language devices that people can take as second nature in the future for representing content in a high-value but output-neutral way is better for everyone.

Segmentation

Segmentation is the process of dividing content into convenient and useful chunks. In other words, segmentation is the process of putting content in components. Obviously, before you begin segmenting content into components, you must know what kinds of components you're trying to create.

Cross-Reference

In Chapter 23, "Designing Content Components," I discuss the nature and composition of components. Here, I want to focus on the processes that creating them involves.

The process that you use to create components depends on the source of the components, as the following list indicates:

- Components that you author one at a time you can create as you author them.
- Components that you acquire you can segment after or as part of the conversion process.

How you segment content depends on the nature of its source. In all cases, however, what's most important is how well marked each component is in its source. Components are generally marked in source files in the following ways:

- **By a file boundary:** You may receive 50 files, for example, each containing one office location fact sheet. If each file is to become one Office Fact Sheet component, the segmentation is already complete. If there are two fact sheets per file you receive, then you have some segmentation work to do.
- **By a database record boundary:** Suppose, for example, that you receive a database full of employee data, and you want to turn it into a set of Employee Profile components. You process the database records one at a time. Each time that you load a new employee record from the database, you create a new component in your system.
- **By explicit markup:** Suppose, for example, that you're receiving international restaurant menus from a good XML syndication source. Each menu yields a Menu component. You're unlikely to receive one file per menu. Rather, you're apt to receive one file with many menus. Each menu is *explicitly* marked (by, say, a `<Menu>` tag) to show where it begins.

- **By implicit markup:** Suppose, for example, that you receive a file that contains a printed product catalog from which you want to create a set of Product components. Each new product is marked in the catalog by a hard page break. As you process through the catalog, you look for this *implicit* markup to begin a new component.
- **Not marked:** Suppose that the print-product catalog that you receive marks each new product with the font Times New Roman, 12-point, Bold. Times New Roman, 12-point, Bold, however, is also used in other ways throughout the catalog and isn't a reliable marker.

Cross-Reference

As I discuss in Chapter 30, "Processing Content," in the section "No correspondence," if you have a "Not Marked" situation, you need a trained eye and not a computer to ensure that you can find the component boundaries in the file.

Most segmentation of acquired content occurs as part of the conversion process. I categorize segmentation, however, as part of the aggregation process. Although mechanically it may consist of stripping and mapping, conceptually you can much better describe it as a process of bringing outside content into your CMS.

Metatorial processing

Metatorial processing is designed to ensure that the metadata you apply to incoming content is complete and consistent. But how does this process actually happen? To explore further, this section follows a fictitious metator, Argun, through his day.

At 8 a.m., Argun arrives at work. After fixing his tea, he sits down to his e-mail. He's set up a rule in his e-mail system that routes any e-mail message displaying the prefix "CMS-Related" into a particular folder. (The CMS administrators previously set up the system so that any e-mail message it autogenerates carries the prefix "CMS-Related."). His CMS-Related folder shows about 40 e-mail messages since the last time that he checked. Twelve of these messages are asking him questions about how to tag something. The rest are notifications of newly submitted content that he must review by the end of the day.

By 9:30, Argun dispenses with the questions by pasting the same 10 or 12 sections from the metatorial guide into his e-mail responses. In one case (the only one he enjoyed dealing with), the contributor found a nice hole in the rules. Argun sets himself a task to update the guide to cover the new situation. In each e-mail message that requests a review is a link Argun can click to go directly into the CMS Web-based administration system and view the component to review. He opens the first new component and begins to review it. The message is as he expects, given the author. She's entered the required management metadata and made some pretty good guesses on the access metadata that wasn't required. She knows her audiences well and always nails the targeting metadata. With a few short corrections, Argun sends this component on to its next phase, editorial review.

Within the next hour, he dispenses with the majority of the new content. He's cheated, however, by opening the ones from the "good" authors. Argun takes a morning break and decides to switch from tea to coffee before returning to the "bad" authors and the sources.

This time, Argun goes for the worst first. He opens the e-mail message that announces the arrival of the latest catalog conversion. It involves updates to 35 of the 123 entries in the catalog. The attached conversion log tells the whole sad truth. Only 45 of the entries were reliably converted into Product components. The failed entries are attached in a pseudo-XML file that needs hand-working. Argun sighs to himself, "With one or two more people on the conversion staff (or just one good programmer), my job would be cut in half. Yeah right," he quickly adds. "I'm cheaper."

Argun loads the pseudo file into Microsoft Word (the only editor he really feels comfortable working in). Using the tool that he kludged together with the help of a programmer friend, he begins his work. It's not particularly hard — just tedious. He highlights a section of the document and then chooses the appropriate custom toolbar button. The button performs the appropriate tagging for the section he has selected by hand. He moves steadily through the file, marking the Product components first. As he presses the component button, the right XML elements automatically surround each block of text. Next, he makes a pass for the elements within the components that are never quite caught right — the price and the customer quotes. Finally, he presses the button that blasts through the file and stores all his marked components in the CMS database. Argun has no idea how this process actually works — only that it doesn't always work the first time.

After a late lunch, thanks to the catalog team, Argun returns to work on the stuff from the good sources and bad authors. The good source turns out to be better than he expected. It's a syndicated source of industry news items. After he's repeatedly asked for it, they finally sent him the one with keywords in it. Now, instead of taking him 30 minutes to add the right keywords to the stories, reviewing and verifying them takes him only 10. The hard part of this source — but the part that Argun likes best — is finding the places in the existing content that he can link to the new stories. This task represents one of the few times during his day that Argun feels that he's functioning at the peak of his abilities.

With the news finished, Argun moves directly to the bad authors. Mostly managers with a lot of other stuff going on, these misbehaving contributors do as little as possible and never think before typing. They put stuff in the wrong fields on the Web forms, never consider who they're writing to, and *never* fill in anything that they can get away with leaving blank. After Argun set up the forms so that they couldn't submit stuff without filling in all the required metadata fields, he got nothing (except excuses) from this lot. Brow furrowed, he plods through the bad authors' work. The only saving grace is that he can anticipate by author the kinds of mistakes to expect. He makes a little game out of seeing how many of the mistakes he can name before opening the component.

Four o'clock comes, and it's Argun's appointed hour to head off future problems. He long ago vowed to himself (and unfortunately to his boss) that he'd avoid getting plowed under the load by spending at least one hour a day heading off problems upstream. He sends his weekly e-mail reminder to the catalog team, asking again for a meeting to discuss ways that they can work more consistently. He begins work on a mapping file that he intends to use that can, eventually, automatically convert the keywords that the news agency uses to the ones that his company uses internally. He wishes that he'd had the agency's list when he started so that he could just use its terms. To change over now, he laments, would require days of retagging.

On his way out of the building, Argun casually drops by the office of one of the bad authors. He tells the manager that he has a question about something she wrote and asks whether she'd please open it on-screen. She hesitates because she doesn't know how to open it on-screen. Argun knew full well that she didn't know how to do so and counted on using that fact as an excuse to provide a little guerilla training.

At home that night, Argun makes a revision to the metatorial guide based on the one original question he received that morning. He dozes off in his big recliner reading *Women, Fire, and Dangerous Things: What Categories Reveal About the Mind* by George Lakoff.

Collection services

CMS collection services aid the collection process. The main service that they provide is to help get content into the repository. This service in turn breaks down into the following tasks:

- Authoring components directly into the CMS repository.
- Loading previously created components into the repository one at a time or in bulk.

The most common way that someone authors components directly into the repository is through Web-based forms. In these forms, an author types blocks of text, enters metadata, and uploads images and other media. Most CMS products include some tools that enable you to design a form where you can collect all the information that you need to create a particular component. To contribute to the system, an author goes to a Web page, chooses what sort of content to contribute (that is, which component class), and then begins typing. After the author clicks the Submit plowed under the load by spending at least one hour a day heading off problems upstream. He sends his weekly e-mail reminder to the catalog team, asking again for a meeting to discuss ways that they can work more consistently. He begins work on a mapping file that he intends to use that can, eventually, automatically convert the keywords that the news agency uses to the ones that his company uses internally. He wishes that he'd had the agency's list when he started so that he could just use its terms. To change over now, he laments, would require days of retagging.

On his way out of the building, Argun casually drops by the office of one of the bad authors. He tells the manager that he has a question about something she wrote and asks whether she'd please open it on-screen. She hesitates because she doesn't know how to open it on-screen. Argun knew full well that she didn't know how to do so and counted on using that fact as an excuse to provide a little guerilla training.

At home that night, Argun makes a revision to the metatorial guide based on the one original question he received that morning. He dozes off in his big recliner reading *Women, Fire, and Dangerous Things: What Categories Reveal About the Mind* by George Lakoff.

Collection services

CMS collection services aid the collection process. The main service that they provide is to help get content into the repository. This service in turn breaks down into the following tasks:

- Authoring components directly into the CMS repository.
- Loading previously created components into the repository one at a time or in bulk.

The most common way that someone authors components directly into the repository is through Web-based forms. In these forms, an author types blocks of text, enters metadata, and uploads images and other media. Most CMS products include some tools that enable you to design a form where you can collect all the information that you need to create a particular component. To contribute to the system, an author goes to a Web page, chooses what sort of content to contribute (that is, which component class), and then begins typing. After the author clicks the Submit button, the system creates a database record or XML element in the repository to store the component instance. Fancier systems can even read a DTD or relational database structure and automatically create a Web form with all the right input controls and validations.

People can also author components outside the system and load them in after they're complete. The most common tool for authoring text-based components outside the system is Microsoft Word. The usual convention is that one Word file yields one component. Although Web-based forms give you tight control and validation, open tools such as Word give you flexibility. Creating a form that spurs creativity in your authors is difficult. On the other hand, creating a word processor-based system that can constrain the creativity of your authors is also difficult. The usual method of control in a word processor is to create a template that authors must follow. The template contains spaces where the author types text, enters metadata, and specifies images and other media. In Word, these templates can be as simple as an example file that you load, modify, and then save, or as complex as template (DOT) files with fill-in fields, validation macros, and data connections to metadata lists and the CMS repository.

Finally, you can move content into the repository in bulk. In some systems, collection services connect to non-CMS databases and automatically upload and componentize database records. Other systems incorporate sophisticated services that scan the Web for targeted content and automatically download and componentize it.

The Management System

The management system in a CMS is responsible for the long-term storage of content components and a range of other resources. The management system contains the repository, workflow, and administration facilities. At the highest level, it enables you to know what you have and what its disposition is. Your management system, for example, should be capable of telling you the following:

- Details about your content, including what kinds of components you have and where in its life-cycle each is.
- How well utilized your staff is and what bottlenecks are coming up.

The Management System in a CMS

You may wonder why I use the term "managing content" to describe one of the three parts of a CMS. Here's why:

I use the term *content management* to capture all the processes involved with the procurement and use of content. You may, however, hear a popular and more constrained use of this term. Rather than using it to stand for the end-to-end process, many hear the term *content management* refer to only the part of the system that actually holds and administers the content. As one CMS product company puts it, "Our system focuses on managing the content. We let other companies focus on creating it and distributing it." I can certainly understand this usage. On the other hand, the processes for creating and distributing content seem to me so tied up in its "management" that trying to separate them too far is useless. Given this argument, I originally shied away entirely from the more limited meaning for content management. I used the term *repository system* to stand for that part of the system that stores and administers content. But no one really understood that term. After much discussion, they'd eventually say, "Oh, you mean management." Thus I decided that the best course was to trade some collision between the two meanings of content management for the fact that it's the right term in both its general sense and its constrained sense.

Still, after you're past the surface of the management system, you're likely to see that its core is a repository that holds and makes accessible the content and supporting files that you need to manage. So I use the term *management system* to refer to the part of the CMS that contains the repository, the workflow system, and the administration system. I use the term *CMS* to refer to the entire system that includes these management functions, as well as all the other processes and functions necessary to collect and publish content.

- How you're using components in publications and which content is unused or ready for removal.
- Who has access to what content and who's contributed the most.

In short, whatever reasonable questions you have about your content, publications, or collection system, you should find the answers available from your management system.

The repository

The *repository* is the main piece of the management system (see Figure 7-7).

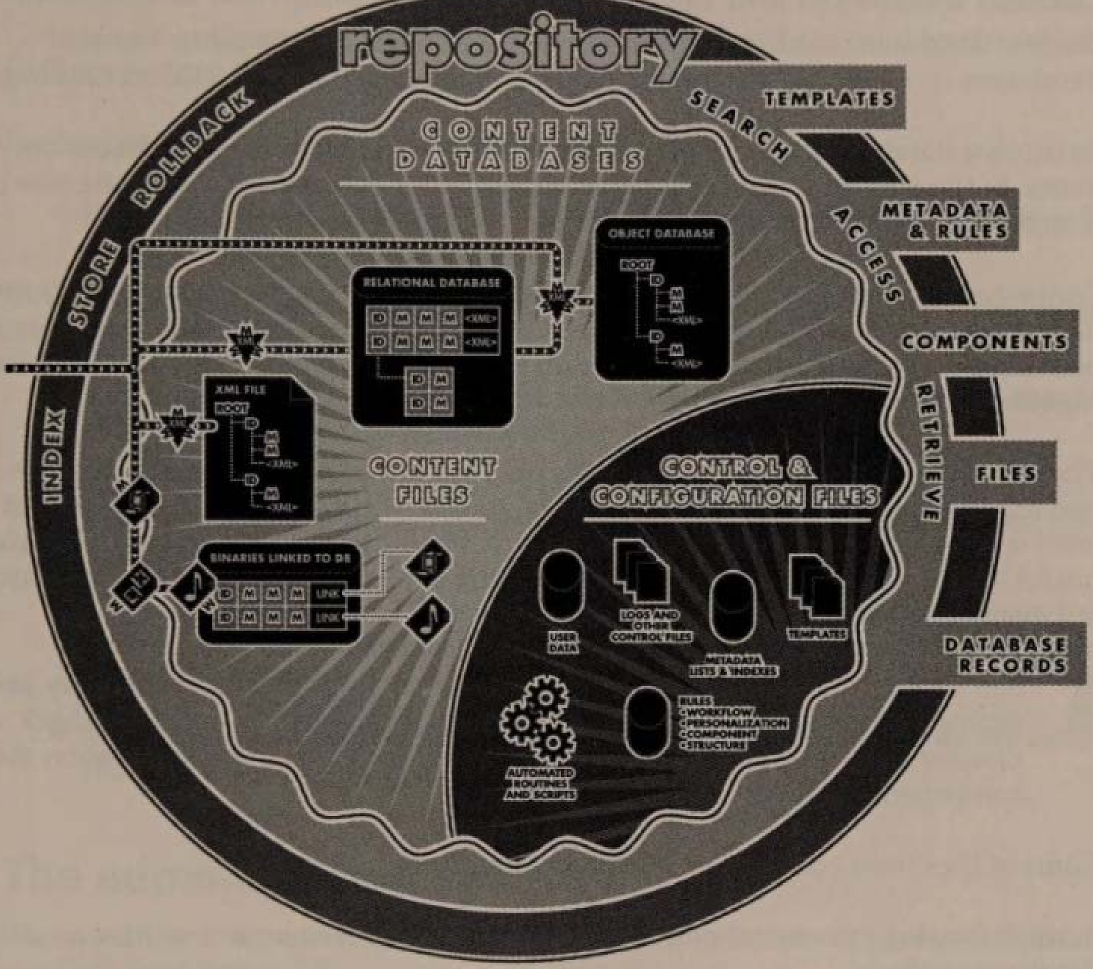

repository
STORE
ROLLBACK
INDEX
SEARCH
ACCESS
RETRIEVE
TEMPLATES
METADATA & RULES
COMPONENTS
FILES
DATABASE RECORDS
CONTENT DATABASES
RELATIONAL DATABASE
OBJECT DATABASE
ROOT
ID
M
<XML>
XML
XML FILE
CONTENT FILES
BINARIES LINKED TO DB
LINK
CONTROL & CONFIGURATION FILES
USER DATA
LOGS AND OTHER CONTROL FILES
METADATA LISTS & INDEXES
TEMPLATES
RULES
• WORKFLOW
• PERSONALIZATION
• COMPONENT
• STRUCTURE
AUTOMATED ROUTINES AND SCRIPTS

Figure 7-7: The repository contains the CMS databases as well as other storage and retrieval mechanisms.

The repository is the set of databases, file directories, and other system structures (for example, custom settings for the CMS) that store the content of the system as well as any other data associated with the CMS. Components and other CMS resources come into the repository via the collection services, and the publishing services extract them. The repository can contain the following components:

- Content databases and files
- Control and configuration files

Content databases and files

Content databases and *files* hold the system's content components. Content databases can consist of standard relational databases or XML Object databases.

Relational databases use standard tables, rows, and columns to represent components. In the simplest case, you have one table per component class, one row per component instance, and one column per component element.

Content databases can also consist of object databases, where components are fully represented as XML and are stored in a large hierarchy. In this case, component classes, instances, and elements all have their own tags, which a DTD brings together into a complete system.

These two technologies aren't mutually exclusive. You can store an XML fragment in a relational database field, for example, to give you control of the content at a level more granular than a database column. In addition, a new round of hybrid object/relational databases are now becoming available that blur the distinctions between the two.

Cross-Reference See Chapter 32, "Building Management Systems," for the complete story about storing components in relational or object databases. In addition, check out Chapter 23, "Designing Content Components," for much more information about component instances, classes, and elements.

Content files hold content outside of any database.

In the following circumstances, the repository can store content as files as well as database entities:

- You can store media files in a database, but you more often store them as binary files and link them to database records. In some ways, this procedure is a throwback to the days when databases were incapable of handling binary data very well. In other ways, having media files directly accessible through the file system is quite convenient.
- Files such as word-processing files and spreadsheets, as well as other files such as Adobe Acrobat (PDF) files that are intended for use by the CMS in their existing format, you can store and distribute as files. In this case, the CMS is functioning more as a document management system than a content management system. See Chapter 9, "The Roots of Content Management," section "The Principles of Document Management," for more information on the distinction between document and content management.
- Rather than using a database at all, the main content storage facility may consist of one or more XML files that the CMS services manage. The structure of the XML file is similar to that of the object database (classes, instances, and elements all having unique XML tags), but standard and custom XML programming techniques replace the accessing and indexing of an object database.

Control and configuration files

Control and configuration files are the noncontent files that you manage within the CMS repository.

Control and configuration files include the following types:

- **Input and publishing templates:** You retrieve and use these templates for content input or publication output, respectively.
- **Staff and end-user data files and databases:** These elements hold information that you use for access and personalization.
- **Rules files and databases:** These elements hold the definitions of component types, workflows, and personalization routines.
- **Meta information lists, content index files, and databases:** These data sources augment the metadata that you store directly in the content files and databases.
- **Log and other control files and structures** (such as system catalogs and registries): You use these files for analysis as well as to store the parameters that control CMS routines and functions.
- **Scripts and automated maintenance routines:** These elements are programs that the CMS uses to help manage content.

The administration system

The *administration system* is responsible for setting the parameters and structure of the CMS (see Figure 7-8).

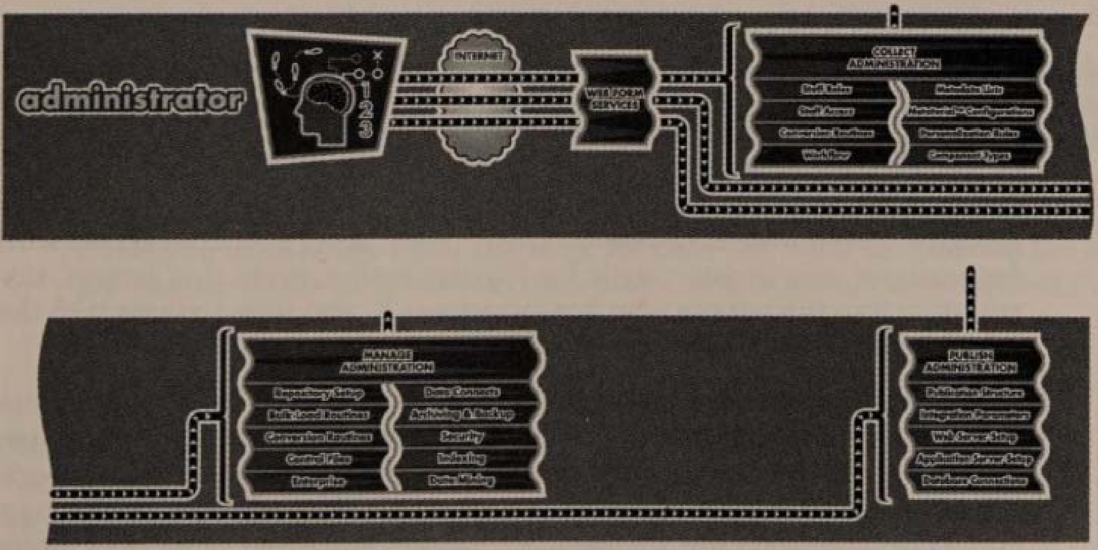

administrator
1
2
3
INTERNET
WEB FORM SERVICES
COLLECT ADMINISTRATION
Staff Roles
Staff Access
Conversion Routines
Workflow
Metadata Lists
Metatorial Configurations
Personalization Rules
Component Types
MANAGE ADMINISTRATION
Repository Setup
Bulk Load Routines
Conversion Routines
Control Files
Enterprise
Data Connects
Archiving & Backup
Security
Indexing
Data Mining
PUBLISH ADMINISTRATION
Publication Structure
Integration Parameters
Web Server Setup
Application Server Setup
Database Connections

Figure 7-8: The administration system affects all three parts of a CMS.

The administration system affects all the parts of your CMS in the following ways:

- **In the collection system:** Administration includes staff configuration, where roles and access rights are set; metatorial configuration, where the system of metadata fields and lists is maintained; and system configuration, where the structure and workflows of the content management system are maintained.
- **In the management system:** Administrators perform many of the usual database administration tasks such as user maintenance and permissions, backup, and archiving. In addition, management-system administrators perform content-specific tasks such as creating content types, performing metadata reviews, and creating workflows.
- **In the publishing system:** Administrators ensure that all the hardware and software for displaying content is working according to plan. For Web publications, the job is much like that of a site administrator who ensures that the Web server, application server, content management application objects, databases, and other associated programs are always running and never overtaxed. For other publications, the administrator watches over the publication creation process. If the publication is one that requires work by outside agencies, you may deliver publication files to them via removable media such as floppy disk or CD-ROM. The administrator will oversee the production process to ensure that the right publication files are created and delivered to the vendor that does the final production.

The workflow system

The *workflow system* is responsible for coordinating, scheduling, and enforcing schedules and staff tasks. The workflow system affects all three parts of the CMS as follows:

- **In the collection system:** Here you find workflows for content collection, creation, and aggregation tasks. In most cases, the workflow follows a particular kind of content from creation until it's ready for publication. You may, for example, create a workflow that involves creation, review, and approval tasks for policy statements that you list on an intranet. After all steps of the process are complete, a policy statement is ready for display. In some cases, you design a workflow that crosses content types. A content-conversion workflow, for example, may include steps for logging in all content that arrives, distributing it to the people who are to process it, and then passing it to the appropriate editors after conversion.
- **In the management system:** Here you find workflows for standard administrative tasks such as backup and archiving. Additionally, you find workflows for reviewing, changing, and verifying the usefulness of content. Other workflows schedule data-mining and synthesis tasks. Finally, workflows also manage the connection between the management system and other non-CMS systems that provide data to the CMS.

- **In the publishing system:** Here are publication cycles and their associated workflows to ensure that, each time you create a publication, it's as good as is humanly possible. Suppose, for example, that your Web site operates on a daily publishing cycle, during which you update news and special announcements. You may create a workflow that includes steps such as those for reviewing all pending content, performing test builds of all affected pages, testing personalization rules against new content, and changing status to publish.

Connections

You likely need to connect the management system to various infrastructure and data systems, including the following:

- **The organization's LAN and WAN environments:** You connect the management system to these environments so that you can receive and deploy content anywhere in the organization.
- **The organization's user-management system:** You need to connect to this system so that, instead of retyping user data, you can simply reference the company's directory of users. If, for example, authors of a particular sort are all in one department and have a particular job title, you can reference the entries for these folks in the organization's user administration system. Security clearances, e-mail addresses and biographical information becomes accessible as well through enhanced administrative or HR systems.
- **Company metadata systems:** If the company has a global product catalog system that's not part of the CMS, for example, you need it to supply the list of product names for the CMS to use. A direct connection to the outside system with frequent updates ensures that the metadata that you use in the CMS stays current. You may also want a two-way connection so that content that you create or modify in the CMS you can post back to the outside source.
- **Enterprise data systems:** If, for example, your CMS holds a copy of employee information records (with the master remaining in your organization's enterprise resource planning [ERP] system), you need a connection to the ERP system. The connection is a physical one that employs the LAN as well as a logical one that maps entities and relationships in the enterprise system with those you need in the CMS. You may, for example, want to combine the multiple address fields in the ERP system into a single Contact Info element in the Staff component type in the CMS.

The Publishing System

The *publishing system* is responsible for pulling content components and other resources out of the repository and automatically creating publications out of them (see Figure 7-9).

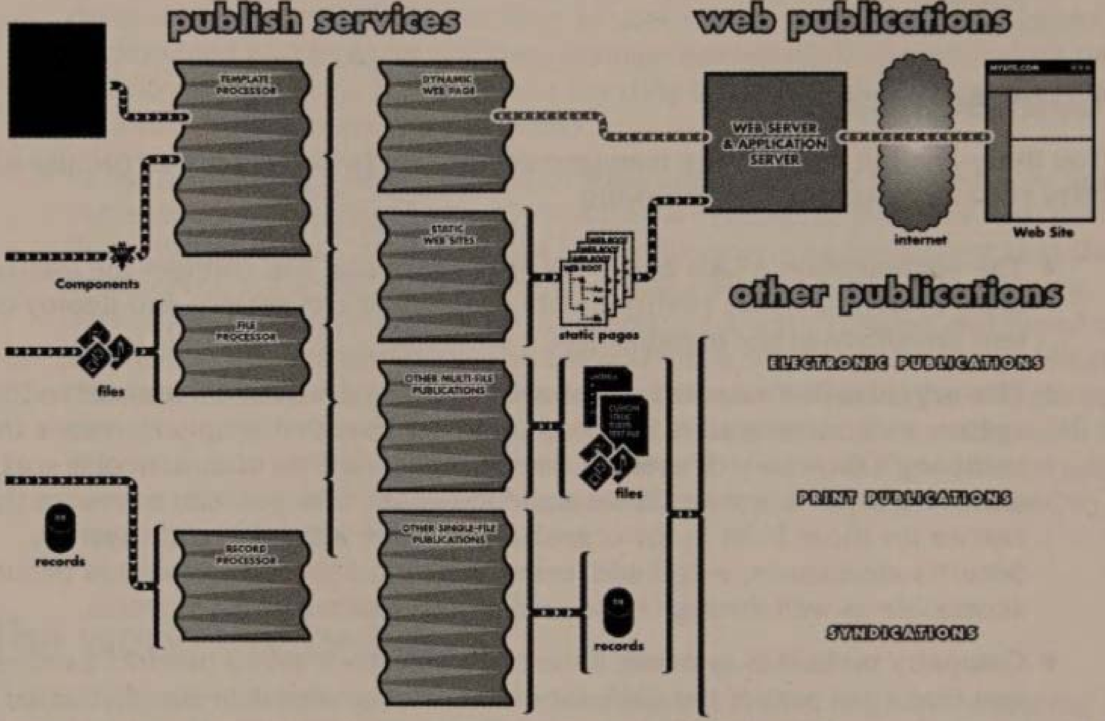

publish services
TEMPLATE PROCESSOR
FILE PROCESSOR
RECORD PROCESSOR
Components
files
records
DYNAMIC WEB PAGE
STATIC WEB SITES
OTHER MULTI-FILE PUBLICATIONS
OTHER SINGLE-FILE PUBLICATIONS
static pages
files
records
web publications
WEB SERVER & APPLICATION SERVER
internet
MYSITE.COM
Web Site
other publications
ELECTRONIC PUBLICATIONS
PRINT PUBLICATIONS
SYNDICATIONS

Figure 7-9: A publishing system uses templates and publishing services to produce Web sites as well as other kinds of publications.

Publishing templates

At the heart of the publishing system are *templates*, which bridge the gap between the neutral content in the repository and the needs of a particular publication (see Figure 7-10).

- [Template]
  - [Static Elements]
    - [HTML]
    - [Scripts]
    - [Media References]
  - [Pub Service Calls]
    - [Personalized]
    - [Get Components]
    - [Build Navigation]
    - [Other Template Logic]
    - [Convert]
  - [External Calls]
    - [Enterprise Data]
    - [Web Services]
    - [Commerce]

Figure 7-10: The publication template is a program that builds publications from the content in the repository

Publication templates are files that guide the creation of a publication from the content stored in the repository. Unlike more common word-processing templates, CMS templates are programs that use either a proprietary or an open programming language to specify publication-building logic. Templates include the following components:

- **Static elements:** These elements are text, media, and scripts that pass directly through to the publication without further processing.
- **Calls to publication services:** These calls retrieve and format components and metadata from the repository and perform other necessary functions such as running personalization rules, converting content, and building navigation.
- **Calls to services outside the CMS:** These elements integrate publications into a wider organizational infrastructure by calling in enterprise data and functionality and other Web services.

Cross-Reference

In Chapter 22, "Designing Publications," you find much more information on publishing templates.

Publishing services

Publishing services are the application logic and business services that a CMS provides that aid in the creation of publications from the content and metadata in the repository.

The publication services provide the following functions:

- **Load and execute templates:** These services process the personalization, conversion, content extraction, and navigation-building calls that the templates make to create a publication.
- **Provide publication-specific services:** These services include output to PDF for print publications or incremental updates to a Web site.
- **Provide a bridge to non-CMS services:** These are services that you need to call and that provide data that you need to include in publications.

In the case of a dynamic publication (such as a live Web site), you invoke the publishing services via a request from a browser and produce a single page. In static publications (static Web sites as well as other publications), a staff member or a prescheduled automation routine triggers the publication services that then produce a complete publication.

You may create part of a publication by using services that aren't part of the CMS by having the publishing services call them as independent software objects. Non-CMS services generally provide e-commerce transactions and access to enterprise data and other resources not under the control of the CMS. They may also augment interactivity in a Web publication by providing collaboration or other software intensive features. These services often provide functionality that appears within a user interface that the CMS publishes onto Web pages.

Connections

The publishing system often maintains connections to other (non-CMS) enterprise data systems, such as those mentioned in the previous section. Enterprise data is the organization's data that you want to use but don't want to store in the CMS repository. Examples include ERP application data, user data, and other files and databases that you maintain outside the realm of the CMS. Data from these systems you read live from the source at the time that you create a publication (which, on the Web may be constantly), and you lay it out appropriately on the publication page at that time as well.

Note You can also load enterprise data periodically from the source to the CMS repository. In this case, you consider the enterprise data source as "acquired" content.

The CMS isn't responsible for the management of enterprise data, but it can handle publishing enterprise data and publishing the functionality that enables users to interact with enterprise data systems.

Cross-Reference

In Chapter 10, "The Branches of Content Management," you can find a number of examples of a CMS handling the publication of non-CMS data.

Web publications

What most people want most from a CMS today is for it to create their Web publications (see Figure 7-11).

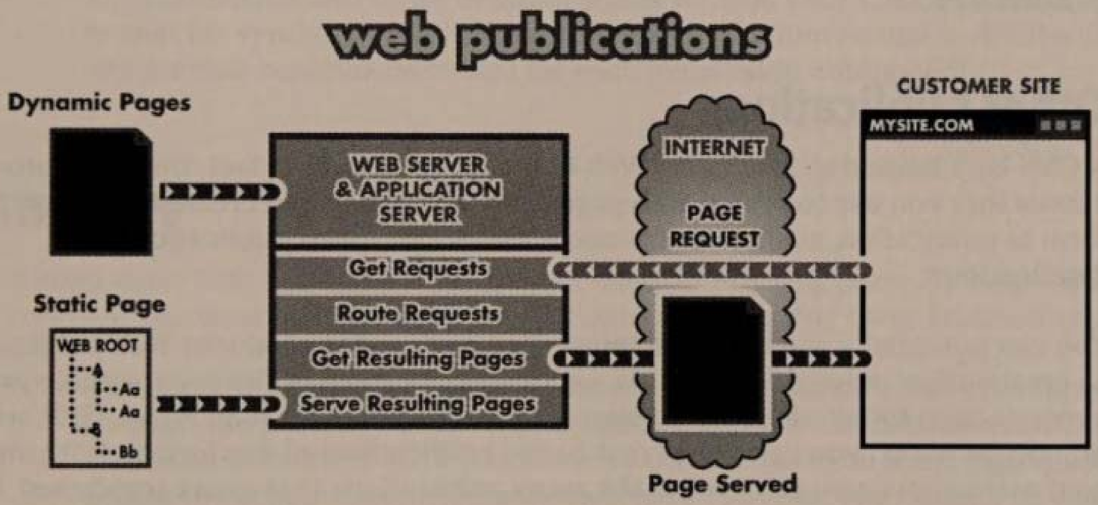

web publications
Dynamic Pages
Static Page
WEB ROOT
Aa
Aa
Bb
WEB SERVER & APPLICATION SERVER
Get Requests
Route Requests
Get Resulting Pages
Serve Resulting Pages
INTERNET
PAGE REQUEST
Page Served
CUSTOMER SITE
MYSITE.COM

Figure 7-11: The Web-publishing system can produce a fully dynamic site.

Web publications are Internet, intranet, and extranet sites that a CMS produces. If they're dynamic, the CMS produces these sites one page at a time in response to user clicks. In this case, the user's click passes a page request to the Web server that triggers the CMS publishing services to do the following:

- ✦ Load a template.
- ✦ Pass it any parameters that came along with the user's page request.
- ✦ Execute the code in the template to produce a finished page.
- ✦ Pass the finished page back to the Web server for display in the user's browser.

If they're static, the CMS produces them all at once and serves them as HTML files. The CMS administrator triggers a build of the static site using some user interface in the CMS. The CMS then calls the appropriate publishing services and templates to produce all the pages of the site.

The pages are a combination of content elements drawn from the repository by templates. The pages can also include functionality drawn from non-CMS services called in by templates and integrated into the content and navigation that the CMS provides.

Especially in dynamically built sites, the CMS publishing services are embedded within a Web server and an application server. The Web server is software that takes care of the basic function of receiving requests from the Web user and sending her results. The application server is software that provides caching, database connection pooling, and other services that help the CMS scale and increase its performance.

Other publications

A CMS isn't limited to producing Web-based publications. In fact, the same processes that you use to create Web pages you can also use to create any other form of publication, such as electronic publications, print publications, and syndications.

You can potentially use the same templating engine that produces Web publications to create other publication formats. Unfortunately, most of the commercial systems in production today offer limited support for formats other than HTML. Still, with a little extra effort, you can create text-based publications of any format. With significant extra effort, you can even make many publications that aren't text-based. Of course, many of the publications that you may want to produce are HTML or variations on XML.

Note

If at all possible, you should seek to avoid manual post processing. *Manual post processing* is where you must change by hand the publication files that the CMS produces. You may, for example, need to manually number the figures in a print publication. As soon as the CMS and supporting automated routines can't automatically produce the publication file, you have two content branches. One branch is the master content that you store in the repository. The other branch is the content that's in the publication file which may contain the changes that you make. If one branch changes, you need more manual effort to update the other branch. Not a happy situation.

Publications that aren't destined for the Web include the following:

- **Print publications:** For print publications, the appropriate components and metadata are drawn out of the repository and converted to the format that the print system expects. Templates and conversion routines, for example, could create a FrameMaker (MIF) file that goes to a printer for publication. Other print formats such as QuarkXPress and Adobe Acrobat (PDF) files require that you use specialized tools (some of which you need to create yourself) to fully automate the process, such as macros or scripts.

- **Electronic publications:** These publications are static Web sites that you distribute on CD-ROM or any other type of CD-ROM- or network-based multimedia system. The CMS uses templates and services that can produce the sort of text and media format and navigational structure that the electronic publication requires. Microsoft Help files, e-mail files, and the variety of formats now under development for wireless phones and Personal Digital Assistants (PDAs) are examples of non-HTML electronic publications
- **Syndications:** Syndications are sets of content components that you publish for distribution and reuse in publications outside your CMS. The most useful format for syndication is XML, but the most common format is ASCII with a header that contains metadata for each syndicated component.

Summary

A large scale CMS is a complex system of many interacting parts. To help divide and conquer this complexity, I break the CMS into the following three interacting parts:

- The *collection system* shepherds content from an author or an existing source through the process of conversion and aggregation. *Conversion* maps the format and structure of the information to the one that you want. *Aggregation* brings the information into the component, editorial, and metatorial frameworks you establish.
- The *management system* provides an administrative infrastructure. This infrastructure includes a repository for storing content and necessary system files, an administration module for setting up and maintaining the CMS, and a workflow module for defining the tasks and processes that you expect to occur.
- The *publishing system* uses templates to extract appropriate content from the repository and fashion it into a publication. Although Web sites are the main event for today's CMSs, with a bit of extra effort, you can create almost any other kind of publication that you want.

In the next chapter, I outline the reasons why an organization would go through all the effort that a large CMS requires.

✦ ✦ ✦

CHAPTER 8

Knowing When You Need a CMS

Any organization that creates publications practices some form of content management. Even the sole proprietor organizes files on her hard drive and tries to keep track of her content and share it across publications. The sole proprietor, however, has little need for the formality and tight structure that I present here. But if the sole proprietor grows to a small organization and then to a large organization, file-system directories and informal content sharing begin to cost too much and take too long. A content management system (CMS) may then become necessary to help organize and automate the process.

You need a CMS if your collection, management, and publishing processes are too complex to manage informally.

You can gauge complexity by the following guidelines:

- **The amount of content that you have.** This amount includes not only the number of items, but also the number of kinds of items that you're trying to manage.
- **The amount of contribution.** Not only the number of contributors counts, but also their relationship to you as well as the complexity of the existing content that you receive.
- **The amount of change that you expect in your content.** Change can occur in the amount of content you add or remove from the system or in the amount of design change that you intend to support.
- **The number of publications that you intend to create.** Complexity in publication depends on the number of different publications that you intend to create, as well as on the degree of personalization that you intend to implement.

◆ ◆ ◆ ◆

In This Chapter

What having a lot of content means

How your sources of information affect your need for a CMS

How much change warrants a CMS

Publications and personalization and their effect on your CMS

How you can roughly estimate your need for a CMS

◆ ◆ ◆ ◆

Figure 8-1 illustrates what you need to look for to determine whether you need a CMS.

Too much content • Content items • Content types	Too many contributors • Diverse authors • Complex sources
Too much change • Content throughput • Design revision	Too many publications • Content channels • Personalization

Figure 8-1: I break the complexity of a content problem into four areas of concern.

Gauging the Amount of Content

You need a CMS if you have too much content to process by hand. Although the concepts of content management can help you with even small projects, until you have a big system, you can't realize enough cost savings to justify the effort that constructing a solid CMS involves.

What is a "big" system? A big system is one that's too big to fit in one person's head. Say, for example, that, in the past, you counted on your Webmaster to simply know what content's on your site and where it all goes. Maybe you recently started to notice that she can no longer keep up with the influx of information and can no longer quickly assess and place new information on your site. You may now need a system to help the Webmaster (or, at least, some other people to help). In particular, a big system has the following characteristics:

✦ **A lot of content:** Based on my experience, by the time that a Web site reaches 1,000 pages, it's clearly too big for anyone to manage informally. This number, however, is far from standard. You must view the sheer size of the content base in the context of all the other factors that I list here to finally decide whether your situation warrants a CMS. And although pages make up the usual unit of site size, the number of content components on your site is a better measure of the size of the content base than is the number of pages.

- **A lot of content types:** A content base of 1,000 components, all of the same type, is obviously easier to manage than a base of 1,000 components of five different types. For each component type, you must invent, implement, and oversee a corresponding collection, management, and publishing process. The more types that you have, the harder this task becomes to accomplish without help.

Cross-Reference In Chapter 23, "Designing Content Components," in the section "Components vs. pages," I go into detail about the contrast between components and pages.

You manage 1,000 items of content the way that you manage 1,000 items of anything: by arranging them into groups and storing them in a place where you can find them by type or by other key qualities (color, size, and the like). Content management systems provide the tools to divide content into like chunks (or components), arrange them into groups (in outlines and indexes), and store them in a place where you can get to them (in databases and XML files). By using a CMS to break 1,000 components into five types, you reduce the number of things that you need to know from 1,000 to five. Rather than needing to know what's in each of the 1,000 components, you need to know only what's in each of the five types. Reducing the complexity of a body of content so that you can again understand and master it is what content management is all about.

As you're measuring the amount of content you have, don't forget to include multiple languages or localities. If you have 100 items of content, for example, but you must deliver them in 10 languages, you really have 1,000 items of content to deliver.

Managing the Size of the Contribution Base

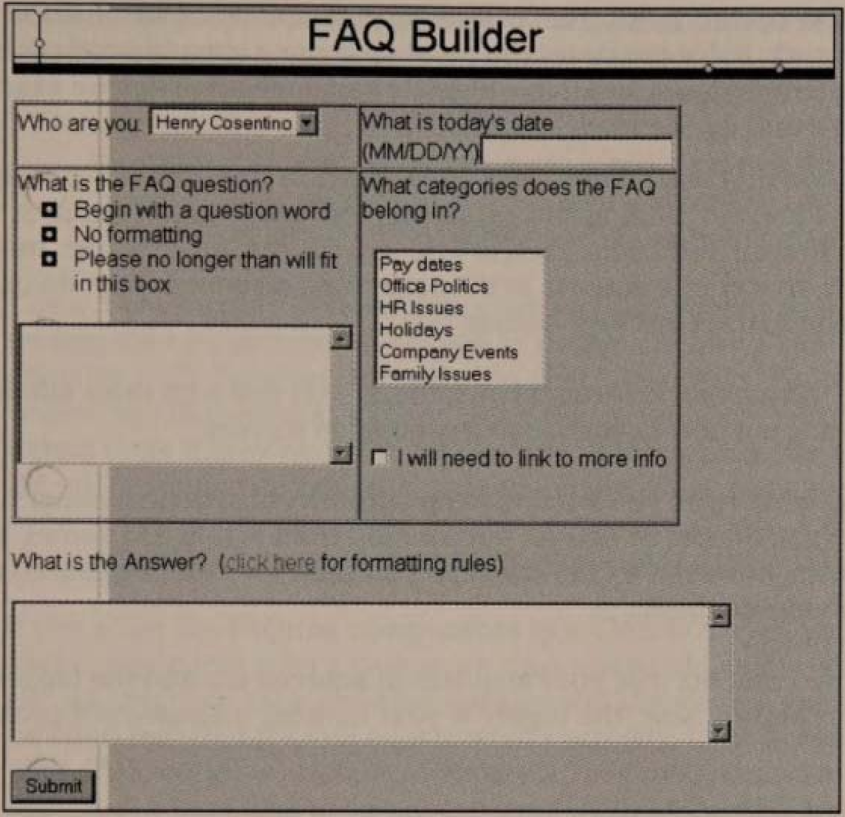
If one person creates all the content that you provide, you can usually count on her to create her own rules and follow them. As soon as you involve two people in the creation process, however, they need to agree on these rules and not change them without prior notice. If the system expands to dozens or hundreds of contributors, a much more complex system of automatically enforceable rules becomes necessary.

Suppose, for example, that you have 30 people throughout the organization whose task is to create Frequently Asked Question items (FAQs). Without further instruction, they all go their merry ways, creating FAQs in any way that they choose. The result is 30 different types of FAQs that you must reconcile. If, on the other hand, you're using a CMS, you can define a FAQ content class as follows:

- FAQs have the following required elements: question, answer, date, submitter, and subject area.
- Questions may run no longer than 256 characters and may contain no formatting. They must start with a question word (who, what, where, when, why, or how).
- Answers can run no longer than 3,000 characters with an optional link to more information. They can contain only the formatting codes that you specify in a document.
- Dates must appear in the correct format (MM/DD/YYYY).
- Submitter name must match exactly one of the names on the list of submitters.
- Subject areas must come from your accepted list. A FAQ must appear in at least one subject area but can appear in others as well.

You can probably see that simply writing and publishing these rules goes a long way toward standardizing FAQs. With a CMS at your disposal, however, you can turn these rules into part of the fabric of submission, as the following steps describe:

1. First, you can specify in the CMS that you have a content class that you call FAQ. As an author submits a piece of content, that author must specify whether it's a FAQ or something else. No space is available for submitting content that's not of some recognized class.
2. Second, you specify in the CMS that the FAQ class displays certain elements. If an author submits a FAQ, no discussion about what a FAQ consists of is possible. Either it displays the elements that the CMS knows about or it's not a FAQ.
3. Third, you specify in the CMS what values you allow for each of the elements of a FAQ so that the CMS can validate the content that an author submits. If an author types a date for a FAQ, the CMS can validate it and decide whether it's an acceptable value for the date element. If it's not an appropriate value, you can ask the author to correct it before submitting the item. For textual content, you have somewhat less control. Authors can, if they choose, type nonsense and you have virtually no way to validate whether the content has any meaning. You can, however, set limits as to the length of content in a text field.
4. Finally, the CMS can give you a convenient and easy-to-use way to present and facilitate the submission process. You can, for example, create a Web-based form for FAQs that neatly lists all the elements and requirements (see Figure 8-2).



FAQ Builder

Who are you: Henry Cosentino

What is today's date (MM/DD/YY)

What is the FAQ question?

- Begin with a question word
- No formatting
- Please no longer than will fit in this box

What categories does the FAQ belong in?

Pay dates
Office Politics
HR Issues
Holidays
Company Events
Family Issues

☐ I will need to link to more info

What is the Answer? (click here for formatting rules)

Submit

Figure 8-2: A hypothetical Web form for submitting FAQs

Where limitations exist, the form can state them. Where choices are constrained (for example, to lists and check marks), the form can make them simple to specify. Many commercial CMS products provide strong support for Web-based forms that seriously reduces the time necessary to create and maintain them.

Cross-Reference See Chapter 24, "Accounting for Authors," for more information about figuring out how to collect material from authors.

In addition to authoring it, most organizations acquire information as well. Acquired content is originally created for some other purpose. To bend acquired content to your own purpose requires work. The amount of work depends on how different the content source is from what you want, as the following list describes:

- If you must convert the source from a format that's different from the one that you need for your system, that process requires work. Such conversion can prove quite difficult if, for example, your source is Microsoft Word format, and you must convert it to XML.

- If one unit of the source yields multiple content components, acquisition will require more work. If, for example, you must split one source document into three articles, two reviews, and five editorials someone must decide exactly where to split it and do the work to split it.
- If the information that you want to find and extract in the source is poorly marked or isn't present, finding and extracting that information requires work. If you need to find an author for each article in a source, for example, but the source doesn't always cite authors in the same way, someone must do extra work to find and extract authors' names.

Cross-Reference

In Chapter 25, "Accounting for Acquisition Sources," you find a lot more information about figuring out how to analyze your acquisition sources.

A CMS can't tell you what rules you need to keep authors consistent. Neither can it tell you how to convert, divide, or extract information from acquired sources. After *you* decide these rules, however, a CMS can help you keep the rules organized and can really help you enforce them.

In any case, the more complex that your acquisition sources are and the bigger and more diverse is your author base, the bigger is your content management problem.

Anticipating the Amount of Change

Most people focus on the volume of content they have. In fact, after the system is up and running, the amount of change, not the volume, is what determines the complexity of a system. A 1,000-component system is simple if it never changes. The following two sorts of changes most directly affect complexity:

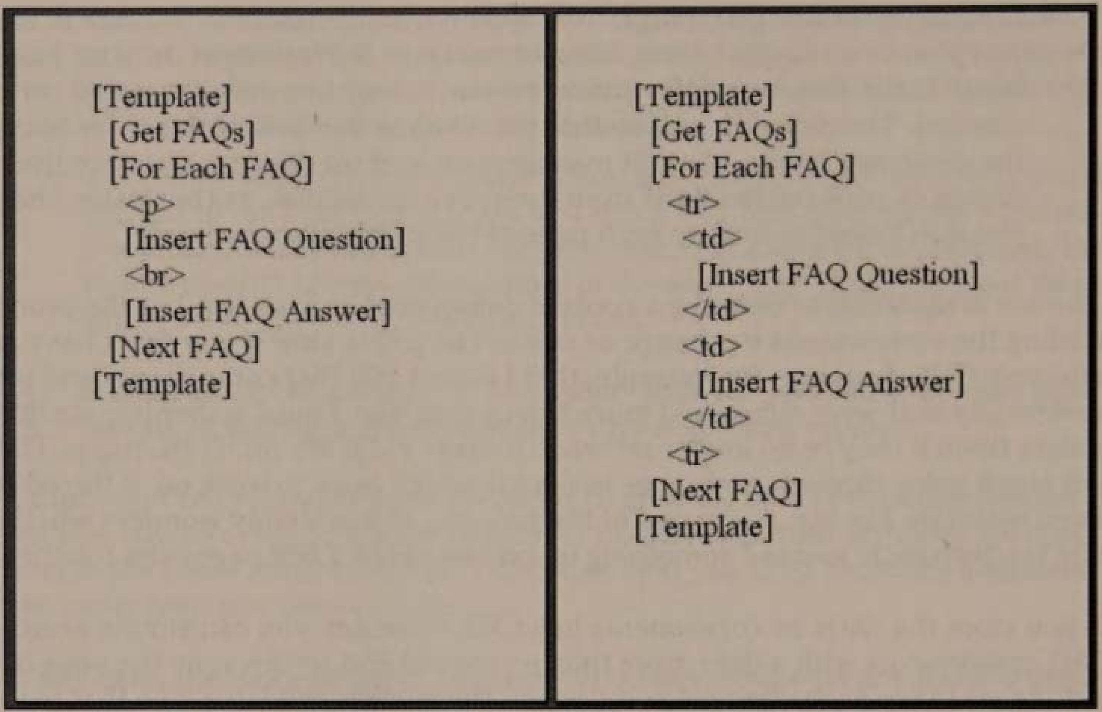
- **Component throughput:** This type of change refers to the number of content components that you modify in some way per unit of time. If, for example, you intend to create 100 new components per week and believe that about 50 per week are sure to become outdated and need deleting or archiving and that about 50 more need editing in the same period, your total throughput is actually 200 components per week.

- **Frequency of design change:** This type of change refers to the rate at which you plan to do major layout, look, or navigational redesigns on your publications. In the Web world, for some level of redesign to occur monthly isn't unusual. The more frequently that you change the design of your publications, the more you need a content management system that can separate the design of your publications from their content so that, as the design changes, you don't need to modify each page of the publication by hand.

The act of updating or deleting a content component can be fast, but the process of finding the components to change or delete can prove slow if you don't have an effective CMS. Suppose, for example, that I have 1,000 FAQ components and want to review any that were submitted more than a year ago. I must either update them or delete them if they're no longer relevant. If these FAQs are on HTML pages, I may get stuck going through each page to decide which ones to work on. If the editor is even remotely like me, at the end of the process, she probably wonders whether she inadvertently messed something up on any of the 1,000 pages she touched.

If you store the FAQs as components in a CMS, however, you can simply search for FAQ components with a date more than a year old and review only the ones of interest. As you save each changed component, the system validates it so that you can be confident that it's still constructed correctly. Even better, you can possibly use your CMS workflow module to send you an e-mail reminder message after a FAQ ages past a year. The message can even include the FAQ and a link to it in the CMS. If it's okay, you just delete the message. If the FAQ needs changes, you click the link in the e-mail message and go directly to the CMS Web form for that FAQ, where you can do your work.

Outside a CMS, making design changes can become a nightmare. Simply moving a button from one position to another may require a Web editor to open and make the same change on all the pages of the site. In a CMS, design changes become quite easy. Suppose, for example, that you want to change the layout of your FAQ page from answers below questions to answers to the right of questions in a table. If your FAQs are in an HTML file, you need to perform an extensive edit of the page and undoubtedly mess something up. If, on the other hand, you store your FAQs in components in a CMS, the CMS builds the HTML FAQ page automatically from a page template (see Figure 8-3).



```
[Template]
  [Get FAQs]
  [For Each FAQ]
    <p>
    [Insert FAQ Question]
    <br>
    [Insert FAQ Answer]
  [Next FAQ]
[Template]
```

```
[Template]
  [Get FAQs]
  [For Each FAQ]
    <tr>
      <td>
        [Insert FAQ Question]
      </td>
      <td>
        [Insert FAQ Answer]
      </td>
    <tr>
  [Next FAQ]
[Template]
```

Figure 8-3: This illustration shows two templates for the same content. A small change in a template can make a global change in design.

The templates that I provide in the figure show the small bit of code that inserts each question and answer. For each FAQ, the template on the left inserts an HTML paragraph break (`<p>`), following it with the question, an HTML line break (`<br>`), and an answer. The template on the right goes through every FAQ, too. Instead of separating the questions and answers with a line break, however, it embeds them in an HTML table.

In a minute or two of edits to the template on the left, you can change it to the template on the right, thus changing the layout for any number of pages. You don't touch any FAQs, so you can't possibly mess any up. In addition, you can try different table layouts and styles very quickly simply by changing a few lines in the template. Of course, not all templates are this simple, and even the ones that I show here aren't actual code but just short descriptions that stand in for code. Still, you can see that, with the help of templates, design changes needn't prove hard or scary.

Knowing the Number of Publications

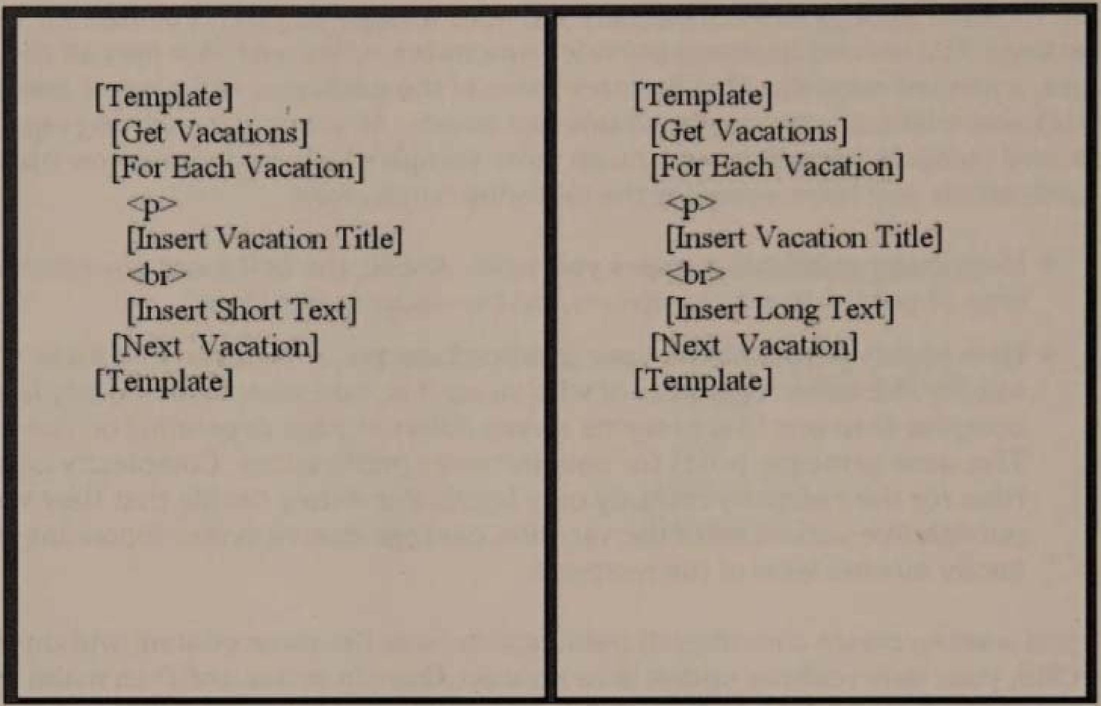
Even a small system can become complex if you need it to create a number of distinct publications. Compare, for example, two businesses that both sell the same

100 vacation packages. One company provides a single printed brochure for each package. The second business provides brochures, a Web site that lists all the packages, a printed magazine that features some of the packages, and a set of one-page, black-and-white, electronically faxable fact sheets. All other factors being equal, the second company clearly faces a much more complex task. In gauging how many publications you have, consider the following two factors:

- **How many publication types you have.** Above, the first company offers one type of publication (a brochure) and the second uses four.
- **How highly personalized your publications are.** A Web site that looks exactly the same regardless of who views it is, for example, obviously less complex than one that presents a very different page depending on the user. The same principle holds for nonelectronic publications. Complexity certainly rises for the company creating only brochures if they decide that they want to publish five variations of the vacation package descriptions, depending on the family income level of the recipient.

If you want to create a number of publications from the same content without using a CMS, your only realistic option is to produce them in series and then make independent updates to each. The vacation company with four publications, for example, is likely to first create a set of brochures with all the right facts and pictures, as well as all the formatting and layout necessary, on a glossy brochure. They then pass the source files for the brochures on to three teams (or, as likely, one person wearing three hats). The Web team strips away all the formatting and layout and rearranges the brochures into Web pages. The magazine team reintroduces all the extra text that the brochure creator needed to delete for lack of space and also dismantles and reassembles the brochure. Finally, the fax team does a similar reconstruction of the brochures, simplifying the layout and deleting most of the graphics that don't transmit well via fax. If all goes well, four publications may prove only twice the work of one. But all rarely goes well. Prices change, times change, and resorts send in better photos. Each publication must deal with these changes independently. You have no way to avoid each change becoming four times the effort. Across a large number of vacation packages, this extra effort can represent a huge cost (not to mention the cost of customer service and angry customers because of inevitably inconsistent information).

Using a CMS streamlines this process enormously. The CMS can contain a set of Vacation Package components. Each component contains enough elements to cover the needs of any of the publications. A Vacation Package component can, for example, contain both a Short Text and a Long Text element. Short Text is what you need for the fax version, while Long Text is for the Web site (see Figure 8-4).



```
[Template]
  [Get Vacations]
  [For Each Vacation]
    <p>
    [Insert Vacation Title]
    <br>
    [Insert Short Text]
  [Next Vacation]
[Template]
```

```
[Template]
  [Get Vacations]
  [For Each Vacation]
    <p>
    [Insert Vacation Title]
    <br>
    [Insert Long Text]
  [Next Vacation]
[Template]
```

Figure 8-4: Small differences in templates can make very different publications.

To create the two publications, the CMS simply uses two different templates. The one on the left in Figure 8-4 inserts the Short Text, and so you can use it to create the faxes (given that your system can fax HTML). The one on the right inserts the Long Text. You can use it to create Web pages from the same Vacation Package components. Of course, templates generally involve much more than I show here, but the idea is the same. The same content produces two very different publications depending on which template you use.

You do incur an up-front cost for developing these templates, but much of it's the same cost that you incur designing these publications in the first place. So the cost of developing the four publications in a CMS is more than the cost of developing them separately — but not necessarily *that* much more. (Your mileage may vary.)

The real advantages to a CMS approach come after you start to produce publications. As content changes occur, they occur once in the component in the CMS repository. Then you simply hit the button to rebuild your publications. Or, for dynamic publications, after each updated component is approved, it automatically appears for any Web visitor who requests that page. Any time that you create a new publication, it's guaranteed to contain the most up-to-date information. Your staff is less stressed, thereby, and your customers are more satisfied.

Note You must wait a few years for a CMS that can update magazines and brochures that are already printed. It's hard to make ink on a page rearrange itself.

Cross-Reference In Chapter 22, "Designing Publications," you find more information about multiple publications and the use of templates.

Personalization adds an entirely new dimension to publications. On the Web, you're not surprised anymore that a site knows your name and offers you content different from that for other users. Even in the print world, the custom printing of magazines or catalogs that offer a look and content specific to unique customer segments is becoming more common for many publishers.

Cross-Reference In Chapter 10, "The Branches of Content Management," in the section "Personalization," I fully detail the relationship between personalization and content management. Here I simply want to make the point that, the more personalization you require, the more a CMS can help.

The whole point of personalization is to locate and deliver content that's well-targeted to a particular audience. If you've ever tried to target an audience without the tagging, storage, and retrieval functions of a CMS, you no doubt found that you must create the equivalent of a CMS yourself. By using a CMS, especially one offering strong personalization tools, the job becomes much easier.

The degree of personalization that you want to achieve matters. If you simply want to put someone's name at the top of a Web page, you don't need a CMS. If, however, you want to restructure the look and feel of a page and lay out targeted content depending on who's visiting, you need a CMS.

Cross-Reference In Chapter 22, "Designing Publications," in the section "Analyzing Publications," I cover how you can accomplish personalization in a CMS.

Estimating Complexity

Content management isn't an exact science. It offers no real governing equations and laws. Still, for the purposes of level-setting and estimation, I propose the following formula to yield a rough guide to the complexity of a content system:

```
(Authors) X (Sources) X (Components) X (Types) X (Throughput) X
(Publications) X (Personalization) X (Redesigns)
```

Table 8-1 explains the various factors and values of this formula.

Table 8-1
The Factors of a Content System

Factor	*Description*	*Value*
Authors	The number of authors you intend to include.	For fewer than 20 authors and no complex information sources, the value is 1. For more than 20 authors, the value is the number of authors divided by 20.
Sources	The number of complex sources.	For one or zero complex sources, the value is 1. For two or more complex sources, the value is the number of sources divided by 2.
Components	The number of components (or pages if you can't count components) in the system at its inception	For 1 to 500 components, the value is 0.5. For 501 to 1,000 compontents, the value is 1. For more than 1,000 components, the value is the number of components divided by 1,000.
Types	The number of component types in the system, calculated as in column 3.	For one to three types, the value is 0.5. For four and five types, the value is 1. For more than five types, the value is the number of types divided by 5.
Throughput	The number of components created or modified per week plus the number of components archived or deleted per week.	For zero to 25 components, the value is 0.5. For 26 to 50 components, the value is 1. For more than 50 components, the value is the number of components divided by 50.
Publications	The total number of different publications the system must create.	The number of publications.
Personalization	The degree of personalization that you intend for your publications.	A value of 1 indicates no personalization; 2 indicates that you personalize for a small number of large user segments; 3 indicates that you personalize for a large number of small user segments; 4 indicates that you personalize for each user. In all cases, the personalization needs to be more than simply putting someone's name at the top of a page.
Redesigns	The number of major layout, look, or navigational redesigns that you plan to perform per year.	For one to two redesigns, the value is 1. For more than two redesigns per year, the value is the number of redesigns divided by 2.

The way that the equation works is to multiply all the factors that influence complexity, as follows:

- All the factors have some method for arriving at a quantity (components per week, redesigns per year, and so on).
- Some factors also have a scaling value so that you can compare their quantities on a par with the rest. The number of components, for example, you divide by 1,000. If you don't divide by 1,000, the effect of one new component on the overall complexity of the system is the same as the effect of one new publication. From another angle, you can say that the effect of adding a single component to the system is one one-thousandth as great as the effect of adding a publication. (I created these scaling values based on intuition and experience.)
- I've imposed values on the components, types, and throughput if they're below threshold values. I add these thresholds so that the factors don't gain undue influence over the others at small values. For example, for a system with 50 components the components value is 0.5 because of the threshold I imposed. Without the threshold, the value would be 0.05, which would unduly bring down the overall value of the equation.

As you gain experience with your own system, you can tweak the scaling factors to best reflect the relative importance of the factors that you see. You can also add factors that you determine affect complexity in your situation.

I designed the equation so that a result of one is the break point for recommending a content management system. At this value and higher, a content management system is likely to prove a valuable asset in managing the complexity that you face. For results of less than one, the value of a content management system is questionable, and as the result increases from one, the necessity of a system increases.

Note I did not impose caps on the values of the factors. Therefore, the overall value of the equation can get very high very quickly. Don't be surprised if your large systems score in the thousands.

In the following sections, I run through a few scenarios that illustrate the use of this formula.

Vacation company "A"

I start by following the example of the brochure-only vacation company. In addition to what I've already stated, assume that the company also wants to do the following:

- Produce or change two new brochures per week.
- Redesign the brochures two times per year.
- Have only one person do all the authoring.

Their data arrays as follows:

Complexity Variables

Authors	*Sources*	*Components*	*Types*	*Throughput*	*Publications*	*Personalization*	*Redesigns*
1	0	100	1	2	1	1	2

Complexity Values

Authors	*Sources*	*Components*	*Types*	*Throughput*	*Publications*	*Personalization*	*Redesigns*
1	1	.5	.5	.5	1	1	1

The calculation is as follows:

```
[1] x [1] x [.5] x [.5] x [.5] x [1] x [1] x [1] = .125
```

At a complexity value of .125 (1/8), this company's need for a CMS is small. It may benefit from getting more organized, but unless its employees plan to change what they're doing, I see no reason for them to invest the time or money necessary to put a CMS in place.

Vacation company "B"

Now I'm looking at a second vacation company. Say that the company now acts as a broker for many vacation package companies, performing the following tasks:

- ✦ It handles 600 vacation packages of four basic types.
- ✦ Each week, 30–40 of these packages change in some way.
- ✦ The company's only publication is a Web site, which it wants to redesign two times per year.
- ✦ The company has two authors and one complex source (say, a set of printed materials) to digitize and tag.

The data on this company arrays as follows:

Complexity Variables

Authors	*Sources*	*Components*	*Types*	*Throughput*	*Publications*	*Personalization*	*Redesigns*
2	1	600	4	35	1	1	2

Complexity Values

Authors	*Sources*	*Components*	*Types*	*Throughput*	*Publications*	*Personalization*	*Redesigns*
1	1	1	1	1	1	1	1

The calculation is as follows:

```
[1] x [1] x [1] x [1] x [1] x [1] x [1] x [1] = 1
```

At a complexity value of 1 (which I rigged, of course), this company could consider creating a CMS. A value of 1, however, doesn't indicate that the company's need is overwhelming. It would definitely benefit from the sort of thinking that a CMS requires, but as likely as not, it can still get by with mostly manual processes.

The key question for this company is "Which way are we going?" If the numbers are steady, a CMS is overkill. If the numbers are rising, the company may want to start planning before the system starts to overwhelm its personnel.

Vacation company "C"

Finally, I look at a vacation company with a big need. Say that it wants to dominate the vacation package industry by performing the following tasks:

- It plans to handle 1,600 vacation packages of 10 types.
- Each week, 100 of these packages change in some way.
- In addition to its Web site, which it wants to redesign two times per year, the company plans to produce a printed catalog and a set of data sheets that it can send out via fax.
- The company wants to personalize to at least the level of broad user segments by promoting the packages and extras that they believe each of these four audiences will be interested in.
- It has five authors and four complex sources of information (and, as luck has it, each vacation-package supplier has its own hard-to-convert format).

This company's data arrays as follows:

Complexity Variables

Authors	*Sources*	*Components*	*Types*	*Throughput*	*Publications*	*Personalization*	*Redesigns*
5	4	1600	10	100	3	2	2

Complexity Values

Authors	*Sources*	*Components*	*Types*	*Throughput*	*Publications*	*Personalization*	*Redesigns*
1	2 (4/2)	1.6 (1600/1000)	2	2 (100/50)	3	2	1

The calculation is as follows:

```
[1] x [2] x [1.6] x  [2] x [2] x [3] x [2] x [1] = 76.8
```

At a complexity value of 76.8, this company should definitely consider creating a CMS. I suspect that the five authors are straining to keep up with the workload, the redesigns are late, two of the three publications are out of date, and unless its employees are very organized, the company's lack of a system shows in the quality of its publications.

If it's like most of the aspiring companies I've seen, this situation is only the start. Management wants to see more publications, faster redesigns, and a higher throughput of information. If so, the managers are in just the right frame of mind to pay for a CMS.

Evaluating your results

Table 8-2 summarizes these examples into a general framework for evaluating your own situation.

Table 8-2
Evaluating Your Need for a CMS

Complexity	*Your Need for a Content Management System*
Below 0.25	Little need for a system.
0.25 to .05	Could begin thinking about a system if you believe that your need may grow.
0.5 to .075	Should begin thinking about a system if you believe that your need is likely to grow.
0.75 to 1.0	This range is the beginning of the gray zone between a need and no need. In this range, go with your intuition about what you need. Take into account whether your complexity may grow. If yes, this point is a good time to begin a content management analysis in preparation. If your need isn't likely to grow, a system may well prove more effort than it's worth.

Complexity	*Your Need for a Content Management System*
1.0 to 1.5	This range is still in the gray zone. A system, however, is more likely a need than not in this region. Even if your complexity doesn't expect to grow, a small system may save more than it costs.
1.5 to 10.0	A system is recommended. You can perhaps start slowly or cover the factors that have the highest values, but eventually you want to implement a system.
Above 10.0	A system is necessary. You're likely experiencing content management problems now. If your complexity is much more than 10 and is still growing, you may need to act quickly to either control complexity growth or implement the parts of a system that relieve the highest pressures. As your complexity becomes much greater than 10 (say 1000) your need may become acute.

I want to emphasize again that I've designed this method to help you estimate, not pin down, your need. At its core is the assertion that a set of factors affect complexity and, therefore, the need for a CMS. Whether the factors that I present are the complete set for your situation and whether the scaling that I provide adequately reflects the relative importance of these factors in your organization and content base is really up to you to decide.

In addition to giving you a ballpark estimate of your need for content management, this formula also helps you focus on the capabilities that you need to find in whatever system you choose. If your need for level 4 personalization is what's driving your complexity number so high, for example, you can immediately assume that your solution must be very apt at personalization. And you can begin to apply whatever sort of near-term solution that you can find to personalization to buy you time to implement a full system.

Summary

If you ask most people why they may need a system to measure their content, they're likely to say, "Because we have too much." That point, however, is just the beginning of the story, as the following list indicates:

- ✦ Too much content can mean too many items or too many types of items.
- ✦ Too many contributors or too much hard-to-process information spurs the need for a CMS.

- Too much change in the content or its design in publications is also a reason to use a CMS.
- Finally, too many different publications or a high need for personalization in your publications can prove reason enough to pursue a CMS.

Using the rough calculation that I provide in this chapter, you can begin to assess your own need for a CMS and decide whether—or when—you need to make the leap.

In the next chapter, I trace some of the industries and disciplines that contribute to the concepts and practices of content management.

✦ ✦ ✦

CHAPTER 9

The Roots of Content Management

◆ ◆ ◆ ◆

In This Chapter

How collection and publications came about

How document management could have captured content management

Where the IT department leaves off and content management picks up

What the multimedia industry started

Content management in the documentation back room

Why librarians are the gurus of the future

How programmers stand behind all of CM

What the marketing mentality has to do with content management

◆ ◆ ◆ ◆

I didn't invent the concepts of content and content management. Rather, what I'm trying to do is synthesize the best of the concepts that I've found in various fields into a coherent framework that you can apply to the information problems that you confront. In this chapter, I discuss these older disciplines and concepts and tie them into the framework that I've constructed around content management in earlier chapters of this book. I apologize in advance for the disciplines I've left out and for any misrepresentations that someone more fully versed in these fields may detect.

The Knowledge Base of the Publishing Industry

The point of content management is to create good publications. The definition of a good publication comes directly from the publishing industry, which has known for a very long time what is necessary to deliver content in a compelling way to a particular audience. In addition, the publishing industry has laid the groundwork for much of the collection process that I describe. On the other hand, the publishing industry is mostly concerned with handcrafted single publications. This focus gives the industry the luxury of collecting content with a very particular output in mind. Because of the direct connection between collection and publication, the publishing industry hasn't generally dealt with the abstractions of metadata, content components, and repositories.

The publication

Publishing—which I define in Chapter 22, "Designing Publications," in the section "What Is a Publication?" as making public content that was previously private—has been around as long as humans have communicated. And from the first orators to the latest e-zines, the requirements for success in its pursuit haven't changed. A successful publication knows why it exists, speaks to a particular kind of person, is structured so that its content is easily assessed and accessed, and is laid out in a way that is compelling. A professional publisher knows these truths intuitively. Unfortunately, in the electronic publications world, you find few professional publishers and, thus, a definite lack of publications that follow these truths or even know that they're important.

Web sites in particular have blossomed—not as an outgrowth of established publication organizations but—as a groundswell of individuals and small groups with a penchant for the technology and some amount of content that they feel is important to share. Unsurprisingly, they've followed little of the field's established wisdom in their pursuit. I'm certainly not saying, however, that the Web should have developed directly out of the publishing industry. The way that the Web developed was just right. The massive parallel development of Web sites was just what this new medium needed to reach critical mass. Now that the Web is at critical mass and the dust is beginning to settle, you need to take the time to realize that what has been true for publications in the past is still true for Web sites.

Content collection

Nothing causes an organization to get efficient about its processes quite like a deadline. Moreover, nothing else equals a continuing string of deadlines to really hone an organization's processes. The periodicals publishing industry has just such deadlines. Consider a newspaper organization that produces a large book's worth of content each day. Streamlining every part of its process, from content creation through final publication, is more than essential for a newspaper. I was involved in the launching of one of the first online news organizations—MSNBC. I was quite impressed to see how it (and we) got better and better over time at putting out the day's news on time. In the end, we developed a hybrid between standard news processes and new electronic processes that enabled us to move a piece of content through a sophisticated series of tasks quickly and reliably.

Out of this environment of continual deadlines and large team efforts comes much of what I describe as part of the *collection process*, including the concepts of workflow, aggregation, and acquisition. I'm lifting my entire concept of syndication intact from the publishing world.

Abstracting the process

Although some publishers deal with separate content components that are simultaneously going into a variety of publications, most don't. Instead, the usual publishing process is a direct connection between collection and publication. It proceeds loosely as follows:

1. The publisher decides what sort of publication to create (daily journal, book, leaflet, magazine, and so on).
2. The publisher finds authors who are experts in the chosen medium (journalists, book writers, marketing writers, and so on).
3. The publisher creates a style guide that constrains the way authors create content by tone, word usage, length, and audience.
4. The authors create content that they target to fill a particular spot in a particular version of the publication.
5. Editors, formatters, and/or layout staff merge the authors' content into the publication one piece at a time until the publication is complete.
6. Post-production staffers do the final formatting and conversion to enable mass production of the publication.

The assumption behind this process is that you're creating a single target publication. The assumption behind a content management system (CMS), on the other hand, is that you're creating a number of simultaneous publications that you must fill with the same content or that the same publication changes so frequently (for example, every time that someone accesses it) that a direct connection between collection and publication isn't feasible. Moreover, the promise of a CMS is that it can harvest and organize the information and functionality not only for the publications that you now create, but also for any of a range of publications that you may need to create in the future. The general CMS-based process proceeds loosely as follows:

1. Those running the organization decide what value they have in information and functionality. This information and functionality makes up the content that the organization must manage.
2. They decide what sorts of people to deliver this content to and what parts of the content to target to which audience.
3. They decide what sorts of publications can best reach these people.
4. They install systems to produce, store, and manage the content effectively and to retrieve the content and automatically form it into the right publications at the right frequencies.
5. Over time, they collect feedback on the effectiveness of their publications and adjust the system as necessary.
6. Over time, they add new kinds of content and audiences to the system, which affects the types and composition of the publications.

The first publishing process is direct and manual. The second publishing process is indirect and automated. The first process begins with the publication. The second begins with the audiences and content. The first process is particular to the publishing industry and small groups within organizations. The second process borrows from the publishing industry and is useful to any organization. The first process describes how publications have already been produced. The second process describes how they're going to be produced.

The Principles of Document Management

According to Merriam-Webster OnLine (at `www.m-w.com/home.htm`) computer files are defined as a "complete collection of data (as text or a program) treated by a computer as a unit especially for purposes of input and output."

Files, of course, were folders full of papers in file cabinets long before computers were a gleam in Alan Turing's eye. The metaphor carries nicely into the computer world. Computer files are folders that are full of bits of data on a file system. And anyone who's ever used a real file cabinet knows the following:

- Determining into which file a particular piece of information goes is often a very arbitrary decision.
- The files serve simply as a system for getting to the particular information inside them.

In the computer world, as in the "real" world, files are convenient objects of manipulation. In fact, much of what a computer operating system does is manipulate files. The operating system enables the user to create, open, edit, and delete files.

Note As foundational as files are to the operation of a modern computer, nothing about them is necessary. They're just an arbitrary wrapper around a set of bits that enables the operating system to treat these bits as a unit. Still, in computers, everything comes from files and to files it all must return if you expect to see it after the next crash.

Given this definition of files, how do they relate to the concept of content and content management I'm developing?

Files contain content

First, files are containers for information. Just as content components do, files house information in a useful way. Just as content components do, files wrap information in a set of metadata that you can use to store and retrieve them. Although people often use files as content components, in general, they're inadequate for that purpose for the following reasons:

- **The metadata that the files contain is too general** and is the same regardless of the information that you store in the file. You understand this concept if you've ever tried to really organize your computer's file system. You can get only so far trying to organize files based on name, edit date, file size, and file format. Soon you realize that that you also want to use other kids of metadata to organize your files and that different kinds of files need different types of metadata for effective management.
- **The file's metadata applies to the entire file.** Thus, files can contain only one component. If you try to put two components in the same file, you lose the capability to use the file's metadata to identify individual components.

These two concerns are not new to content. Computer scientists realized long ago that the basic function of the file is to provide a standard way to store any data and that applications, not the operating system, are what need to access the data. Thus, they invented *file formats*. File formats are structures inside the standard file wrapper that enable you to add whatever metadata and other structure you want. So, although you *can* treat files as components, the better course is to let files serve as the gateway to a file format that can better represent the variety of components that you create.

Cross-Reference

Look in Chapter 2, "Content Has Format," in the section "Binary Format: Storing Information," for a further discussion of file formats that I there call *binary formats*.

Databases enable you to apply different metadata to each information type. In addition, they can store vast numbers of components in a single file (or a small cluster of files). Databases are simply a specific file format that enables a database application to store and retrieve information. Similarly, an XML file is a file that enables you to store and retrieve numerous components from within a single file.

Files store BLOBs

Databases have only recently been given the capability to store large amounts of data in a single field. With the digitization of pictures, sound, and motion, you can now assume that a single field in a database can contain many megabytes. The term *Binary Large Object* (*BLOB*) was invented to describe the collection of binary data that you can store in these large fields. Databases, however, are far from optimum for storing this sort of data. Thus, standard procedure today is still to store BLOBs in files. Instead of putting an image directly in a database, for example, the preferred method is to put metadata *about* the image (such as its name and size) in the database and keep the image in a separate file. Programmers have solved many of their problems in storing BLOBs in databases, but the penchant for storing media in files remains. Going hand and hand with the tradition of using files (which dies hard) is the fact that accessing media through the file system is often easier than doing so through a database interface.

XML was invented with BLOBs in mind, so XML files have little problem storing large binary data sets. Still, because most software for authoring and editing BLOBs (image, sound, and video editors) expects to work with files, storing BLOBs in separate files is still the most convenient way to work.

Publications are files

The end purpose of a CMS is to create publications. All publications are some sort of file. A Web site is a set of HTML files; a print publication is one or more files in a desktop-publishing format; and a syndicated publication may exist in the form of an XML file, in a database, or some other structured data stream. So the end purpose of a CMS is to create files of various formats.

In a way, this system is incredibly paradoxical. You go through a lot of work in creating your CMS, bringing information out of restrictive proprietary formats and into an open workable repository, only to push that information back out into a set of restrictive proprietary formats.

Computer scientists invented the file as a generalized container for any sort of data format that *you* care to invent. The result is that programmers have invented different formats for each software product. Their formats perform well under the restricted environments of their particular programs, but today these restrictive environments are butting up against each other as organizations need to coalesce what were previously many efforts into one.

A CMS can help bridge the gap between today, when content creation and publishing occurs in a wide variety of formats, similar in nature to a cybernetic "Tower of Babel," and the (maybe distant) future, when a single master file format can serve for all your content creation and publishing needs.

XML Is the Beginning and Not the End

Just the other day, I sat with a client who wanted something quite simple and reasonable. He was unhappy with the way that his organization continually needed to work around the peculiarities of the electronic and print publication systems that they'd purchased. He wanted to put his entire print and electronic publication process into XML and be done with all this restrictive proprietary software. As we worked through his options, we arrived at the following two conclusions:

- In the near future of computing, you're unlikely to see the print and electronic worlds converge enough that you can use something such as XML for a simple, coherent solution.
- Even if all publication systems can magically agree on and implement a standard such as XML, each publication system has very different needs. The divergence of needs of each publication type always requires very different systems for each type. So even under the best of circumstances, you can't create a single, simple system.

Document is an empty word

The word *document* is not a very precisely defined term. According to Merriam-Webster OnLine (`www.m-w.com/dictionary.htm`), the prevalent meaning of *document* is "an original or official paper relied on as the basis, proof, or support of something." If you search back through your old movie memories, you find any number of scenes in which the border guards say to the heroes trying to escape, "Show me your documents!" Clearly, however, if you're talking about electronic information, this definition is of no use. Computer documents are rarely original or official, and they're never paper.

Most computer references I checked don't even list the term *document*. The Microsoft Computer Dictionary defines a document in the following way:

> "Any self-contained piece of work created with an application and, if saved on disk, given a unique filename by which it can be retrieved. Documents are generally thought of as word-processed materials only. To a computer, however, data is nothing more than a collection of characters, so a spreadsheet or a graphic is as much a document as is a letter or report."

The definition boils documents down to word processing files and other common file formats (despite the fact that spreadsheets and graphics don't consist of characters). On the other hand, the definition clearly states that a document is any file. So why use the term *document* at all? For files, you can say exactly how they behave and what they contain; for documents, you can't. As widespread as this word is, I believe that it has little place as any sort of official term. I intend to stick to the word *file* instead.

Document management versus content management systems

Despite the vagueness of the word *document*, the industry of document management is quite well defined and has enjoyed considerable success in the 1980's and 1990's. Document management systems seek to organize and make files accessible. You can think of them as the file system that you wish your computer had. The better document management systems have the following parts:

- **File storage:** The system knows the physical location of each file that it tracks but doesn't require the end user to know that location.
- **File categorization:** You can assign file types and groups based on the criteria that you choose.
- **Metadata services:** You can attach any kind of extra data to a file (such as owner, status, create date, and so on) based on its type.

- **Collaboration services:** You can check files in and out of the system and jointly edit them.
- **Workflow services:** You can route files from worker to worker in an organized and standard way.
- **Versioning services:** You save a historical series of files and can retrieve them as you need them.
- **Access services:** End users can find files through tables of contents, indexes, and full-text searches.

The first thing to notice is that document-management systems have little to do with even the vague definition of document. Although you may often target systems toward word-processing and other office files, they have no restrictions at all about the type of files that you put in them. Thus, they're much more accurately called *file-management systems.*

The next thing to notice is that document-management systems share a lot with the concept of a CMS that I'm developing. Both systems seek to categorize information, apply metadata to it, organize its creation through workflow and collaboration, and give end users complete access to it. On the other hand, the two systems differ in the following significant ways:

- **Document-management systems deal with files,** while content management systems deal with content components. You may recall that, in the preceding section on files, I say that files don't make good containers for content. They're an extra container that's unnecessary for the storage of content components. In addition, they can hold only one component per file if their metadata is to prove at all useful for finding individual components.
- **Document-management systems were invented to manage files** that other applications create. These systems make no attempt to open any of the files under their control and work with what's inside the files. Content management systems, on the other hand, are directly concerned with creating content as well as managing it. In both systems, you control the creation of metadata for storage and access, but only in the CMS do you control the creation of the content as well.
- **Document-management systems are to provide access to the files** under its control. The purpose of a CMS is to create publications that are a combination of the components under its control. To create publications, content management systems need all the access power of document-management systems plus an extra capability to automatically construct a publication out of the components that it finds. Although some document-management systems can create "compound documents," which combine separate files into one, their capabilities fall far short of those necessary to create a solid and varied set of publications.

In a way, you can say that document-management systems are content management systems in which each content component is a file and publications typically contain only a single component. For this reason, I fully expected the document-management industry to easily capture the content management market; to date, however, it hasn't. Rather, a new industry with new companies has grown into the content management space. Given the document-management industry's depth of experience in metadata and workflow and its primary position in many of the world's largest organizations, adding the capabilities that I mention in this section should have proven relatively easy. Perhaps the document-management industry's success has created enough inertia to prevent it from taking even these relatively small steps into the future.

Whether the document-management industry comes to dominate the content management industry in the future, much of what's central to content management comes directly from document management.

The Limits of IT Groups

Every organization of any size has a department that takes care of its computers. This department is generally known as the *Information Technology* (*IT*), *Information Systems* (*IS*), or *Management Information Systems* (*MIS*) department. As the old motto of Microsoft's IT group goes, they "keep the train running." Until recently, every computer decision, from the official word processor to the obscure transport protocol of the company's network, flowed from this group. Before the Web, if you wanted to know anything about computers or how to use them in your organization, you'd ask someone from this department. Typically, an organization's IT group installs and maintains the organization's data systems, runs the network, and supports desktop applications. Some IT groups also develop software around the organization's data systems.

Today, many organizations avoid their IT groups like the plague in the areas of the Web and electronic publication systems. Why? I offer the following two separate but related answers:

- IT departments have avoided the Web.
- The Web begins where IT groups leave off.

The following sections elaborate on these answers.

IT departments have avoided the Web

IT departments have avoided the Web because it's been an unstable and out-of-control technology. A client of mine is a major software company with one of the world's biggest Web presences (measured in pages and throughput). Although (or maybe because) this company invented much of the software that runs the Internet, its IT group approaches the task of keeping the Internet running with extreme caution. The IT group often views the Internet effort as more of a train wreck than a train ride. I certainly can't blame them for this attitude. Especially in the early days, the company's site was growing by an order of magnitude each quarter. The staff's ability to create wild and wacky new Web stuff far exceeded this IT group's ability to certify that the next cool thing didn't bring the whole bubble-gum and rubber-band system to its knees. To get even a semblance of control over this explosion, the group created a set of super-strict guidelines for what people could and couldn't post to the Internet. Of course, these guidelines upset those who were champing at the bit to release all the great stuff that they'd created.

Note In the past few years, many IT groups have really woken up and realized that if they're not Internet savvy, they are shooting themselves in the foot. IT execs are increasingly being measured at least partially based on their company's success on the Web.

Look at this problem from another angle: How do you evaluate an IT group? On its vision and far-reaching strategy? No, that's for marketing and R&D. You evaluate IT on its ability to stay within the budget that management gives the department and on its ability to keep corporate information systems up and running. Cost overruns and downtime are death. No wonder IT departments avoid the flaky technology and costly downtime of the Internet.

Content begins where IT groups leave off

A friend of mine tells a wonderful story of the company CEO who sees his daughter playing on the Internet at home. He comes to work the next morning and tells his IT department head, "We need one of those things, too." The IT head says, "I'll get right on it, sir." He walks down the hall and calls on his top manager. "The CEO wants a Web presence," he says. "Can you do it?" "Sure," says the manager. "I'll get right on it." The manager forms a rapid response team from his top systems people. For a month, the team scours the world for the top Internet products. They spend three million dollars in product and services. The manager reports to the IT head, "We've created a world-class system." The head reports to the CEO, "Sir, for only three million, we've created a world-class system," and he scribbles the URL (that the company needed to purchase from a domain name "investor" who was expecting the call) on a piece of paper. The CEO comes proudly home and shows the paper to his daughter who's busy surfing to her latest finds on the Web. "Here," he says. "We have a Web site, too." His daughter pulls herself out of her Shockwave-based Web game, embedded in a moving, swirling page. She types the recently bought URL into her browser's Address text box. Up comes the great new site. It consists of a blank page with the following words in the upper left-hand corner:

```
Text Goes Here
```

Most IT groups just haven't the mindset or skill set to deal with content. Yet, as the story shows, their companies are calling on them to do just that. Dumb organizations expect that IT can just figure it out and do electronic publishing technologies. Smart organizations build alternative groups that pick up where the IT group leaves off. The smartest organizations recognize that the very nature of IT is changing and are building a new awareness and skill set in their IT groups. Fortunately, such a new department doesn't need to change its name. It needs only to fully embrace its current name. For a long time, computers were called on to deal only with data, and you could blithely imagine that data was information. Information Technology groups grew up in this environment. Today, as any CEO's daughter knows, information is a whole lot more than data but no less important to the organization.

I believe that, to survive, IT groups must take their part of the responsibility for Web sites and content management systems that create sites and other publications. What's their part of the responsibility? To install and maintain the data systems behind the CMS and electronic publications.

The Legacy of the Multimedia Industry

Before the Internet came the multimedia industry. This short-lived industry (which was at its height in the early 1990's) was where some of the key issues of electronic publication first were explored and confronted. Basically, multimedia groups and companies were interested in collecting interesting information and making it available on CD-ROM. Thus, people often refer to this industry as the CD-ROM industry, but CD-ROMs were merely the first medium with enough memory to distribute the content.

The essential aspect of the multimedia industry is right in its name. When I was young, I remember going to multimedia presentations. They consisted of slide shows with music playing and a person speaking. My sister Linda introduced me to multimedia art, where the artist uses acrylic- and oil-based paint and maybe embeds some objects in that paint. In both of these more archaic uses of multimedia, the concept is the same — that of bringing more than one medium of communication together to create a fuller communication experience.

Merriam-Webster OnLine defines media, in part, as "a channel or system of communication." Text, pictures, sound, and motion are all channels of communication. The multimedia industry brought the media of text, pictures, sound, and motion (video and animation) together with computing functionality to create the first electronic publications with real popular appeal. In the few short (unprofitable) years that the multimedia industry was in vogue, it managed to formulate much of what the Web became and is still becoming — in particular, the multimedia industry fostered the following breakthroughs:

- The multimedia industry pioneered the electronic publication.
- Multimedia developers where among the first (maybe just behind game developers) to feel the pinch of trying to simultaneously display text, pictures, and sound. In addition, multimedia teams needed to face the task of a massive collection and cataloging effort to prepare content for display. In response, they created some of the first end-to-end content management systems.

Electronic publications

Only a short time ago, you'd have been hard-pressed to find anything that you could call an electronic publication. Computer display technology has only recently become sophisticated enough to represent the color and pixel density that consumers require before using a publication. In addition, electronic publication tools are in their infancy, and people are only now really beginning to debate the standards of "online" publications. Electronic publications began in earnest only with the advent of the graphical user interface (GUI). While some software engineers labored to create the visual standards to guide all future programs, others began to use the enhanced capabilities of the GUI operating systems to present text in multiple fonts and to incorporate images, sound, and, later, motion.

The multimedia industry, which arose to take advantage of the vast new possibilities of electronic publications, produced everything from history to science fiction on CD-ROM. Along the way, it set much of what became standard in electronic publication. Many basic digital-design principles (the icon, VCR controls, and button bars, for example) first appeared on a computer screen in one of the thousands of custom user interfaces that late-night designers and programmers hacked together. Artists, creative writers, and media experts of every variety were co-opted and merged into the world of software development. Although the Web may have put a wet blanket on the multimedia CD-ROM, it simply fanned the flames of the growing move to digitize and display information.

End-to-end content management

The multimedia industry swept hundreds of thousands of content creators online. Along with them, they brought what they knew of publications and a whole lot of content. Despite the chaos of the multimedia-development process, these creative types crafted their own semimanual systems for moving content from place to place and ensuring that it arrived where it belonged on the CD-ROM. Having built a number of these systems myself, I can say with assurance that the process wasn't pretty. They did, however, make an impossible process merely hard.

Authoring systems, as they were known, provided creation, tagging-conversion, and compilation tools. Storage was generally in a proprietary binary format. Few used commercial databases to store content. Specialized proprietary browsers (not that the term *browser* actually existed at the time) were created that could lay out and display content. Over time, many of the companies that produced high-content, multiple-version CD-ROMs (such as those containing encyclopedias, periodicals, and corporate directories) developed systems that displayed most of the features of a full CMS.

Somewhere about midway through the multimedia epoch, the term *asset* crept into the vernacular. To my mind, this word did more than any other to popularize the component way of thinking about content. Merriam-Webster OnLine says that an *asset* is "an item of value owned." The two key words here are *item* and *value*. Rather than a continuous stream of information defining a publication, the multimedia world came to define its systems in terms of the individual items they managed. Each item has its own value (sometimes literally, as in the case of photographs that you may purchase), and you track and manage each separately. These items, which you store as separate components, were brought together in the creation of a particular multimedia CD-ROM, and you could mix and match them as necessary on the published page.

Those multimedia teams that forged decent content management systems had little trouble moving to another publication type — the Web — after it swept away CD-ROMs and the multimedia industry.

The Challenges of Documentation Projects

At about the same time as multimedia was working its way to the top of the public mind, software manufacturers and large industrial companies were confronting problems that were just as important to the nascent field of content management. In software companies, the need to create online Help to supplement existing print documentation gave a boost to the notions of multiple publications from the same content base. In large industrial companies, the sheer volume of documentation about their products, as well as their enormous parts catalogs, spurred the creation of the first real, structured text-management systems.

Huge information bases

Because I live in Seattle and am in the electronic publication business, about once a year I hear the story of how the documentation for a Boeing 747 requires a 747 to transport it from place to place. As the story goes, Boeing reduced the load to the cost of an airmail stamp (or two) by putting all that documentation on CD-ROM. What Boeing and many others like it did to create these CD-ROMs has a lot of bearing on the current world of content management.

The main problem that these industrial giants needed to tackle was organizing the enormous amount of information they'd created. Older print publication systems had no capability to know what the information was "about" — only how to format it on the printed page. To get organized, what Boeing and others needed was a way to capture and store information — not by how it looked but by what it meant. The answer they found was the *Standard Generalized Markup Language* (*SGML*).

Cross-Reference

In Chapter 28, "What Are Content Markup Languages?" I talk a lot more about SGML. For now, simply understand that SGML brings databaselike rigor to the creation and tracking of content.

At the same time, a group of software companies provided tools and management systems around SGML. This set of vendors and customers together pioneered the use of strictly structured, text-based management systems to author, manage, and publish huge volumes of rapidly changing information.

Many of the vendors who were most popular with the industrialists have since made the leap to XML and are now serving the wider content management industry. Unlike those companies in the document-management industry, they've branched out as their potential client base broadened. As is true of those still in the document-management industry, however, their original client base may prove their worst enemy. The closed, arcane world of SGML and its back-room clientele has weighed these companies down and made them move much more slowly than their newer competition toward the future uses of structured text in electronic publishing.

Simultaneous publications

With the advent of Windows 3.0, the Microsoft side of the personal computer world turned many corners. In addition to the main events of its graphical user interface (GUI) and API-based programming, Windows Help launched. In the previous DOS world, Help meant a load of unformatted text that scrolled off the bottom off the screen and was all but unreadable. By using a tool that became known as WinHelp, the Help author could create well-formatted, well-indexed hypertext systems that he could tie directly to the application functions they described. The system worked well enough that it could contain large amounts of documentation. It was also one of the big new standards of Windows programming. In other words, to be a good Windows application, a program needed to come with documentation created in WinHelp.

A few software companies used the new system as a pretense to do away with the most expensive part in their shrink-wrapped box—the book. Most simply added a requirement for Help to the list of the deliverables they expected from their User Education departments. This new requirement fundamentally changed the way these departments worked. It mandated that a group of people who'd in the past defined themselves simply as book writers now become proficient in the ways of electronic publication. To most, this new requirement meant that they needed to translate their books into Help systems. To others, it caused a fundamental rethinking of what publishing means and what content is in the absence of a particular publication to which it's targeted. The result of this rethinking was a spate of single-source publishing systems. These systems took a single base of content and alternatively produced either a Windows Help system or a print publication.

My very first content management systems were of this variety. They used Microsoft Word for Windows 1.0 and its macro language to create a simple repository for content that contained enough information to produce either a book or a Help system. The systems contained a rudimentary authoring system, which aided in tagging new content, a publication-neutral storage format, and routines for producing either of the two publications.

With two publications of very different format and structure to produce, many of the concepts of content management were necessary to keep the creation and later publication organized and efficient. The content needed, for example, to remain separate from the final format. Structural metadata needed to be added to the content so that it could be automatically chunked into named Help topics or book sections.

Unfortunately, the efforts of the documentation team were directed only toward documentation and weren't always applied to the wider questions of Web publishing. The success of Windows Help also led to a group of people who considered themselves Help authors rather than electronic publishers. Thus, much of the talent and innovation of these documentation groups wasn't tapped. Still, people with a background in electronic documentation are a great source of talent, especially for the collection process in a CMS.

As the tide of enterprise content management rises, however, you can expect a lot of the knowledge of the documentation groups to spill out into the wider organization.

The Traditions of Library and Information Science

Librarians have felt the pinch of organizing information bases for as long as libraries have existed. Librarians organize and provide access to large volumes of information. Whether they oversee books, periodicals, images, sounds, or moving pictures, librarians create and maintain collections, catalogs, and content. The parallels to the concepts of content management should be obvious. Just as does a library, a CMS must catalog a collection of content.

Information science, which grew out of cognitive science and computing and is now in the process of merging with library science, has brought depth and substance to our understanding of how people find and consume information. Information scientists study and create the theory of electronic information categorization, retrieval, and use. They created the theory on which I base much of my theory of content management.

In addition, the *management* in content management I base quite clearly on the principles and practices of library and information science.

Categorization and information seeking

Librarians invented and commonly use the hierarchies, indexes, and cross references that a CMS must also employ.

Cross-Reference See Chapter 26, "Designing Content Access Structures," for more information about these terms.

Access structures are the tools of the trade of librarians. Each library's collection is organized into a taxonomy that's no more than a hierarchy. Librarians use all kinds of indexes to map the keywords of a collection to each other and to the content in the collection. Cross-reference strategies lead the visitor from one category of book to the next. In addition, the reference librarian is expert at turning a vague question into a firm strategy for access.

In both their conceptual categorization of information and very practical strategies for finding the right content, librarians hold the keys to much of what the metator needs to effectively catalog content in a CMS. Although many of the world's finest librarians remain sequestered in stacks, many others are joining the ranks of the digerati and applying their incredibly valuable skills to digital information. Print librarians are always going to be needed, but the future holds a much larger need for the same skills in a broader (and more highly valued) context.

For years, I've been fond of saying that librarians are the gurus of the future. When I said so in the late 1980s, I got nothing but blank stares. At the turn of the century, I still get some confused looks, but the insight is slowly dawning that those who can make information accessible — and especially those who can find the content needle in the haystacks of electronic content — are eventually to reign among the most valued professionals in Western society.

The structure of information

Information scientists study the gross and minute structure of information. They provide a solid theoretical basis behind the sometimes shaky process of content management and electronic publication. Among others, information scientists confront the following central issues:

- **How do you look for information?** If you need information, how do you verbalize that need and then how do you go about searching for it.
- **How can you retrieve information?** What are the best ways to store information so that you can most easily find it again? How do you learn from the kinds of searches that people have previously conducted? What are the fastest and most efficient indexing processes so that you can most effectively retrieve information?
- **What is relevance?** How do you measure the value of information? How can you importance-rank the information that you find in response to a user question? How do you resolve ambiguities in peoples' questions?

Information science overlaps considerably with cognitive science, which seeks to answer questions such as "What is context?" and "How do people perceive and understand?" As content management moves from art to science, these are the questions that you need to address. Information science also overlaps with linguistics and artificial intelligence. Information scientists are in a unique position to apply the concepts of human language processing and computer language processing to the content management issues of automatic tagging and advanced personalization, based on intelligent rules.

Note If anyone eventually creates some future generation of content computers that can transcend the limitation of data computers, the idea may well start in the mind of an information scientist.

Unfortunately, the skills and tools that information scientists possess all too often fail to make their way out of the university setting. People possessing these skills, however, are prime candidates for structuring an organization's information and deciding how best to make it accessible.

The Technology of Software Development

The software industry powers all of content management and provides the most basic principles of electronic publication. Physically, a CMS is a collection of computer hardware and software. That the work of a software developer lies behind every CMS is, therefore, no surprise. In particular, the software development industry's created all the technologies currently in use to collect, manage, and publish content. In addition, software developers (and architects) were the first to confront the issue of the computer-human interface. Their work led to all our standards for representing functionality in electronic publications.

Collection technologies

Long before the age of the electronic publications, programmers grew tired of typing. To help themselves, they invented powerful text-manipulation programs. Tools with names such as GREP (the UNIX tool to Globally search for Regular Expressions), SED (the Stream Editor), AWK (created by Alfred Aho, Peter Weinberger, and Brian Kernighan), and finally Perl were created to enable programmers to reprocess code, efficiently produce reports from organizational data programs, and convert data from one system to another. Perl, as you may know, became the basis for most of the early automation systems behind Web sites. Perl and many other tools have also served as the basis for the kind of text manipulation that you need in a CMS collection system. Creative programmers have also used the text-manipulation capabilities that are native to most programming languages (C, Java, BASIC, and so on) over and over to make effective content-processing systems. These custom systems include text parsing, automatic tagging, form-based input, content preview, and custom workflow management.

Much of this early work is still in place, moving content. New work here continues as well, but it's now becoming more focused on filling the gaps in the now commercially available CMS collection systems rather than creating new systems from scratch.

Management technologies

Databases are at the center of both the programming world and the CMS world. Database software and database programmers are as much a part of a CMS implementation as they are for any other major application. In addition, the administrative functions of a CMS are mostly the same as they are in any enterprise database system. In any application that serves multiple users (for example, the tasks of defining and changing user records and giving them access permission) the data administration tasks are basically the same. As the capability of databases to store, index, and retrieve increases, CMS software is sure to change as well to take advantage of the new possibilities.

Most CMS products work with one or more of the major commercial database products. From a software developer's perspective, the content management system is nothing more than a complex user interface to a standard database and file system.

Publishing technologies

Programmers created (and continue to create) all the electronic delivery systems that a CMS uses.

Today, the major publishing technology is the *template*. Templates are programs. These programs decide what content you need on the page, make queries to the repository to retrieve content, and format the content for correct placement and display in the publication. The first Web publication templating system that I ever saw was Microsoft's IDC/HDX system, which shipped with an early version of Internet Information Server (IIS). This system, which Microsoft's Active Server Pages (ASP) system quickly superceded, was invented by programmers to plug the newly discovered gap between Web pages and databases. The system was little more than a set of database queries embedded in an HTML page. From that sort of humble beginning, programmers evolved a world of technologies that go way beyond queries to now provide any sort of data retrieval and formatting that you may want. The latest templating tools that I've seen feature a graphical user interface and enable you to drag and drop elements onto a Web page. Some elements are simply static text and graphics, while others are complex programming objects that perform a tremendous amount of content access and processing.

Templates perform content layout and formatting using many of the same text manipulation programs that collection systems use. They also employ specialized routines for layout, personalization, and building navigation.

Needless to say, programmers created the Web browser, which is *the* vehicle today for electronic publications.

The functionality in electronic publications

Web sites have developed a language of their own. Standard display elements such as the link, the banner, and the icon didn't come from software developers but from designers. Still, wherever you find functionality on the Web, you also find the same small set of standard controls as appear in any application. Buttons, lists, check boxes, and menus are as common on the Web as in any standard application. And in any publication that includes functionality, a programmer is lurking somewhere in the background, writing code to hook the buttons and other controls to databases and other back-end programs.

Software developers were the first to confront the issue of how to represent functionality. In response, they borrowed from the mechanical age of functionality (the button, for one example) and invented some new methods (the drop-down menu, for another example). Over time, they created and perfected a small but very versatile set of controls that successfully direct attention and convey meaning in the impoverished environment of the computer screen.

You may never have stopped to think about it, but the following small set of widgets (creatively combined and artfully extended) provides the interface for almost all computer functionality:

- **Text box:** This control enables you to type small snippets of text. Three variations of the textbox are shown in Figure 9-1: the *Password* control, which masks your input; the *File Field* control, which accepts only valid paths and filenames; and the *Text Area* control, which is simply a large textbox.
- **Check box:** This control enables you to specify "true" or "false" to the computer.
- **Radio Button:** The name of this control comes from that of the little push buttons on a car radio that you use to select preset radio stations. This control enables you to answer multiple-choice questions for the computer where the responses are always mutually exclusive.
- **List box:** This control enables you to display lists of choices from which you can select an item. Two variations appear in Figure 9-1: the *Dropdown*, which opens up after you click it, and the plain *Listbox*, which is always open.
- **Button:** You click this control to make something specific happen. The *Submit* and *Reset buttons*, as shown in Figure 9-1, are just buttons to which HTML browsers assign particular functions.
- **Menu:** The menu is a standard control that doesn't appear in the illustration. Menus provide a hierarchy of commands. They are, in effect, buttons in an outline.

These few controls have had amazing durability. When these controls first arrived on the consumer's screen, I'd never have thought that they'd remain so many years later, as the basis of all functionality in both electronic publications and standard applications. The success of the standard controls was in part because they were promoted so well by the major software firms. I believe that their capability to continue, however, is based as well on the forethought and hard labor of the programmers who invented them.

A measure of the success of these controls is how often they now appear in print and television advertisements that seek a high-tech appeal. These controls have literally become the icons of the electronic age.

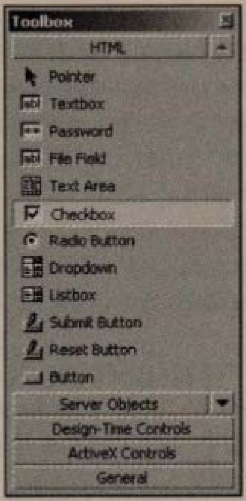


Figure 9-1: The Toolbox window from Microsoft's Visual InterDev shows the basic controls available on a Web page.

The Premises of Marketing

The central premise of content management is to collect and deliver value to a set of known audiences. This premise is familiar to anyone involved in marketing, corporate communications, or public relations. They're intimately involved in defining value in the organization, understanding the organization's audiences, and ensuring that the right audience gets the right value. Manipulation notwithstanding, marketing people know how to define and deliver value. From this realm, I've drawn the following concepts:

- ✦ **Goal setting:** My concepts of project definition and goal setting derive from standard business planning as well as from the marketing realm. To become successful, any organization must begin by knowing exactly what they're trying to accomplish.
- ✦ **Value determination:** As any good marketer knows, you begin by understanding what you have that's of value to others. My notion of value propositions and value-based content chunking follow this same reasoning.
- ✦ **Messaging:** My ideas of creating the messages that a publication must deliver derive directly from the message-setting exercises used to develop marketing communications.
- ✦ **Audience analysis:** The kinds of information that you need to gather about an audience, the conclusions that you can draw from this information, and the techniques of audience segmentation come directly from marketing.

In the high-tech world, among the mostly liberal masses, the word *marketing* is often followed by some less than laudatory moniker. The derision isn't wholly undeserved. One lesson that the marketing family of disciples can teach the rest of its high-tech cousins, however, is how to quantitatively know your audiences. Moreover, unlike many other disciplines that give mostly lip service to "user-centered" design, marketers know that success comes only if the audience is the central concern of your entire system.

Summary

Content management began when the first scribe needed to keep track of and reuse the scrolls he'd amassed. Since then, any number of disciplines have arisen that have some wisdom or practice to offer the new discipline of content management. In this chapter, I detail the contribution of the following fields:

- The publishing industry provides much of the knowledge behind content collection but must now adopt a multiple publication paradigm.
- Files and the computer's file system provide an inadequate system for organizing content. The file isn't the best container for content components.
- Document-management systems contain a lot of what you need in a CMS (such as metadata and workflow) but fall short because of their reliance on files rather than on content components.
- The information management discipline provides a good source of infrastructure and can usually implement much of a CMS, but it's not designed to handle most of the collection and publication tasks in a CMS.
- Multimedia, although short-lived, provided much of what I build my work on. In its focus on types of media other than text and its adherence to managing units of content separately from their final published form, the multimedia industry anticipated many of the problems and solutions now in use by content management systems to create Web sites.
- Computer and corporate documentation systems provided the first real tests of our capability to manage large-scale content bases and produce multiple publications from them.
- Library and information-science professionals provided a firm footing on which to base the content management concepts of metadata, storage, categories, and information retrieval.
- The software development industry, obviously, provides all the software that you need for content management and pioneered the distribution of electronic functionality that all electronic publications now use.
- The marketing and communications disciplines provide much of what I base my concepts of audience and personalization on.

In the next chapter, I go from the roots to the branches of content management. I describe the places where content management can prove — and already has proven — of use to organizations.

✦ ✦ ✦

CHAPTER 10

The Branches of Content Management

✦ ✦ ✦ ✦

In This Chapter

How content management enables personalization

How a CMS supports the creation of advanced Web sites

How a CMS supports the creation of multiple publications

How e-commerce is enabled by content management

How content management underlies knowledge management

How online communities are built on content management

✦ ✦ ✦ ✦

One of today's content management companies is fond of saying that content management is the operating system for e-business. Hyperbole aside, this statement does contain some truth. An operating system is the infrastructure that lies below applications. It provides a common set of services that applications draw on. Similarly, content management can underlie many of the Web technologies and applications that constitute e-business. In this chapter, I discuss how content management can underlie the following key business applications:

- **Personalization:** To be effective, delivering personalized content requires a content management system (CMS) behind it to make content accessible and targetable.
- **Advanced Web sites:** Most of the technology to create and maintain large and complex Web sites is based on content management.
- **Multiple publications:** To effectively share an information base across multiple publications, you need the infrastructure and publication systems that a CMS provides.
- **E-commerce:** You can use content management systems to directly manage the catalogs behind commerce sites. In addition, a CMS can fully manage the presentation and personalization of e-commerce functionality.
- **Knowledge management:** If knowledge management is the collection, management, and distribution of what an organization "knows," a CMS can prove the best platform behind a knowledge-management system.

- **Online communities:** As are geographic communities, online communities are based on affiliation and common knowledge. A CMS can provide the infrastructure behind the community.

Every segment of the computer industry tends to put itself at the center of the universe and see the other segments as revolving around it. I don't see content management this way. It's not at the center of the computer universe but at the bottom of it. Just as an operating system provides basic services that any number of applications can draw on, so, too, a CMS provides the basic system of information capture and distribution that any other system can draw on to meet its particular needs.

Just as important, the CMS mindset is one that applies generally. A CMS requires you to get a solid understanding of your information sources, your audiences, and your information outlets. This exact knowledge can help you succeed whether you're creating Web sites, e-commerce ventures, community sites, or any of a range of other systems. In the end, all these systems have the same goal — get the right people the right information and functionality at the right time through the right channel.

Personalization

If you personalize a publication, you tailor its content and presentation to the particular individual who's viewing it. Although personalization has taken on an almost mystical air in the current Web era (an era lasting about six months), the basic concept is quite simple: Match content to audiences.

What is personalization?

Personalization is delivering the most appropriate content to a person inside a standard framework. Although personalization is a Web buzzword, the concept applies to any delivery channel (as the personalized address labels that I keep receiving in the mail clearly show). More specifically, personalization proceeds as follows:

- You collect data about the user.
- You match that data to content.
- You deliver that content within a standard context (for example, a Web page with standard branding imagery, banners, navigation, and so on).

Suppose, for example, that you want to produce a personalized automobile Web site. On the home page, you may ask what kind of car the user owns (*collecting user data*). Then you use the car that the user enters to select content for the next page (*matching content*). Some parts of the second page, such as your logo and company name, are always present and in the same positions (*delivering within a standard context*).

You want to create a dynamic dance between the data that you collect and the structuring of the content in the system. On the one hand, you want to structure your content according to the level of the data that you want to collect. (What's the point of finding out the age of the user if you're not going to tag the content in a way that enables you to deliver age-specific content?) On the other hand, if you can identify obvious discerning parts to your content, collecting the data that enables you to personalize based on that content structure makes sense. If a summary level and detail level are already distinguished in the structure of your content, for example, why not collect the data that you need to decide which level the users want?

You can use either of the following basic types of personalization:

- **Profile-based personalization:** If you can identify traits in the audience of your publications, and group those traits into profiles, you can deliver information based on a user's profile. On the content side, you must identify the components that are of most interest to members of that segment. You can use this method to target entire segments of your users or to create a one-to-one publication, where each user sees a publication personalized to her particular profile.
- **Behavior-based personalization:** You can potentially infer a great deal about what someone wants to see, basing it on what they previously chose to see. If you can identify visited components, associate them with desires, and then link them to other components that meet the assumed desires, you can build behavior-based personalization into your dynamic publications. For static Web sites and other publications, behavior-based personalization isn't possible, because you have no way to monitor behavior within the publication.
- **Campaign-based Personalization:** A campaign is a planned content "push" to users who meet certain criteria. For example, you might plan to push information about an upcoming lunar eclipse to users that you have identified as being interested in astronomy. Campaigns have defined start and finish dates and a defined audience.

Content management underlies personalization

I believe that personalization is so much a part of content management that it serves little purpose to tease the two very far apart. In this work, I consider personalization as simply the last piece of management that happens to content before you lay it into a publication. Every other process prior to that last act also serves to target content to audiences. In fact, the entire purpose of a CMS is to deliver the right content to the right audience. So you may say that the entire CMS is a personalization system. On the other hand, a specific, generally agreed-to part of the process is known as personalization: the rules and software that select particular content and navigation into a publication, basing that selection on some data that the system obtains about the person viewing it.

Cross-Reference You can find more information about personalization in Chapter 22, "Designing Publications."

In the following sections, I discuss how a CMS underlies the three steps in personalization.

Collecting data about the user

The structured delivery that a CMS provides enables you to very accurately associate the user information that you collect to its context. Instead of asking what kind of car a person owns, for example, your system can observe that the person went to the page for a Toyota. From that choice, you can assume that she's interested in Toyotas. That's a fine design, but for a CMS behind the scenes delivering structured content types to standardized pages, this is a difficult task. First, if you did not systematically produce the pages, you have little chance of knowing exactly what's on any particular page. Second, the user data that you collect is itself content for some system to manage.

Matching data to content

Suppose that you want to personalize your home electronics site—that is, you want to provide information that's most relevant to the people visiting your site. Your marketing research uncovers the following four basic visitor types:

- *Generation M* (for millennium), ages 6 to 20, whose interests lie in electronic games.
- *Generation X*, ages 21 to 35, whose interests lie in handheld audio devices, such as MP3 players.
- *Baby Boomers*, ages 36 to 55, whose interests lie in home stereo systems.
- *Empty nesters*, ages 56 and older, whose interests lie in videophones.

You want to deliver information about these product types to these demographic groups. In addition to figuring out what group a particular visitor belongs to, you also must locate the right type of electronics for that visitor's group. To do this personalization, you need many of the main features of a CMS, including the following:

- **Content components:** To deliver product information, you must differentiate it from other types of information. You can then find and deliver the product content components.

Cross-Reference You can find more information about content components in Chapter 23, "Designing Content Components."

- **A repository:** To effectively organize and find your product components, you must store them in a database or other repository system that can deliver them to your personalization program quickly.

Cross-Reference You can find more information about repositories in Chapter 32, "Building Management Systems."

- **Metadata:** To find the right product components, you must mark each product component with additional information about which demographic it applies to. Then, after you identify a visitor as a member of one of your target audiences, you can query your repository, basing your query on this audience metadata, and return the appropriate information.

Cross-Reference You can find more information about metadata in Chapter 20, "Working with Metadata."

Delivering personal content in a standard context

Regardless of the particular information that interests a user, certain publication elements needn't change. The name of your organization very likely remains the same regardless of the audience you're serving. Your brand and general layout of your publication also needn't vary. These elements remain the same—not because they can't change but rather because they usually don't need to. To stay cost effective as well as present a unified identity to all your audiences, you need to create a standard frame around personalized content.

Note I mean *frame* in a general sense, although on a Web site, you may actually use frame sets to create the standard frame.

In a CMS, publishing templates provide this sort of framework. They house the standard elements that don't change from page to page. They also host the computer code that decides who the user is and what particular content to include on the page inside the context that the standard elements create.

Advanced Web Sites

As I write this chapter, I'm en route to yet another client whose Web site is getting a bit out of control. To quote one of the documents that I received from this client:

> "Most of the attendees agreed that establishing a consistent publishing process is important to ensure both the quality and accuracy of Web content. Some said that, if their department had an established editorial process, they either didn't know what it was or it wasn't being followed. Although the content management system must allow for the different needs of all departments, establishing a consistent process was viewed to be important to a collaborative publishing effort.
>
> "Some of those in attendance cited a 'startup mentality' as a problem. Without an established publishing flow model, content creators are often forced to act as writer, editor, producer, and reviewer. Some of the consequences of this

> situation that were mentioned were a lack of accuracy, substandard quality, and a feeling of isolation among editors and writers within different departments of the company."

As in so many other organizations, these folks started with a few energetic technologists and writers who were enthralled by the new challenge of the Web. They attacked this exciting new venture with the entrepreneurial spirit that the phrase "startup mentality" captures. In a startup venture, the premium is on getting something together and releasing it as quickly as possible. This situation usually involves heroic efforts by a few very dedicated individuals who are willing to do whatever's necessary to keep the venture afloat. This startup mentality's driven the Web to its current heights at its current velocity, but it can last only so long. Eventually, staff members become burned out and unable to continue the pace. In addition, after innumerable crises that they could easily have averted with a modicum of planning, they become cynical and stop picking up the slack. Finally, and most important, after the effort is really running, the entrepreneurs leave for the next exciting venture, and managers, whose job it is to go from a startup to a running system, replace them.

Content management, as the name implies, is a manager-type activity. It is, in fact, antithetical to the startup mentality that's permeated Web activities so far. A CMS is a well-oiled machine; its purpose is to create a smooth and manageable process around the publication of a Web site (and other publications). Interestingly, the introduction of a CMS often accompanies an exodus of the types of individuals who thrive on the chaos of a startup technology.

Tip

If you're smart, you anticipate this situation and find other cutting-edge activities for these valuable adrenaline junkies to lead before they leave you for more stressful activities.

The reason that this company and most other organizations with large Web sites are pursuing content management is that the startup mentality that launched their sites is too expensive and too inefficient to keep their sites going.

In particular, their sites now display the following characteristics:

- A lot of content and types of content.
- A lot of change in their content and their designs.
- A wide and distributed contribution base.
- A lot of content sharing between pages.

A CMS provides an appropriate infrastructure for handling the challenges of an advanced site. In fact, most commercially available CMS products currently make this type of infrastructure their main goal.

A CMS is designed specifically to deal with content in bulk. The idea of content components is to categorize and organize your content so that you can manage it as classes and not as independent parts. Although you manage the content as classes, you can still deliver it as individual chunks. In a CMS, templates enable you to separate content from the design of particular publications. As publication design changes, therefore, you needn't modify content. Generally, you can accomplish a design change very quickly after a CMS is in place, because only a small number of templates are affected.

Most CMS products offer the capability to present easy-to-use forms that give remote and nontechnical authors a quick way to submit content. The forms guide the authors through the process of entering content in the way that the CMS expects them to structure it and provide a foolproof way of ensuring that they add the appropriate metadata to the content as necessary.

Finally, because you store content in a CMS as independent components and later form them into pages by using templates, sharing content between pages is simply a matter of calling the same component into both pages. Of course, you must create components and decide which pages get them, but at least you don't need to retype the same content in two different places — nor do you need to keep two different versions in synch if the content changes.

Multiple Publications

Today, the content management industry emphasizes the Web. In fact, many companies that now offer CMS products previously saw print as their main target. Many have now all but abandoned this older medium, pointing their products only at the Web. This situation is unlikely to last. As many of my clients tell me, they experience an enormous duplication of effort in their organizations for each different publication format. My typical publishing client, for example, produces a final book or magazine and then passes print files on to its Web team. That team pulls the final files apart and puts them back together as Web pages.

Clearly, a CMS, correctly implemented, can save a tremendous amount of effort and enable companies to create their print and electronic publications simultaneously. Of course, print publications aren't the only issue. Most organizations create an Internet site as well as an intranet site. They may also create extranet sites aimed at their partner organizations or their customers. In addition to the variety of Web sites that such companies can produce from the same content base, there is everything from Wireless Application Protocol (WAP) telephones to Personal Digital Assistants (PDAs), to television set-top computers. Organizations have a large and growing need to feed information into a wide variety of delivery channels. No organization can possibly create a separate staff, system, and support structure for each channel. Fortunately, with a CMS in place, they don't need to.

As an example of the sort of system that's still most common, consider a magazine publisher that wants to put its great magazine content on the Web as well as in print. Figure 10-1 shows the process such a company might use. Notice that the print publication is complete before the Web publication can begin. Notice, too, that the company must discard much of the good work of the authors because it's of no use to the print process.

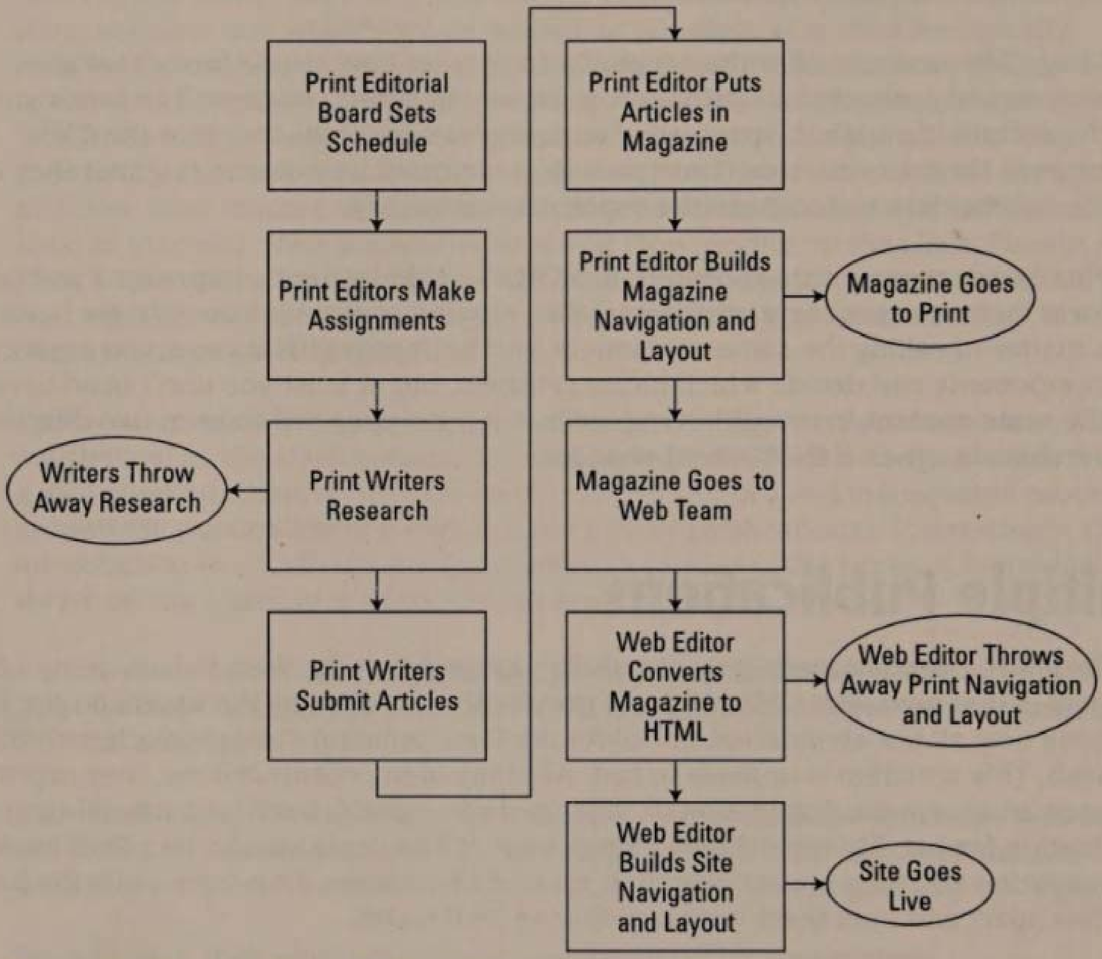

Print Editorial Board Sets Schedule
Print Editors Make Assignments
Print Writers Research
Writers Throw Away Research
Print Writers Submit Articles
Print Editor Puts Articles in Magazine
Print Editor Builds Magazine Navigation and Layout
Magazine Goes to Print
Magazine Goes to Web Team
Web Editor Converts Magazine to HTML
Web Editor Throws Away Print Navigation and Layout
Web Editor Builds Site Navigation and Layout
Site Goes Live

Figure 10-1: A typical pre-CMS publishing process

The editorial board of the magazine creates a publishing schedule for the year. For each issue, the board members identify the articles that they want to publish. The next level of editor assigns articles to writers, giving them a subject, angle, and a word count. The writer begins researching the article. She collects a tremendous amount of background information, performs extensive interviews, and writes an article that may run two or three times the word count that the editor assigns her. She puts all her research in a box, cuts out half the words from her article, and passes the result to the editor. The editor expects a certain number of words from the author, which she lays out in a desktop publishing tool to see how it fits in the magazine. She either adds to or, more commonly, takes out text to make it fit. She breaks the article into columns and divides it into the part that goes in the front of the magazine and the continuation that jumps to the back of the magazine. Other editors do the same until all the articles fit. The editors then build the table of contents and indexes for the magazine, make their final checks, and release the magazine files to the printer.

Only then does the Web process begin. The Web editors start by stripping out all the print layout and navigation. They convert the print files to a Web format and put all the split-up articles back together. Finally, they rebuild the navigation and indexing in HTML. All in all, a lot of time and redundant effort goes into bringing the same content to both print and the Web.

Note This process that I describe is a generalized version, and I don't intend it to represent the definitive word on what publishers do.

To simultaneously create multiple publications that share significant amounts of content, you must perform the following tasks:

- Segment the information that you want to publish into content chunks that are independent of any of the particular publication's needs but that you can make serve the purpose of any of them.
- Separate the publication design from the content so that each publication can have its own separate format and structure but still accommodate the content.
- Create rules for how to select the right content chunks for each publication, how to format them correctly, and how to lay them into the overall publication structure.

Clearly, this process is a job for a CMS. Using content chunking, workflow, publication templates, and publication rules not only makes multiple publications possible, but also makes them inevitable. The CMS provides the infrastructure to turn the old serial process of creating multiple publications into a much more efficient parallel process. You can, for example, streamline the publication process that I outline earlier in this section considerably by using the tools that are available in a CMS. Print and Web publications can occur simultaneously and can, therefore, take advantage of different parts of the authors' work, as shown in Figure 10-2.

No need to change the editorial board and assignments, which, for this organization, is where they create value. This situation is also true beyond a print publisher. Instead of altering the basic information-creation process, what organizations need most to do is to generalize the process so that it can feed the entire range of target publications. For the publisher, this generalization means that the writer submits all the content and not just finished articles. Although the print publication contains only the tip of the research iceberg, the site can contain all the content that the writer produces, including background information, links, interview transcripts, and anything else that the writer can turn up.

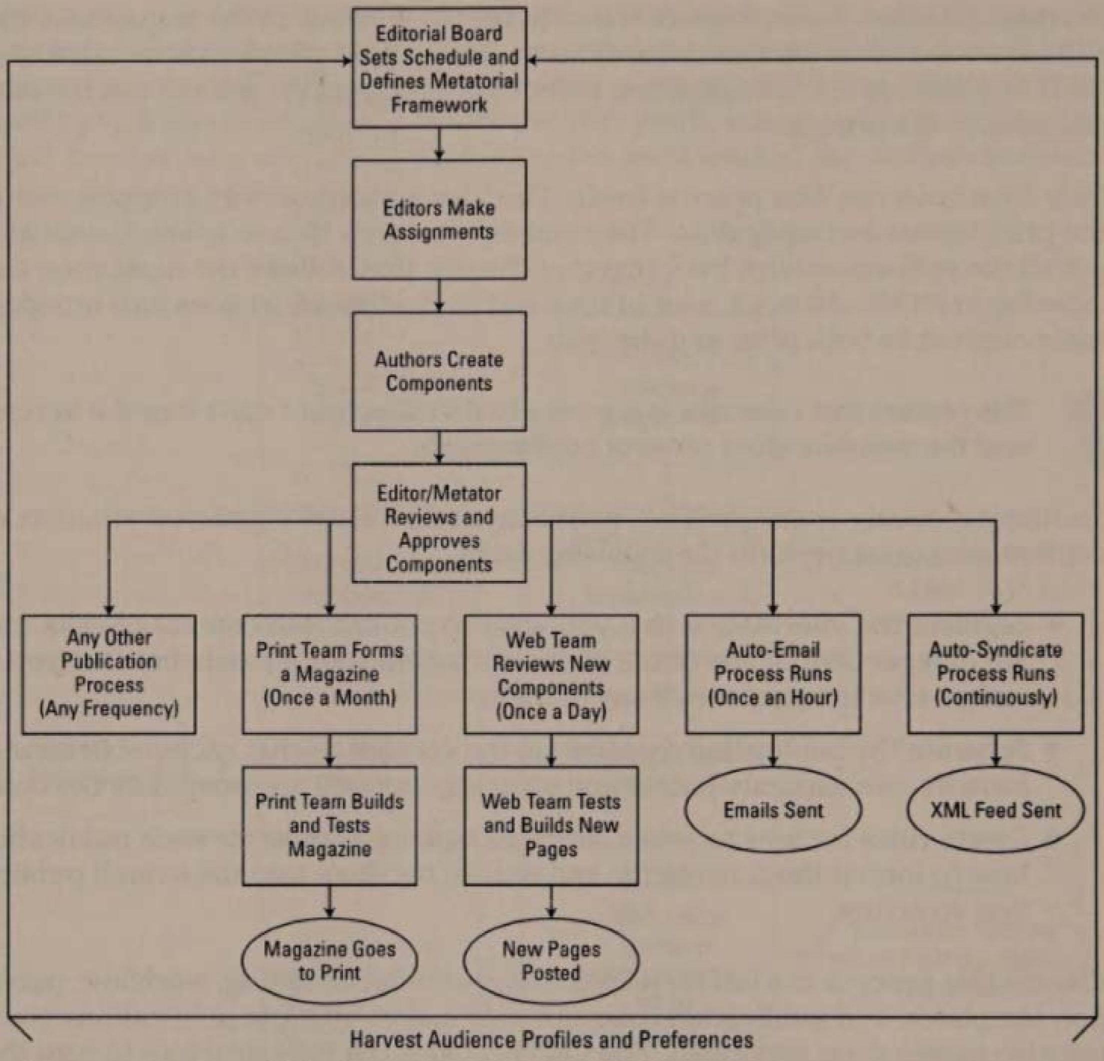

Editorial Board Sets Schedule and Defines Metatorial Framework
Editors Make Assignments
Authors Create Components
Editor/Metator Reviews and Approves Components
Any Other Publication Process (Any Frequency)
Print Team Forms a Magazine (Once a Month)
Web Team Reviews New Components (Once a Day)
Auto-Email Process Runs (Once an Hour)
Auto-Syndicate Process Runs (Continuously)
Print Team Builds and Tests Magazine
Web Team Tests and Builds New Pages
Emails Sent
XML Feed Sent
Magazine Goes to Print
New Pages Posted
Harvest Audience Profiles and Preferences

Figure 10-2: The publication process if a CMS is present

In the old model, Web production happens after print production is complete. In the new model, the print and Web production happen in parallel. You can approve each chunk of content separately and move it into its range of publications independently. You can, therefore, update the site continuously, even if the magazine comes out only once a month. Even before a full article receives approval, you can approve its abstract or headline and push it to the Web or other electronic channels, such as e-mail, to build excitement for the coming full release.

Producing multiple simultaneous publications isn't without its problems. In fact, to create such a system, your organization must overcome a number of serious hurdles, including the following:

- **The attitude of content contributors** who normally think of only one target as they create.

- **The need to share content** across output formats, which mandates that you store the content in an output-neutral format such as XML.
- **The need to store and organize more information** than is necessary for any one publication. In the magazine example that I describe earlier in this section, the writers must organize and upload their research as well as the small article it produces. In the past, they could safely discard this extra information.
- **The need to create publication designs** that you can fill automatically. This situation is especially a problem for sophisticated print publications, where designers normally handcraft every page. If handcrafting is to occur in the context of a CMS, it must do so as a post-process, after you draw content out of the CMS repository.

A CMS is the clear choice for an organization that needs to create multiple simultaneous publications. Even so, using a CMS for this purpose doesn't mean that the process is necessarily an easy one. Automatically creating simultaneous publications that are as "good" as those that you create through separate, targeted processes is a difficult task. (*Good*, in this context, means of the same quality as if you were to prepare each part of the publication by hand with the utmost care and finesse.) On the other hand, a CMS approach offers the possibility of creating better publications than a manual process does. (*Better*, in the CMS context, means fuller, more timely information, targeted more directly to the consumer, that you deliver in whatever format the consumer needs.)

E-commerce

A few years ago, Web technology matured enough to deal with monetary transactions. After it did so, the rush to e-commerce began. Every organization with even the most rudimentary Web site began to plan for how to accept credit cards. Certainly, large organizations had been making electronic transactions for years — but not using the young Web technologies. In the rush to e-commerce, many organizations lost sight of the wider uses of the Web and other electronic channels. E-business, as I define it, became reduced to e-commerce. As the furor over transactions died down, the core of e-commerce emerged in the form of an online catalog that site visitors can browse through and buy from. In addition to the basic catalog, e-commerce encompasses a range of supporting functionality, such as exchanging money across the Web; hooking into legacy systems for inventory, pricing, and fulfillment; and other commerce mechanics such as the use of a shopping cart, where customers can add items and see how much they're spending.

Cross-Reference You find more information about e-business in the Introduction.

Catalogs and content management

The demands of a full-featured, usable, accessible catalog are content management demands. In particular, the catalog system must do the following:

- **Collect information:** The workflow and editorial process behind getting a product into the catalog mirrors that of any CMS. You must, for example, figure out what information that you need to collect for each product. (That is, you must define the product component.) In addition, you need to establish the steps that take each product's information through review and approval. (That is, you must establish a workflow process.)
- **Manage information:** A catalog is a repository. Management and administration of the catalog follows all the familiar trails of a content repository. You need to delete or archive old products, for example, and you also need to periodically review and update certain pieces of information (for example, price and availability).
- **Publish information:** To be useful on the Web, the catalog system must show product information on an HTML page. It may need to contribute to a print catalog as well. Thus, the catalog system may need the same sort of query and template capability that a CMS offers.

The content in the catalog has many of the same needs as any content in any CMS. This does not mean, however, that the catalog always needs to be integral to the CMS. The catalog data may be stored within the CMS repository, or it may be better to store the catalog in a separate database. If, for example, the catalog is already in a separate database whose collection and management procedures are established, there may be little point to moving the catalog content into another system. Also, a product catalog often has built-in relationships to other enterprise systems, such as inventory and manufacturing systems that would be hard to migrate or duplicate in the CMS. Thus, it is not obvious where the content of the product catalog is best kept. It depends on the situation.

Regardless of where you store the catalog, however, using the CMS to publish the product content is usually the best way to go. Catalog databases rarely have robust templating capabilities. And to standardize look-and-feel across all publications, your best bet is to use a single system. If you intend to store the catalog outside the CMS repository, you must make sure that the CMS can get product information out of the catalog system and format it for presentation on publication pages (as most can).

Tip

In some cases, your best route is to store the basic product data in a separate catalog system and supporting information in the CMS. If the existing system stores only pricing, part numbers, and a short description, for example, you can store a picture, a long description, and reviews of the product in the CMS. Then, after you build the product page, the CMS can take some information from the catalog system and some from the CMS repository to create the complete presentation.

E-commerce functionality and content management

Next to the presentation of a catalog, e-commerce most involves the functionality to support and conduct transactions.

Cross-Reference

I detail the relationship between functionality and content management in Chapter 4, "Functionality Is Content, Too!" You also see this relationship in each of the functionality areas that I cover in the following sections.

Exchanging money across the Web

Consumer e-commerce sites that accept payment on the Web generally communicate with the payee's bank behind the scenes to complete the transaction. Business sites reference an existing purchase order or, in some other way, validate that the transaction can go through. In either case, exchanging money across the Web isn't about information as much as it's about functionality. So although a CMS doesn't create the capability to exchange money on the Web, it can be responsible for putting the capability to exchange money on the Web on a particular page at a particular time. The CMS can treat the functionality that enables money exchange as a piece of "content" that it needs to manage and present.

Legacy systems and content management

Companies found out pretty quickly that e-commerce systems needed to talk to a lot of other corporate systems to support any significant amount of sales. In fact, they needed to fully integrate these systems into a vast range of existing (or legacy) data systems. Many software products help connect Web pages to older systems. The best of these products today use XML as a translation layer between the legacy system and the software that runs directly on a Web server. Exactly as in the case of exchanging money across the Web, these connections are functionality and not information. Thus, a CMS doesn't create the capability to connect to these other systems; instead, it gives you the capability to put the code that enables connections on the right page at the right time.

Take the example of a chunk of computer code (the functionality) that passes order information to a legacy inventory system and returns a "yes" if the item is in stock or a "no" if the item isn't in stock. The inventory system manufacturer or some third party that specializes in integration likely created this functionality outside the realm of the CMS. After its creation, however, the chunk of code (or the chunk of code that invokes the chunk of code) can enter into the CMS repository, where the CMS tags it appropriately and delivers it to the pages that need it, just as it does with any other piece of information. In this way, the CMS manages the delivery of the functionality without needing to know anything about what the functionality actually does.

Tip

Alternatively, you can type the code directly into the CMS templates that produce the pages that need the functionality.

Shopping mechanics

The shopping baskets, order status screens, and order histories that are common on e-commerce sites generally consist of functionality that interacts with Web databases and legacy systems. As such, these types of functionality follow the same path through the CMS as transaction functionality and legacy integration functionality—that is, the CMS manages the presence of the code on the page but doesn't play a significant role in creating the functionality or exchanging data with the background databases with which the functionality communicates.

Knowledge Management

The term *knowledge management* has significant currency in today's computer vernacular. In fact, it has much more currency than the less glamorous concept of content management. Furthermore, in many a conversation, I find that people who are uncomfortable using the term *content management* freely use the term *knowledge management* to describe the process that I define here as content management. The vagueness of both terms notwithstanding, the knowledge-management industry addresses a related but different concern from the content management industry. Although knowledge is a kind of content that you can manage, the core of knowledge management is discovery and synthesis, while the core of content management is collection and distribution. Still, knowledge that you discover or synthesize you must also collect and distribute. To my mind, the appropriate relationship between the two disciplines is that content management is the infrastructure to amass and distribute the knowledge that your organization identifies by using knowledge-management tools.

What is knowledge management?

The knowledge-management industry serves organizations with vast but balkanized repositories of data and terabytes of word processing and other binary files. These organizations pose what seems at first blush a modest question: What do we know? With all this information at your disposal, surely we know a lot, but how do we get to it? A number of answers come back from product companies and a few academics, each of which addresses a different part of the knowledge management issue, as the following list describes:

✦ **Synthesis and autodiscovery:** Through advanced data mining, analysis, and synthesis, many products seek to find the needles of knowledge in an organization's data haystack. Using the latest search technologies, they index and pattern-match their way through the terabytes of text and data to isolate just the right snippets of knowledge. This approach attempts to electronically winnow-down vast information stores to squeeze out the knowledge.

- **Categorization:** Many products attempt to categorize and index work product as you create it. By being embedded in every application that your staff uses to create work product, the systems attempt to proactively capture and route significant information to those who need it.
- **Knowledge portals:** Many products take the approach of organizing and personalizing an organization's knowledge into a single easy-to-use interface. They seek to connect all the diverse sources of information and put them all at your fingertips in a one-stop shop on your computer screen.

If you put all three of these approaches together, you get a more complete answer to the question of how an organization knows what it knows. Your organization must obtain tools for synthesizing and collecting knowledge from the vast data and file stores that you now possess. Furthermore, you must obtain tools for correctly cataloging new information as you create it. Finally, you must deliver the knowledge that you collect as an integrated and personalized view into what each staff member needs to know, given that person's current tasks and concerns.

Knowledge is content to manage

Each of the three approaches to knowledge management that I describe in the preceding section relates to the concept of content management that I develop in this work.

The same synthesis and auto-discovery tools that you may use to ferret out knowledge in the organization, you can also use to help discover and tag content that a CMS is to collect and manage. Unfortunately, these sorts of tools do little more today than prescreen and tag easily discerned metadata.

The categorization tools that live within a content creator's native environment play a larger role in a CMS. Many CMS products include categorization interfaces that show up as new buttons or menus inside authoring applications such as Microsoft Word and Macromedia Dreamweaver. These interfaces enable nontechnical users to store and retrieve content in a CMS repository without knowing anything particular about the CMS or its repository.

The idea of knowledge portals is where knowledge-management systems and content management systems share the most in common. In fact, the creation of personalized pages of information that you draw from a variety of sources is much more a content management than a knowledge management task. The templating and personalization systems within a CMS are portal-creation tools.

Although knowledge management most concerns discovery and synthesis of data and content into information for decision support, content management most concerns collection and distribution of any information. Clearly, a CMS can help collect and distribute the kind of information that enables us to know what we know.

Online Communities

In society, *communities* are groups of people that share some common purpose or kinship ties. The situation is no different online. Although many people talk about creating online communities, few do so from this sort of understanding. The most popular concept of community I have heard is to become "the place" to go for some sort of information. If you add a chat room or a threaded discussion, and collect user data, it is called a community. Many community sites hope to "own" the attention of their users and sell it to advertisers or direct it toward other organizational goals.

Without real community, however, which offers a core of common purpose or kinship (in the widest sense of sharing some important aspect of life), these sites can never really be communities. To be a real community, an online community needs to fulfill its members' needs for affiliation and knowledge. Affiliation is the members' desire to belong to something. Knowledge is the members' desire to know something. More to my current interest, a Web-based community needs a system behind it to support both affiliation and knowledge.

What is a community?

An *audience* is a group of people who share common traits and to whom you can communicate in the same way. To become part of an audience (for example, women in their childbearing years), individuals needn't know about each other or care to communicate with other members. An audience is like a neighborhood where each family keeps to itself. Each family is likely to display similar traits (income level, interests, hometown, and so on), but they don't consider themselves really *part* of the group of people who live near them.

In contrast, *communities* are like neighborhoods where everyone knows each other. These neighbors share experiences and knowledge, have a lot of cross-communication, and most important, a feel that they're part of the neighborhood. The neighborhood undoubtedly contains meeting places (schools, parks, the grocery store, and so on) where members congregate and share information and advice and feel a part of a larger group.

Online, the same analysis holds. A community site provides a meeting place where members can congregate and share information and advice and feel a part of a larger group. While all sites should communicate to one or more audiences, community sites go further by fostering communication among audience members.

How are online communities constructed?

Figure 10-3 shows the major components of an online community system.

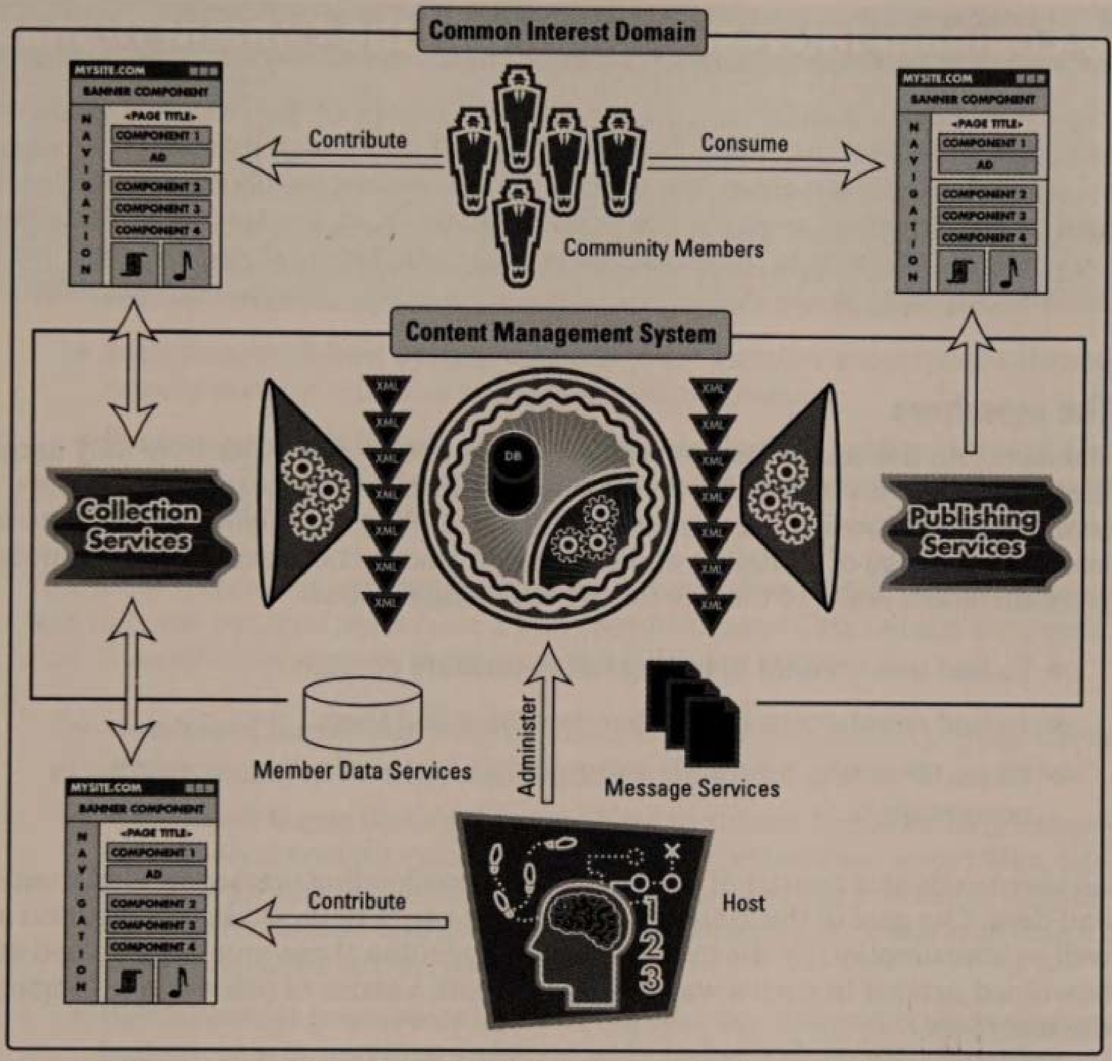

Common Interest Domain
MYSITE.COM
BANNER COMPONENT
<PAGE TITLE>
COMPONENT 1
AD
COMPONENT 2
COMPONENT 3
COMPONENT 4
NAVIGATION
Contribute
Consume
Community Members
Content Management System
XML
DB
Collection Services
Publishing Services
Member Data Services
Administer
Message Services
Contribute
Host
1
2
3

Figure 10-3: An overview of the CMS behind a community Web site

The common interest domain

The *common interest domain* is the boundary around the community. It's the realm of content and interaction. It's the basis of a community. For all the members, it offers a reason why they come together. Specifically, the *domain* is a statement of purpose for the community. The statement can be as general as "We love Barbie dolls" or as specific as "We're all female C++ developers working at atomic-accelerator laboratories on software designed to track the trails of subnuclear particles." Whether specific or general, the statement must clearly define the entrance requirement of the community. It's absolutely the first thing that you must determine about the community, and the rest of the structuring of the system springs naturally from it. The common interest domain defines what members affiliate with and on what subject they want knowledge.

The Common Interest Domain and the Content Domain

The common-interest domain shares a lot with the concept of the *content domain*. As I describe fully in Chapter 26, "Designing Content Access Structures," the content domain is also a boundary and a statement. The content domain defines the boundary between the content that's relevant to manage and the content that isn't. It's a statement of the range of coverage of the CMS, such as "All content pertaining to the history of Barbie dolls."

The members

Members join the community for affiliation and knowledge. After they fully immerse themselves in the community, they contribute as much of these goods as they receive. The purpose of a community Web site and its underlying system is to facilitate the exchange of affiliation and knowledge among the members. Much more tangibly, members come to the site for the following reasons:

- To find new content of interest or contribute content.
- To find members or have other members find them.
- To participate in a communication forum such as an online meeting or presentation.

All these activities consist of uploading and downloading messages, files, content, and data. The goal of the system that runs the site is to facilitate contribution as well as consumption by the members and to position these mundane upload and download actions in such a way that they create a sense of place and belonging in the members.

The CMS behind the community

If the community site is to draw a wide member base, it must provide enough relevant and accessible content to warrant attention. If the site is to hold the attention of its member base, the content must continually grow and deepen. To accomplish this level of content management, a system is necessary. As is true of any CMS, the site must collect, manage, and publish content. Specifically, in the context of a community site, the CMS must perform the following tasks:

- **Enable members to actively contribute** to the knowledge base of the site. Not only does this capability provide a wide base of knowledge flowing into the site, but it also brings affiliation to its maximum depth. Members are most affiliated if they contribute as much as they receive from the community. On the other hand, member contributions are useful only as long as they're pertinent and well-structured enough for the site to effectively store and its members to later find. Thus, building a strong but not overly complex metadata framework that naturally guides members to contribute relevant well-tagged information is particularly important in a community site.

- **Provide a repository** with a fine level of categorization to support maximum personalization.
- **Include the semistructured sources** that come out of the message center. For example, include email threads and messages that can be made into components.
- **Enable the repository to grow** in a constrained way with content expiring as necessary, missing, or ambiguous information clearly identifiable, and new content areas ready for publication to the community members.
- **Include a workflow system** that can route member submissions through community staff for editorial and metatorial processing.
- **Present relevant information** targeted to what the system "knows" about the member. This kind of publishing requires a robust templating system and well-developed personalization capabilities.

To enable structured content to come in to the system and targeted, relevant content to come out, you must have a fairly sophisticated CMS behind a community site. The collection system must have the following features:

- **Web-based forms** that lead members through the process of submitting small pieces of original content (content that they type into fields on-screen).
- **Web-based forms** that enable members to submit files with larger pieces of preauthored content (word-processing files, video files, sound files, and so on). The forms must guide the members through the process of tagging the files with metadata and give them the opportunity to upload the files from their computers to the repository.
- **Batch-content processors** that can harvest the information from community bulletin boards and chat sessions and catalog it along with the other submissions.
- **E-mail acceptors** so that members can e-mail content directly to the community CMS.
- **Syndicated-content processors** that can accept and process information feeds from commercial providers. The site may, for example, include daily targeted news that an outside service provides, delivering stories electronically throughout the day.

To provide relevant information to members, the community CMS must collect user data and use it to subselect content to present to each user. In addition to using member data to target content, you can use member data on a community site to target certain members to other members. Such a member match-up service provides for a much greater sense of affiliation. Members, in the end, want to affiliate with other members — not to a Web site.

To match members to members or to match content to members, the system must collect categorization data on each member. In fact, it must categorize them in a way quite similar to the way that it categorizes content. In this way, affiliation groups can form around subject matter of common interest. In the successful system, the categorization of members and the categorization of content happen hand-in-hand, where one continually deepens the other. Of course, making friends involves more than answering questions the same way. So, the successful member-matching system needs to go beyond mere matching to enable members to experience each other's company and decide for themselves with whom to affiliate.

On the more mundane side of matching members to content, as in any personalization system, the site must have mechanisms for the following tasks:

- ✦ Gathering member data.
- ✦ Tagging content.
- ✦ Mapping the type of data gathered to the appropriate tags in the content.
- ✦ Dynamically rendering the selected content within a standardized page. *Dynamic rendering* is the function of CMS templates. They contain the software code necessary to determine who the user is, access her member data, and match her with the content that is most relevant to her.

Cross-Reference For more information about templates, see Chapter 22, "Designing Publications."

The message and user services

The message services provide the communication hub for the site. They include any or all of the following technologies:

- ✦ Basic e-mail
- ✦ Chat and hosted forums
- ✦ Threaded message boards
- ✦ Online meetings
- ✦ Online presentations
- ✦ Member location services
- ✦ Member classified ads and goods or services exchanges

The exact number and types of technologies in use depends on the common interest domain, the computer savvy of the members, and their degree of affiliation. Generally, the more affiliation your community can muster, the more members put the time and energy into these communication channels. The best sign of a low-affiliation community is one where all the bulletin boards are empty.

The CMS behind the community must obviously support these communication vehicles. In particular, although the CMS doesn't control the communication services themselves, it's responsible for rendering the interface for these services on the Web pages that the members access. In the same way that the CMS can target content to members, it can target particular communication services to users. It may, for example, put a registration form for an online meeting on the home page for all members who express an interest in the subject matter of the meeting. In addition, the CMS must harvest information from the communication services and successfully transition their semistructured, real-time output (like meeting notes and sound clips) into more enduring content that you can deliver by using the same targeting techniques as any other community content.

Cross-Reference

For more information about using a CMS to manage functionality such as that mentioned in the preceding text, see Chapter 4, "Functionality Is Content, Too!"

Member data is essential to the system behind the community. In addition to serving as the basis for personalization, the system needs this data for a variety of other purposes, such as the following:

- Member bulletins and global e-mails
- Member rights to particular content
- Member rights to the communication services
- Member rights to submit and modify content
- Administration of member fees or other initiation rites

For all these purposes, the site needs a strong and extendable user data-management system. That system is the same as what a CMS uses to do personalization.

The host

The community's host is the organization that's in charge of the site's infrastructure and maintenance. This organization must implement a CMS, communication services, and member-administration services. It must also input enough content to get the site off the ground. You find the following two typical types of hosts online:

- **Commercial hosts:** A *commercial host* has the members as a target market for goods or services. This host is willing to trade the cost of maintaining the site in return for exposure to the members. In a typical scenario, the host has the original idea for the community, creates an initial implementation of the site's system, fills the system with enough content to make it viable, and then launches the site and opens it to members. The host continues to feed content in, administer user data, and create communication events. The major issue to resolve in this circumstance is what right the host has to market to the members or sell member data to outsiders. Members can leave if they feel exploited. On the other hand, the host may stop if it sees a lot of cost and little return from the community.

- **Member hosts:** A *member host* is one or more potential members who decide to create a Web presence. Typically, some existing trade or interest organization with a current membership organizes and funds the initial system. As in the case of the commercial host, the member host creates an initial implementation of the site's system, fills the system with enough content to make it viable, and then launches the site and opens it to members. The key issues here are continued funding of the site from often cash-strapped organizations and sufficient attention paid to the site maintenance by what is often volunteer-run organizations. Of course, selling member data is also an issue for member-hosted communities.

For both the commercial host and the member host, the primary issue is to make the site truly belong to the members. As time goes on, members should become the major contributors to the site, with the host needing to supply less and less content. In a high-affiliation community, members even plan and execute the communication events (chats, online meetings, and so on). If the community is successful, its success comes from the host's creation of a system that promotes affiliation and targeted knowledge-gathering — among a group of people who naturally gravitate to a clearly stated and well-founded common interest.

In any case, for the community to become successful, it must have a full-featured and robust CMS behind it. Without a CMS, the site can serve neither its members nor its host. At best, it becomes an enhanced bulletin board, and at worst, it becomes an unenhanced bulletin board.

Summary

Although each of the electronic technologies that I describe in this chapter has its own segment of the computer industry focusing on it, I hope that you can see clearly that they all share the need to collect, manage, and publish content. To this extent, any of these technologies can benefit from the concepts of content management. Moreover, a well-constructed CMS can serve the needs of any of or all the following technologies:

- Personalization systems need the content tagging, selection, and targeted delivery of a CMS.
- Advanced Web sites buckle under their own weight without the organization and efficiency of a CMS.
- Without a CMS, producing multiple publications from the same content base is, at best, a very long process and, at worst, impossible.
- E-commerce systems benefit greatly from the delivery system in a CMS.

- Knowledge-management systems needn't reinvent the collection and publishing software already present in a CMS.
- Online communities can position a CMS at their core to help match members to content and to each other.

In the next chapter, I move on to content management projects. I explain how you may go about getting your organization behind a well-thought-out and presented design and how you can get through CMS implementation with the least hardship.

✦ ✦ ✦

PART

III

Doing Content Management Projects

✦ ✦ ✦ ✦

In This Part

Chapter 11
Staffing a CMS

Chapter 12
Working within the Organization

Chapter 13
Getting Ready for a CMS

Chapter 14
Securing a Project Mandate

Chapter 15
Doing Requirements and Logical Design

Chapter 16
Selecting Hardware and Software

Chapter 17
Implementing the System

Chapter 18
Rolling Out the System

✦ ✦ ✦ ✦

CHAPTER 11

Staffing a CMS

By looking at the kinds of jobs a CMS requires, you can get a lot of insight into what it will take to make one happen. You can classify the jobs involved in a CMS in the following ways:

- Each phase of the CMS process (collection, management, and publishing) has its own jobs and responsibilities. In a well-planned CMS, the people in these three areas can do their jobs without becoming overly entangled in the jobs of people in the other areas.
- CMS jobs generally break down into analysis and implementation. Analyzers figure out what to do and implementers do it.
- There are jobs that you need to design and start up a CMS and there are those you need to run one. Often it takes a large team to start up a CMS but a small team to run it. A common method is to outsource all or parts of the CMS design and implementation effort to an outside vendor while simultaneously putting together the longer-term team that will run the system once it's complete.
- CMS jobs can be categorized by discipline, including managers, information architects, infrastructure staff, and software developers.

In this chapter, I use the last system here to categorize the jobs in a CMS because it balances being comprehensive with dividing the jobs into more than just a couple of categories.

About the CMS Jobs

Your CMS may affect the jobs of numerous people in diverse areas throughout your organization. Few of the tasks that need to be accomplished to start up or run a CMS require full-time long-term staff. Rather, you start up your system with a large short-term staff and run it with a small full-time staff and a larger casual and part-time staff.

◆ ◆ ◆ ◆

In This Chapter

A brief perspective on CMS Jobs

How content managers and staff members control the overall project

How a CMS ties to the business through a business analyst

The role of information architects

The role of the CMS Administrator and other infrastructure staff

The role of the programmers in a CMS

The CMS staff positions that create publications

The staff you need to convert content

◆ ◆ ◆ ◆

Content management is a difficult task—not only does it bring together a large number of people from quite diverse backgrounds, but it also requires individuals who are personally split in their skills and attitudes between different, often conflicting, disciplines. You will find that almost all of the jobs in this chapter call for at least two widely different skills.

The array of jobs I present assumes a very large organization with a big team. I do this to show the most complete picture of the jobs that need to be accomplished. Obviously, in smaller organizations, or in large projects in earlier stages, one person will do many of these jobs. Understand, however, that each job does need to get done in its entirety, even if there is only one person to do them all.

Note I apologize in advance for leaving out your current job title.

In addition, my intention is not to write job descriptions that you can cut and paste and send to your HR department to be filled. Rather, I am grouping the necessary tasks involved in the startup and running of a CMS into logical clusters around personality and task types. You will need to look carefully at your particular situation. Depending on whom you have already at your disposal, who you are yourself, how many people you can feasibly bring onto the team, and how long you have to accomplish the project, the cutting and pasting you are most likely to do ranges from the many positions I have outlined here to the few that you can afford or desire.

Finally, my goal is to be complete, but not exhaustive. You will find the most detail about the jobs that are particular or most pivotal to a CMS and little or no detail about those that are the same with or without a CMS.

Table 11-1 shows how the various positions I detail in this chapter might be distributed between the startup phase of your CMS, where you plan and implement the system, and the run phase of the CMS after it is live. The positions I do not list in a particular column may still be needed but are not the core jobs in the startup or run phase of a CMS.

Table 11-1
CMS Jobs at a Glance

Staff Position	*CMS Startup*	*CMS Run*
Managers	Content manager, Project manager	Content manager, Production manager
Business analysts	High level of participation	Periodic participation
Information architects	Content analyst	Metator
Infrastructure staff	Deployment analyst, Trainer and documentation specialist	CMS administrator

Staff Position	***CMS Startup***	***CMS Run***
Software developers	Software analyst, Template and CMS developer, Custom application developer, Software integrator, Test analyst and test engineer	Template and CMS developer, Test engineer
Publications staff	Publication analyst, Publication designer, Page developer, User interface specialist	Publication designer, Page developer
Content processing staff	Conversion analyst, Tool developer, Content processor, Content QA specialist	Content processor, Content QA specialist
Content creation staff	Writers and other content creators, Editor	Acquisitions manager, Traffic cop, Writers and other content creators, Editor

Managers

As in all group efforts, someone must be in charge. Clearly, an enterprise CMS is too big not to have its share of chiefs. In addition, as you can see from the number and range of positions in this chapter, it is not reasonable to expect that your management staff themselves will have the skills to do all of the work behind a CMS.

Content manager

More and more, organizations are realizing that content management exists, and that someone ought to be responsible for it. That job has loosely come be called Content Manager (or Director/VP/Chief of Content Management). Although this job sometimes falls to someone with a loftier title, such as Chief Information Officer (CIO) or VP of Electronic Media, the idea is the same. Conversely, the title Content Manager might be given to someone who functions more like an acquisitions specialist, or, in a smaller organization, even a Web page creator. Variations in usage notwithstanding, a Content Manager's responsibilities might include the following:

- **Fully understanding the discipline** of content management. For example, they might be required to actually understand everything in this book!
- **Leading the planning and execution** of the organization's CM initiatives. They might be the lead of the CM project team and the person with ultimate responsibility for a successful CMS implementation.

- **Representing the needs of the CMS** to departments throughout the organization. This person must teach, negotiate, and pester groups throughout the organization to get them to help, or at least not stand in the way of, the CMS.
- **Representing the needs of the CMS** to the outside world. The Content Manager is likely to be the person to preside over the selection of CMS hardware and software and to be the official interface between the CMS system and third-party products with which the CMS interoperates.

All in all, the Content Manager is the head of the CM endeavor in the organization. Because there is no degree (that I know of) in content management, there are no recognized curricula to pass to get credentials for this position. In addition, few people can say they have much experience being in such a position. Still, there is a core set of skills that you can look for in a Content Manager. The person must be capable of understanding the multiple disciplines involved in CM. They ought to have a good head for at least the following subjects:

- **Web technologies,** including server and client software and the construction of Web sites and Web applications. She does not need to be a Web developer, but a good Content Manager understands the Web.
- **Editorial processes,** including the writing and review process. She does not need to be a writer or artist, but a good Content Manager must recognize well-constructed content when she sees it.
- **Cataloging and information organization.** A Content Manager does not need to be a librarian, but she does need to be good at finding structure and making sure it is enforced.
- **Information technology infrastructure,** including database and network administration. Because the system has to integrate with the enterprise communication infrastructure and because the collection, management, and publishing systems of a CMS are all network and database applications, some knowledge in this area is a must.
- **Analysis and abstraction.** This may be the most important but least measurable skill of a Content Manager. If you haven't noticed, content management requires you to think very abstractly about information. If you get too stuck in the concrete details of how one page or another looks, you will lose sight of the main goal of CM, which is to break information away from its presentation and instead focus on its structure and how that structure can be used to make any of a range of pages. If this light bulb has not appeared over your Content Manager's head, you will never have a robust system.

Project manager

All project managers attend to the scope, schedule, and budget of their projects. It is really no different in a CM project. Project managers usually appear most in the start-up phase of a CMS. After the CMS is running, the other sorts of managers tend to step in and take charge.

Project managers do the following:

- **Manage project staff,** apprising them of their deliverables and schedules. Perhaps the most difficult thing in managing a CMS project is the potentially long list of characters whose input is needed. Especially during the requirements-gathering phase for the CMS, there are a lot of meetings, phone calls, e-mail messages, and small documents that have to be scheduled, accomplished, and accounted for.
- **Manage budgets,** assuring that the money allotted for the project lasts as long as the project does. Conversely, a good project manager will see an overrun a mile away and let the world know that some sort of change is needed long before the money actually runs out.
- **Create and enforce the project plan.** It might well be beyond the ability of a project manger to create the plan for a large CMS, but it should be fully within her ability to make sure it is updated and adhered to.

The kind of project manager that you want is the kind that can keep a lot of balls in the air and make sure a thousand little things get accomplished and no one is left out. Project managers with strong technical skills but little ability to negotiate competing constraints need not apply.

A good CM project manager (PM) will have strong negotiation skills and a streak of insistence. The successful CM project is a careful balance of stopping the unending conversation about what content and functionality will be included, and assuring that all relevant and necessary content and features are fleshed out and used. Finally, although your PM need not be an expert in any of the CM skill areas, she should have somewhat the same ability as the Content Manager to understand the staff's subject matter enough to know a big problem from a little one.

Production manager

A production manager ensures that content flows into the system at an established rate and quality. I have seen production managers most often in the context of acquisition source conversion, where large bulks of source content have to be processed under tight deadlines. Production managers are called for, however, wherever there is a steady, high volume of content to be produced. If, for example, you need 100 images drawn, checked, converted, and deployed each week, you might want to put a production manager in place to assure that the volume and quality you expect is consistently accomplished. Production managers work both in the start-up phase of a CMS, to convert the backlog of information, and during the "run" phase of a CMS, to assure the required throughput of content.

Production managers do the following:

- **Manage production staff.** They hire, fire, promote, and motivate. You may think that in a production environment, the staff is expected to act like cogs in the machine. On the contrary, a strong production team is one where people's individual skills are recognized and used. Staff can move from entry-level jobs that require little initiative and skill but high focus to more advanced jobs managing the newcomers, handling the special cases that arise, and designing more effective processes.
- **Design production processes** that make the most efficient use of people and the most effective use of automation. A good production manager knows when something is worth programmer time to automate. She can tell when a problem is due to inattention, rather than bad input or faulty procedures.
- **Create and enforce production schedules,** ensuring that content creation proceeds like a well-oiled machine. She will also have methods at her disposal to even out, as much as possible, the inevitable lows and highs in content flow through her system.
- **Create and enforce quality** and speed metrics. In a production environment, you must create a set of gauges that tell you at any moment how the process is going. After creating such gauges, a good production manager will pay close periodic attention to her gauges to catch problems before they snowball.

A good production manager is a cross between an editor, a programmer, and a factory manager—not the easiest person to find. The editor in her understands the way information is constructed and can design processes that editorial staff can and want to use. The programmer is always looking for a way to do it more quickly and knows the hallmarks of a process that can be automated. The factory manager understands the dynamics of the "floor," where emotions as well as tools determine how much gets done in a day.

Cross-Reference

There is much more information about production processes in Chapter 30, "Processing Content."

Business Analysts

Someone must recognize or create the wider business strategy into which the CMS must fit. I call that person the CMS business analyst. Without someone to do this job, a CMS project team can quickly become unmoored in the organization and may find itself in the position of asking for a lot of money for a project that does not seem to do much more than save your group some time. The business analyst is most active in the very early stages of a CMS implementation project, when the business case and mandate for the project are being crafted.

The business analyst does the following:

- **Figures out how the content management project fits** into the overall strategy of the organization. Part of the job is finding justifications for the plans you may already have, but the more important part is finding out how to change your plans so that the project fits cleanly into the organization's most important goals (see Chapter 13, "Getting Ready for a CMS," for more information).
- **Finds out what has been accomplished to date** in the organization. The analyst must be able to track down and document the existing efforts that might contribute to the CMS.
- **Creates a strategy for cooperation and support** in the organization. This job includes determining who the appropriate sponsors for the initiative are, as well as what groups need to be represented and by whom at planning and strategy meetings (see Chapter 14, "Securing a Project Mandate," for more information).
- **Presides over the mandate process.** Although the analyst might not actually facilitate the mandate meetings, she is responsible for ensuring that they reach a successful conclusion. At the end of the process, it is the business analyst's responsibility that there be widespread consensus and support for the CM initiative.
- **Creates and promulgates the project mandate.** The analyst is likely to draft the project's mandate statement, solicit feedback, and distribute the final version. She can serve as the point person for any discussion in the organization about what the project is or how it will be done.
- **Devises a strategy,** in consultation with others in the organization, for how the workload and budget of the startup and running of the CMS will be shared in the organization. In addition, she will have to ensure that the governing body created to preside over the effort coalesces into an ongoing working body (see Chapter 12, "Working within the Organization," for more information).
- **Works with the project team** to ensure that the project mandate is fully embraced and specifically acted on.

There is a lot of responsibility in this position, and it is by no means cut out for a lone ranger who works in isolation. In fact, what the business analyst really must do is ensure that these tasks are accomplished — not do them herself. Most of these tasks require other people to come together and agree. The best person for this job, then, will be someone who has the following qualities:

- **Knows the business.** Obviously, to accomplish the tasks I laid out, this analyst must be able to move nimbly throughout the organization. It really helps if this person knows a lot of people (or at least makes friends quickly).
- **Knows how to negotiate.** All of the analyst's tasks require consensus-building. It is essential that the analyst be considered neutral. It is a great help to her if she is already well respected in the organization.

- **Understands the concepts and execution of a CMS.** The analyst must have a solid understanding of what a workable mandate looks like if she is to facilitate its creation. Without such an understanding, all of the negotiation talent in the world might still lead to an agreement that will not work or be too expensive.
- **Can synthesize and motivate.** The analyst will be faced with a lot of opinion and fact. She will have to quickly distinguish one from the other, sort them, relate them, and suggest ways to combine them into a best approach. Unfortunately, she will not always have time to think first. Therefore, a good ability to think on the move and also acknowledge of how to postpone a decision on issues that do require more thought, as well as the stature to be listened to when she speaks, is a must for this position.

The obvious place to look for business analysts is among the MBA/consultant types who have considerable business, communication, and Web skills. These people are likely to have the interpersonal skills. But remember, they also need the respect of your organization.

Information Architects

Information architects (IA) create and implement strategies for structuring, accessing, and displaying information. They know and can harness the "mechanics" of information to create methods and models for how to structure the content within the CMS and within each publication that the CMS will produce.

Like content management itself, information architecture is a new and evolving discipline. Where you find most IAs practicing today is on the Web, creating layout and access schemes for important sites. Thus, many IAs have worked previously in graphic design, editorial services, or web development. A few universities are beginning to take IA seriously enough to start to formulate and teach its principles. The Argus Center for Information Architecture (`http://argus-acia.com/`) is a good place to get information from the guys who "wrote the book."

An IA with good experience on a large (10,000-plus page) Web site ought to have the right experience to work on a major CMS project.

Note The same caveat applies to IAs as applies to content managers. Content management requires you to think very abstractly about information. If you get too stuck in the concrete details of how one page or another looks, you will lose sight of the main goal of CM, which is to break information away from its presentation and instead focus on its structure and how that structure can be used to make any of a range of pages.

For the kinds of work that you need done in a CMS, I differentiate IAs into content analysts and metators.

Content analyst

The content analyst does or oversees the middle phase of a CMS implementation project, including:

- **Gathering content requirements** and organizing them into a unified system (see Chapter 15, "Doing Requirements and Logical Design," in the section "What Are Requirements?" for more information).
- **Conceptualizing the logical design** for the content aspects of the system (see Chapter 15, "Doing Requirements and Logical Design," for more information).
- **Translating the logical design** into a physical design in the implementation specifications (see Chapter 15, "Doing Requirements and Logical Design," for more information).
- **Creating staffing estimates** and plans for the collection effort.
- **Planning the architecture** and the organization behind localization.

Logical design is the center of the content analyst position and it results in what I call the metatorial guide. But the content analyst can participate throughout the project, helping from the first readiness assessment to the final training during deployment. Wherever the content analyst is, she is ensuring that the content you manage can be and is divided and tagged for maximum access and flexibility. The IA analysts will be central in gathering and processing the content requirements and turning them into a solid content model.

Cross-Reference Chapter 18, "Rolling Out the System," has more information on the metatorial guide.

To do the job I have laid out, IA analysts need these sorts of skills:

- **Business process design.** The collection system of a CMS is a business process for interacting with content authors and sources. To account for this in her metatorial framework, the content analyst must understand the people and processes that can or should be used.
- **Cataloging, storage, and retrieval.** Clearly, this is the heart of the content analyst's world. Without a solid focus and passion for cataloging schemes, the analyst will not succeed. It is not enough, however, to know how to divide and categorize information; the analyst also must understand the technology you intend to use to store and deliver that information. Thus, a strong focus on database or XML technologies is a must.
- **Publication systems.** In the end, the content analyst is there to ensure that the right content can be rendered in the right way in each publication. To do so, the analyst must understand the mechanics of each publication to be produced. For Web sites, this means a very good knowledge of Web applications and, in particular, the concepts and mechanics of Web templates. For print, this means a strong understanding of the particular print publication application that the

organization uses. An end-user understanding is a start, but to be successful, the analysts must also understand publication from the standpoint of one who wants to automate its layout and content structure.

The best content analyst is a librarian or editorial type who has a demonstrated ability to understand and apply technology. The content segments and metadata that the analyst proposes must be collected, stored, and published using programming and networking technology. To succeed, therefore, the analyst must not only know how information should be structured in general, but how it needs to be structured for the particular collection, management, and publishing systems the organization will use.

Metator

A content analyst creates a metatorial framework, and a metator applies it. Metators are quite analogous to editors (as you might have guessed by now). An editor reviews an author's work for style, usage, grammar, and so on, and makes changes to bring the work into compliance with the organization's standards. A metator does the same thing, but for metadata rather than for editorial qualities.

Cross-Reference There is an illustration of a metator's job in Chapter 7, "Introducing the Major Parts of a CMS," in the section "Metatorial processing."

A metator might be expected to do the following:

- **Review an author's submission** of a component to ensure that its metadata fields were filled in correctly.
- **Review the output of content conversion** processes to ensure that they correctly identify, divide, and tag components.
- **Fill in metadata fields** that authors don't understand how to complete, or that conversion processes can't locate.
- **Train and guide others** on what metadata is, how it is used by the CMS, and how to choose the right values for particular components.
- **Distribute and update the metatorial guide.** Metators are in the best position to see what is working and what is not in the system created by the content analyst. They will also be closest to those who must use the guide in their daily work. There is more on the metatorial guide in Chapter 18, "Rolling Out the System," in the section "Creating documentation."

A good metator is a content analyst in training. She has the same basic skills as the analyst, but less experience or talent. Of course, the analyst often must be the metator as well. Metators can serve as CMS front-line support to the content contributors, so it is a good idea to choose metators who have the skill eventually to be considered "power-users" of the CMS.

Tip

A content analyst can gain good insight into how well her system works by doing the job of the metator every so often.

You can treat the job of metator as an entry-level position — that is, because they are applying already-created guidelines, they need not know as much about the CMS or its technology. This is not to say that they need no relevant experience. Rather, the perfect person for the job of metator is someone who has had ample experience as:

- A technical writer who has had to focus on highly structured material
- An indexer who has had experience with electronic publications
- An editor (of course)
- A librarian or archivist who has a knack for technology

Infrastructure Staff

The infrastructure staff defines and builds the system on which the CMS will run. They are in charge of deploying the completed CMS and its related software and hardware and of administering the system once it is running.

CMS administrator

Someone must set up and run the CMS. I call that person the CMS administrator.

Note

I might have chosen to list this position under information architecture, even though most often it is thought of as an IT position. Here is why. The basic purpose of the CMS is to organize information and make it as available and flexible as possible. Thus, the person who runs the CMS should have this perspective above all else. Just as it is preferable to have a person with a background in finance running your enterprise accounting system, so it is preferable to have someone with a background in content running your CMS. This is not to say that CMS administrators do not need to be quite technical – of course they do. Rather, it is to say that they need to be able to confront content problems as well as infrastructure problems.

The administrator is responsible for all of the end-user tasks required to set up and run the system. Like all administrators, the CMS administrator is neither a programmer nor a network professional (although experience in these areas is a help). Rather, CMS administrators are responsible for tasks such as the following:

- **Configuring the chosen CMS.** The administrator enters or imports all of the component definitions, user profiles, publication definitions, and other data that the CMS needs to begin working.
- **Inputting content.** Where content comes into the system in bulk (for example, as the result of a conversion process), the administrator oversees the process and takes action when anything fails. Where content comes in one component at a time (for example, when authors fill in Web forms), she is responsible for ensuring that the proper methods are up and working. This may entail creating new Web input forms, distributing appropriate end-user software, or simply receiving e-mail attachments and importing them into the system.
- **Fully understanding the chosen CMS software.** The administrator ought to be able to make the CMS do anything it can do without additional programming. In many commercial systems, there is quite a bit that does not require programming. Much of what a CMS is capable of, however, requires you to understand content management concepts and to be able to find the right combination of options (which are often buried in text files and the system registry).
- **Maintaining users and workflows.** The administrator is responsible for ensuring that all staff that have access to the system are accounted for by the system. Staff may need to be profiled and added to groups in order to get access to the resources they need. In addition, the administrator needs to set up and maintain the system workflows. This may be as simple as dragging and dropping icons onto a page, or as complex as modifying a massive XML structure in a text editor.
- **Triaging bugs and finding fixes.** After the system is running, the administrator will be the first line of defense against problems that arise. From rebooting after a crash to spending days tracking down why one user can't seem to log in, the administrator must ensure that the train keeps running. Of course, many problems will need to be escalated past the administrator, but like the triage nurse in an emergency department, she sees them all first.
- **Ensuring data hygiene.** The administrator ensures that the content stored in the repository is in the best state possible. This responsibility includes making sure that required data is supplied, that content is archived or deleted on the specified schedule, and that periodic reviews and updates to content and system configurations happen as scheduled.

So, what sort of person can do all of this? Well, I see two basic ways to go: start technical and train on content, or start content-oriented and train technical. Either of these ways could work, given the right person. One woman I know, who graduated with a degree in library science, craved technology so much that she all but gave up on cataloging to pursue a career in programming. I know of a man who began as a network professional but gave it up to write. Either of these two people would make a great CMS administrator. If you can't find a hybrid soul like the ones I described, look for the following qualities:

- **Someone who is a super power-user** of every application they use. The power-user is always looking for a way to make the application do something more, different, or more easily and will stop at nothing until it all works. That is the attitude the administrator needs in order to deal with the powerful but vaguely documented and obscurely located functionality that many CMS products provide.
- **Someone who is not afraid to get technical** but knows where to stop. A good administrator will open up a programming script to see were it is crashing. A bad administrator will try to fix the script and not turn it over to the programmer who created it.
- **Someone who is not afraid of people and process.** At least half of the problems an administrator will encounter concern people, not machines. A good administrator will not always reply with a technology fix, but may change process, or simply diffuse emotions to solve a problem.
- **Someone who is meticulous.** The administrator is running a highly tuned, very complex machine. She must be careful that at all times the machine is well oiled, all the parts are moving as they should, and no dirt (bad content or data, that is) can foul up the gears.

Deployment analyst

A CMS can, but usually does not, stand alone in the organization. As likely as not, it will be connected to many of the organization's main computer systems, such as the staff information systems (network user directories), security protocols, network and database systems, and any of a number of enterprise applications for financial or employee transactions. Someone has to figure out all of these connections. I call that person the deployment analyst. This person's responsibilities include the following:

- **Determining what supporting hardware and software** the CMS will need and how to procure it.
- **Determining what platform and configuration requirements** the organization has for the CMS (supported databases, required operating systems, and the like).
- **Figuring out the integration of the organization's other computer systems** with the CMS. The analyst will need to work closely with the owners of the other systems to forge an agreement and alliance.
- **Determining how to deploy the collection system** of the CMS across as much of the organization as needed.
- **Determining how to deploy the publishing and Web-server system** of the CMS across as much of the world as needed.

- **Planning and overseeing the rollout** of the system in the organization.
- **Determining the appropriate training and technical support** infrastructure for the CMS.
- **Creating estimates and plans for staffing** the management system of the CMS.

Many of these tasks are common to any deployment of enterprise software. And unlike many of the other CMS jobs, there are plenty of people out there who have done this sort of thing before. Although the level of integration and the depth of penetration of a CMS can be larger than other enterprise systems, the skills required to perform the integration are the same. Thus, people with a strong IT and enterprise deployment background (with a focus on the Internet) are the ones to find.

Trainer and documentation specialist

By the time a large CMS is deployed, a tremendous amount of effort has been directed toward organizing information and creating processes to collect, manage, and publish it. Someone needs to write all of this down so that users can read it. Moreover, someone must develop and deliver training to the variety of people who need to interact with the system.

Not unlike other enterprise implementations, CMS documentation and training will likely be a combination of the materials supplied by commercial product companies and the custom processes you have developed. The person (or people) in this position has the unenviable need to be able to offer education to everyone from the most novice content contributor to the most expert developer who needs to know how a sophisticated integration was performed.

Software Developers

Some day, it may be that you can buy a CMS that does not require you to do large amounts of programming. Until that day (and most likely beyond), you will need a development staff. This staff creates the custom software behind a CMS. They create the templates that a CMS product needs. In addition, they create the code to implement custom features, integrations with non-CMS systems, and custom authoring and conversion programs and routines.

Software analyst

Like the other analysts I have described, the software analyst strategizes and plans. The software analyst plans how best to accomplish all of the programming that must be accomplished to start and run the CMS.

In particular, the software analyst is responsible for the following:

- **Choosing the development environment** (or environments) that will be used. There are quite a few competing environments from which to choose. The right one is the one (or ones) that requires the least retraining of the programmers you intend to use, fits most cleanly into the required system architecture supplied by the deployment analyst, and, of course, provides the tools you need to create the needed functionality.
- **Creating a development framework.** In a CMS, there is code everywhere. From the programming macros in a Microsoft Word document template, to the stored procedures in an enterprise resource planning (ERP) database, there are more programming sites than anyone in her right mind would call a single system. It is the job of the software analyst to get all of these different parts of the system corralled into a single programming framework. The framework organizes the creation of code, its versioning and testing, its deployment, and the sharing of code modules and objects between applications. This is no small task in an enterprise CMS.
- **Deciding which features of the CMS should be custom-coded** and which should be done using commercially available tools. For example, should the organization use the database connectivity code supplied by the CMS manufacturer or should they extend the code they have now, which works well with their ERP system?
- **Writing the specifications** and development plans for the programming tasks that require them (see Chapter 17, "Implementing the System," for more on specifications).
- **Creating estimates and plans for the staffing** needed to accomplish all of the programming to start and run the CMS.

As with the deployment analyst, there are people in the world who have done this sort of thing before. Still, someone who is a good general analyst but also has significant experience programming content-manipulation applications would serve you best. This is not a rare skill set, but it is not the most common. People of this ilk can be found in the back rooms of Web sites that use Perl extensively, or in the documentation groups of large companies that have had to use Structured Generalized Markup Language (SGML) to structure and manipulate massive document databases. In any case, stay away from the people who think that content can be treated just like data. They will underestimate the complexity of their task every time.

In addition, like the other analysts, this person cannot be a back-roomer. She must be able to negotiate among the competing development interests (which are no less, and often more, vehement than the business interests) and come to a workable compromise between competing platforms.

In particular, the software analyst is responsible for the following:

- **Choosing the development environment** (or environments) that will be used. There are quite a few competing environments from which to choose. The right one is the one (or ones) that requires the least retraining of the programmers you intend to use, fits most cleanly into the required system architecture supplied by the deployment analyst, and, of course, provides the tools you need to create the needed functionality.
- **Creating a development framework.** In a CMS, there is code everywhere. From the programming macros in a Microsoft Word document template, to the stored procedures in an enterprise resource planning (ERP) database, there are more programming sites than anyone in her right mind would call a single system. It is the job of the software analyst to get all of these different parts of the system corralled into a single programming framework. The framework organizes the creation of code, its versioning and testing, its deployment, and the sharing of code modules and objects between applications. This is no small task in an enterprise CMS.
- **Deciding which features of the CMS should be custom-coded** and which should be done using commercially available tools. For example, should the organization use the database connectivity code supplied by the CMS manufacturer or should they extend the code they have now, which works well with their ERP system?
- **Writing the specifications** and development plans for the programming tasks that require them (see Chapter 17, "Implementing the System," for more on specifications).
- **Creating estimates and plans for the staffing** needed to accomplish all of the programming to start and run the CMS.

As with the deployment analyst, there are people in the world who have done this sort of thing before. Still, someone who is a good general analyst but also has significant experience programming content-manipulation applications would serve you best. This is not a rare skill set, but it is not the most common. People of this ilk can be found in the back rooms of Web sites that use Perl extensively, or in the documentation groups of large companies that have had to use Structured Generalized Markup Language (SGML) to structure and manipulate massive document databases. In any case, stay away from the people who think that content can be treated just like data. They will underestimate the complexity of their task every time.

In addition, like the other analysts, this person cannot be a back-roomer. She must be able to negotiate among the competing development interests (which are no less, and often more, vehement than the business interests) and come to a workable compromise between competing platforms.

Finally, be careful not to choose someone who is more interested in writing code than getting other programmers organized. Note that the software analyst's job description does not say "writes a lot of programs." Even if you are in a position to have only one person who is both analyst and developer, because there are so many different places where code will live, it is essential that this person really be interested in staying organized, creating standards, and adhering to them rather than simply writing code.

Template and CMS developer

You will need entry- to mid-level programmers who have learned the templating language and associated programming tasks of a particular CMS. These people are responsible for the following:

- **Implementing publication templates,** given the specifications the software analyst and publication analyst have created. To do the implementation, the programmers use standard programming environments (such as JSP and ASP), the syntax of the CMS's templating language, and any custom objects that have been created by custom application developers.
- **Implementing input templates,** given the specifications the content analyst has created for content components and metadata. Most input templates are standard Web forms that are enhanced with custom objects supplied by the CMS product you use.
- **Doing the smaller programming-like tasks** associated with a CMS that are too technical for the CMS administrator, such as creating workflow scripts using an XML editor, or writing event triggers that fire when a new file is added to the system.

Custom application developer

You will need mid- to high-level programmers to create any custom functionality that you need added to a CMS. These programmers are generally responsible for the following sorts of tasks:

- **Programming to enhance collection,** including extensions to word processors, special tagging tools, and complete custom authoring environments when a standard tool will not suffice. In addition, programmers may be called upon to create other collection tools, such as automated acquisition routines and collaboration applications for authors to jointly work on content (see Chapter 31, "Building Collection Systems," for more information).
- **Programming for management,** including enhancements, extensions, and (unfortunately) replacements to the CMS's standard database environment, programs for automated review and archiving of content, and special indexing and retrieval routines.

- **Programming for publishing,** including enhancements, extensions, and replacements to the CMS's standard templating system, custom server objects that provide extra functionality for particular publications, and custom browser objects (for example, Java applets) to deliver functionality locally in a Web page.

The most successful custom extensions do not fully supersede the CMS; rather, they integrate closely into the CMS's core architecture. To be good at custom extensions, therefore, a programmer must be familiar enough with the chosen CMS to use its application programming interface (API) and be able to tie the new programs into the user interface (UI) of the CMS as needed. The kind of programmer you want in this position will not be afraid to go way beyond what the CMS can do out of the box, but will always try first to make the CMS do as much as possible without custom code.

Software integrator

You will need mid- to high-level programmers to write the code that glues the CMS to other systems. Like the custom application developers, these programmers will need to be versed in the API and other workings of the CMS. In addition, they will have to fully understand the other system and have skills in the appropriate language for connecting the two. Software integrators may be called upon to do the following:

- **Connect the CMS repository** and collection systems to other enterprise resources, such as the organization's user registry and the organization's ERP systems.
- **Connect the CMS's publications** to enterprise or outside resources. To create a full transaction system on the Web, for example, they may have to do a massive amount of integration between the CMS, inventory and fulfillment systems, financial and banking systems (inside and outside the organization), and product catalog systems.
- **Connect Web publications to the Web server.** The CMS may or may not come with the Web application server and Web database system that you intend to use. If it does not, then a programmer (with assistance from networking professionals) will have to make the connections either through configuration or through custom code.

Without a doubt, the main skill to look for here is proven expertise in the system or systems with which you want to integrate the CMS. If your integration is mainly on the Web, then experience with the target system *on the Web* is best.

Test analyst and test engineer

Quality assurance (QA) ensures that all parts of the CMS system work properly and that the publications that the CMS produces have a minimum of bugs. This is not an easy job. (If you have read through all of these job descriptions, you are no doubt tired of hearing those words). A test analyst is the one who sets up your test plans and coordinates the effort, and a test engineer executes the plans. Test analysts and test engineers must do the following:

- **Develop and execute test plans** against a collection and management system that is constantly changing as the CMS grows and evolves.
- **Work with each user's different configuration** and set of access rights to the system to be sure that the CMS works under all configurations.
- **Be able to test a collection of loosely coupled,** independently operating components (the CMS, that is).
- **Be able to ensure the quality of publications** that are created on the fly or are constantly being updated. They must figure out how you can test publications when you can never see them completely.

Clearly, this is no task for a beginner. The kind of testers you need are comfortable in an ambiguous situation where the test plan is never complete and you can never test all possible cases. Of course, many organizations hardly test their CMS at all and leave the publication testing to the publication team (in other words, the person who develops the page or the template, clicks around a bit, and calls it a day). If you choose this path, you get what you get.

If you choose instead to get as organized about testing as you would about collection, management, and publishing, then find someone who knows how to test code and its potential to create output rather than someone who must click though every page of a Web site each time it is produced to ensure that it is all right.

Publications Staff

The publication staff ensures that each publication is high quality, published on schedule, and makes the best use of the content in the CMS repository. Given the wide differences today between print publications groups and Web publications groups (and, for that matter, between the Internet and the intranet groups) in a large organization, you may very well start with a different group for each major publication. I imagine that as time goes on and the idea of content management takes hold, the various groups will coalesce into a single publications group (but don't hold your breath).

Publication analyst

The publication analyst is a graphic designer or other design professional who is responsible for designing the CMS publication system so that it can automatically create compelling publications.

The publication analyst is responsible for these general areas:

- **Choosing the design tools and environment for the publications.** Like the software analyst, the publication analyst is likely to come into a situation where groups already have chosen tools and are not interested in switching. The analyst must weigh these considerations against the advantages of standardizing tools and using tools that are most amenable to automation by the CMS. Especially in the print publication world, there can be a tremendous difference between tools in the amount of effort it takes to automate them.
- **Choosing which publications to include in the CMS and when.** The analyst must balance between the two extremes to which people will want to draw her. On one side, people will say, "This can all be produced automatically if we dumb it down a little." On the other side, people will say, "Our publication is too complex and our readers expect too much crafting for this to ever be produced automatically." It is the unenviable job of the publication analyst to find the compromise that allows as much as is technically feasible, esthetically advisable, and economically viable to be produced by the CMS.
- **Creating the general publication specifications** that detail how a publication will be created by the CMS.
- **Creating estimates and plans for the staffing** needed to accomplish all of the design needed to start and run the CMS.
- **Creating the overall plan for how content will be targeted** and scheduled for publication. Given that a particular content component may find its way into any number of publications, all on different publication schedules, this can be a difficult task.

The publication analyst is the prime example of the schizophrenia that many CMS staffers must embrace and overcome. The analyst must be firmly in favor of beautiful, compelling, well-crafted publications. The analyst must also understand and believe in the philosophy of automated publication. This publication analyst is the key negotiator between the specific needs and requirements of a particular publication and the generalized and neutralized content that is in the CMS repository. To be successful, this person must know how to work with all of the creative staff, who will want handcrafted pages, and the technical staff, who will want fully automated pages. The result must be a design that can be configured and automated but that still meets the qualitative design goals of the publication. Ideally, this person is someone who really can't decide if she likes art or science better but has practiced both. Finding such a person may not be easy. Following are some of the things to look for:

- **Someone with a strong background in at least two media.** Look for someone who has worked on the Web and in print, for example, or someone who has worked in film and on magazines. This ensures that the person is not "stuck" in one way of looking at the publication process.
- **Someone who is process-oriented.** A highly creative and energetic designer who depends more upon the drop of inspiration than the downpour of perspiration is the wrong choice. The inspiration must be there, but the job mostly concerns getting the publication process organized and systematized.
- **Someone who is not afraid of technology.** The best publication analyst is excited about the possibilities that new technologies can offer, but has a healthy cynicism about how soon or how much the technology will help.
- **Someone, like the software analyst, who knows how to negotiate** within design, within technology, and between the two. This person must be respected and be able to deal with others, whether the room is full of creative people, technologists, or is split between the two.

Publication designer

A publication analyst creates a publication framework, and a publication designer applies it. This relationship is the same as between the content analyst and the metator. In both cases, the analysts' grand plans succeed or fail based on the day-to-day work done by their less lofty associates. In both cases, the analyst is a more experienced and organizationally savvy (and more respected) version of their associate.

The publication designer is responsible for these tasks:

- **Creating the publication specifications** for how each particular publication will be produced by the CMS. These specifications include wire frame drawings, template designs, standard page and section elements, and the plain language version of the logic that will be needed to decide what goes where on the published page.
- **Working with the page developer** and CMS template programmer to ensure that the pages she designed can be coded.
- **Working with the content analyst** to ensure that the content that must get onto a page is segmented and tagged in such a way that it can be found and appropriately formatted.

The publication designer is a negotiator in the world of the template programmer, IA analyst, and page developer. She must find a fit between the best page design, the one that can be produced in the publication medium (Web page, magazine page, and so on), and the one that can be programmed in a publication template. Many of the global issues will have been worked out by the analyst before the designer is involved. But the devil is in the details, and, like the metator, the publication designer is in much closer contact with reality than the analyst.

You can find publication designers in the ranks of analyst "wannabes" and page developers at the top of their talents. Analyst wannabes are people who would like the job of analyst, but don't have quite the experience and maturity yet to fulfill the role. If you find yourself saying "She looks right for an analyst, but she's not quite there," you have found a good designer. You can find out more about the requirements for publication analysts in the preceding section.

Page developers at the top of their talents are people who are ready to move on to more challenging and abstract work than creating pages, but may never be cut out for a job like analyst. They are likely to be extremely skilled in a particular medium (a QuarkXPress guru, for example, or an HTML junkie) but have little interest in moving beyond that medium. They may be very smart and able to negotiate with their current peers, but not ready in the foreseeable future to play at the level of the wider organization. These people are doing superior work in their medium and could easily take on the added responsibility of entire publications. For more information on the job of page developers, read the next section.

Page developer

Someone has to take the design specification for a publication and turn it into actual template pages. I call this person the page developer. I choose the word developer here with a bit of trepidation. The page developer acts as a pseudo programmer. She doesn't actually write code, but she knows enough about it to recognize particular code modules, move them into and around on pages, and slightly modify them so they will work in a new context (changing their parameters, for example). True to CMS form, however, page developers also have an opposing skill: They are pseudo designers. They don't actually design the page, but they know enough about it to pay attention to a pixel's worth of detail, move design elements into and around on a page and make slight modifications to them so they will work in a new context (widening a table, for example, to fit a larger title). Page developers have the following responsibilities:

- **Turning the publication designer's specifications into files** that are usable by the CMS. A developer, for example, might take a drawing created by a designer in Macromedia FreeHand, illustrating layout and design elements, and turn it into an HTML file. Or she might create an Adobe InDesign file that includes all of the publication conventions that a particular publication requires.
- **Working the design to completion.** For example, a publication specification might call for a particular page to have ten variations. The developer actually creates those variations and ensures that they will all work. In this process, there is bound to be a fair amount of interaction with the publication designer.
- **Working with the CMS template developer to merge design with code.** At some point, the page developer and template developer must merge efforts and produce one file that has both design and programming in it. Anyone who has had to share the same file between two competing interests knows that this is always a pain. Nevertheless, the two developers must work hand-in-hand to bring a page to completion.

Page developers are less difficult to find than many of the other CMS staffers. In each major publication medium are large numbers of people qualified for this position who congregate in interest groups and in the publication groups of large organizations. The ones to focus on are the ones who can span technology and design and like to play intermediary between the two.

User interface specialist

This person knows user interface design for a particular publication type. In the print world, this job falls on the publication designer. In the Web world, the publication designer may or may not have the skills to complete the following necessary tasks:

- **Web form layout,** including the naming of controls and their layout on the page.
- **Web application design,** including the logic and presentation of the functionality of the application in a Web page.
- **Usability and page parsibility,** including designing and running user tests to see how understandable and navigable a page is.
- **Resolving issues associated with the integration of text,** media, graphic design, and branding with the application interface.

You may be able to cross-train a publication designer to meet these needs. If your content analyst or metator has worked as an IA on a large or complex Web site, she too, may have these skills. On the other hand, you may have to hire or contract-out this position. In any case, it is worth your time to understand that this is a real need and find some way to staff for it.

Content Processing Staff

Your content processing (or content conversion) staff designs and implements a content processing system for dealing with acquired content. Information that is destined for the CMS that needs to be cut into components, have its format converted, or have its structure altered flows through this group. The content processing staff analyzes the tasks needed to turn acquired information into content, designs tools to aid in the process, and trains and manages staffs of processors.

Conversion analyst

Someone must be responsible for the overall design and creation of the system you will use to change the raw information you acquire into finished content. I call that person the conversion analyst. Her responsibilities include the following:

- **Selecting or designing the tool set that the team will need.** Given the set of information inputs and content component outputs that the content analyst has specified, the conversion analyst will need to figure out what tools to buy or create to make the transformation as efficient as possible.
- **Creating the specifications for the programs** that tool developers will follow. For more information on these specifications, see Chapter 30, "Processing Content."
- **Designing the methods the team will use.** Given the inputs, outputs, and selected tool set, the analyst must determine the right set of methodologies to move information quickly through the system.
- **Designing the facilities and needed infrastructure** if the team is going to be large. For example, you might need to create a large work area where dozens of people can sit and easily interact.
- **Planning and estimating the staffing levels and skills** that you will need to process the expected throughput of information. It can be difficult to accomplish this task if (as often happens) there will be wide swings in the amount of information that will be received per week.

This job is the analyst companion to the production manager job I described earlier. Whereas the manager is responsible for the day-to-day operations of the processing group, the analyst is responsible for understanding the situation and getting the system set up. A good candidate for this position deeply understands text formats such as HTML, XML, PDF, and SGML. They must be very analytical and meticulous by nature. Because this person must write content processing specifications, she must be familiar with, if not well versed in, programming. Because she has to create procedures, she also must be familiar with, if not well experienced in, process management and the techniques used by data processing facilities.

Data processing facilities are one good place to look for people with this skill. Wherever teams of people have been brought together to make their way through masses of semi- structured or unstructured information, you may find people who have the analytical skills and experience to be a conversion analyst. In my experience, I have seen such people in the back rooms of CD-ROM companies that publish huge catalogs, as well as in large law firms that must process grand mounds of court documents.

Tool developer

A tool developer creates the programs that automate the conversion process. These people use whatever text formats and programming tools they need to make processes faster, cheaper, or higher quality. Tool developers create the following:

- **Programs that convert information** from one binary format to another.
- **Programs that create components** by crunching through files and directories, automatically separating and storing individual chunks.

- **Programs that create elements** within components by scanning the component content and inserting tags that separate one element from another.
- **Programs that infer the appropriate metadata** to add to content and automatically insert it.
- **Programs that present choices** to content processors and automatically carry out the result. These programs save the staff hours by doing all of the manual labor and presenting them with only simple choices (simple by human standards, but obviously too hard for the program to make).

Tool developers are strong programmers. But the kind of programmer you want for this position takes a sort of guerrilla attitude toward her work. In other words, she enjoys diving into a problem, applying a rapid response at just the right place to make a big impact, and then retreating to watch the effect. This attitude allows the tool developer to materially affect production processes when the problem occurs, rather than spending a lot of time automating things that are not worth the effort, or stopping ongoing production for too long a time while she hangs back and considers the best approach.

Content processor

I use the title content processor for the people who come in every day to move information through the processes designed by the analyst. Content processors do the following:

- **Check the results of automated processes** to ensure that they worked as expected.
- **Perform manual processes** where automation cannot be applied or where it has not yet been employed.
- **Add metadata to masses** of content following prescribed rules.

You might be tempted to describe the work of a content processor as repetitive and rote—like a line worker at a factory. This is true in the sense that they work on parts moving through a larger process and perform the same function over and over. In a more telling sense, it is not true. For if the procedure a content processor is doing is repetitive and rote, you should be able to get the tool developer to write a program to do it. In fact, the hallmark of a good processor is someone who has the patience to apply good judgment in a consistent way without losing focus or ceasing to care (qualities that forever bar me from the ranks of the processing elite).

On the other hand, these jobs do not require a lot of experience and only rarely require any subject matter expertise. That's good, because you can't afford to pay too much for these positions, especially if you need a lot of staff quickly. It's also bad, because the person best suited to this position is not your run-of-the-mill temporary placement staffer. You need to find people who are both conscientious and underskilled, wise and underpaid—yet another example of the contrary qualities of CMS staff.

Content QA specialist

Quality assurance is a mandatory part of content processing. You need people whose main (or at least most important) job is to go over the work of automation routines or other people and assess how close it is to the quality targets you have set. Generally, these people can be drawn from the upper ranks of the content processing staff. You can usually spot the high-accuracy people pretty quickly and give them the task of checking the work of others.

To find good QA people, look for graduates from the ranks of the content processors.

Content Creation Staff

Content creation is the process of inventing new text, pictures, sound, or other media. I won't attempt to catalog all of the jobs that might be involved in creating content. Rather, here are brief descriptions of some of the creation jobs that are most central to a CMS.

Acquisitions manager

An acquisitions manager handles your relationship with all content acquisition sources. This person does the following:

- **Negotiates and forms an agreement** with the source. The agreement includes quantity and schedule terms and may also include payment if the source is not free.
- **Keeps abreast of the deliveries** from the source and is the point of contact for any discussions of delay or changes to the quantity or quality of the source.

This manager should be skilled at project management and negotiations. She should know enough about electronic content formats and publishing processes to be able to understand the agreements she is forming.

Traffic cop

There can be a dizzying number of separate pieces of content flying around when a CMS is in full swing. You may very well need a person to coordinate and keep all of this information straight. I call this person the traffic cop. Her main responsibilities are as follows:

- **Overseeing the delivery** of content from authors and acquisition sources.
- **Keeping abreast of the CMS workflows** and ensuring that everyone is on schedule.

- **Resolving any bottlenecks** that may be affecting the flow of information.
- **Resolving any disputes** about who is responsible for what.

A good cop has experience in large-scale task management. She is not afraid to pressure people to meet commitments but knows how to do so in a way that does not alienate contributors.

Writers and other content creators

Writers, illustrators, sound artists, videographers, animators, and the like (artists, for lack of a better word) create content for you from scratch. Aside from knowing their own trade well enough to ensure that you are getting the best product from them, they must also be able to understand and embrace the abstractness of the CMS process. For professionals, this ought to present little problem. Most have had experience being hired to create something whose end form they never see until it is finished. For some, especially non-professional artists, it takes some cajoling to get them to pull away from the final product and just create content.

Artists with whom you are likely to have the least problems have the following characteristics:

- **They are used to creating for a purpose.** Writers who have written for a particular marketing purpose or graphic artists who have done illustration for magazine articles are examples.
- **They are used to creating for multiple audiences.** If a candidate does not ask you about audiences in the first few minutes, beware. If they wrinkle their noses at the list of your audiences, stay away.
- **They have created for more than one medium.** For example, people who have only done graphics for the Web may not understand the different constraints of print. People who have only written for print may not grasp the right style for the Web.
- **They don't mind working on a team.** Artists who want to hold on to their work for as long as possible and be its shepherd through every stage of the process will be trouble in a distributed and multistage process.

Editor

Most of the same caveats that I gave you for writers apply equally well to editors. The problem of the editor, relative to content creators, reminds me of the following quip:

> "If you think Fred Astaire was good, consider Ginger Rogers. She was doing all the same stuff backward!"

A good editor knows how to keep in synch with the creator and gently push her in the right direction. The problem is compounded by the fact that a single editor may be responsible for many creators and many types of content (text, pictures, sound, and motion). Professional editors are no strangers to this situation and are generally able to roll with the waves. Still, in the context of a CMS—where the content in not only does not look like what comes out, but must come out in a variety of forms, juxtaposed with who knows what other content—the editor must be able to think extra-abstractly about the content she reviews.

In addition, editors who are tied to particular methods (for example, being able to make changes to the text in the final publication rather than in the copy that is stored in the repository) will have to change their ways even if the new ways are not as foolproof as the older ways. Clearly, an editor who is comfortable only with pencil marks on a piece of paper would be a poor choice for your CMS staff.

Summary

There are a lot of potential positions in a CMS project. You may not need one person (or more) in each position, but you will need someone to cover each of the major roles in a CMS:

- CMS types tend to have multiple, sometimes conflicting, skills.
- Content managers and other types of managers direct and oversee the construction and maintenance of the system.
- Business analysts figure out how to fit the CMS into the organization.
- Information architects figure out the structure of the content you will manage and help out, tagging content as it comes into the system.
- Software developers write the programming code behind CMS templates and integrations.
- The publications staff designs and creates the publications that the CMS will produce.
- The content processing staff creates and runs conversion processes.
- The content creation staff writes the text and creates the media that the CMS manages.

In the chapters that follow, I discuss how you can use all of these wonderful people to design and build a CMS.

✦ ✦ ✦

CHAPTER 12

Working within the Organization

Each department in an organization has something to add to the design or implementation of a CMS, but none of the traditional groups has the inherent right to own the system. Because the job of a CMS is so closely aligned with the job of the organization, to do its job, a CMS must be integrated fully into an organization.

In this chapter, I'll discuss how a CMS fits into a stereotypical large organization.

◆ ◆ ◆ ◆

In This Chapter

The overlap between a CMS and the goals of the organization

How you can harvest the flow of information in your organization

What each major group has to add to a CMS

Some ways you can organize a CMS effort across the enterprise

How to get the scalability you may eventually need

A few of the many hurdles you might face in getting a CMS initiative going

◆ ◆ ◆ ◆

Content Management and the Organization

The purpose of any organization is to create and offer something of value to the world. Commercial organizations produce valuable goods to sell. Government organizations produce laws and regulations for distribution. Nonprofit organizations produce goods and information that promote a particular social concern.

The purpose of a CMS is to help organizations create and offer valuable content and functionality. For commercial organizations, the content and functionality aid in the sale of goods; for government organizations, the content and functionality aid in the production and distribution of laws and regulations; and for nonprofit organizations, the content and functionality support a social concern.

Because the task of a CMS is so closely aligned with the job of the organization, to do its job, a CMS must be integrated fully into an organization. Of course, a CMS might serve a small part of an organization. Many organizations are diverse and have a variety of types of value to create and offer. In these cases, it might be necessary or preferable for the CMS to be designed to serve a sub-organization. Regardless of the size or scope of the organization that the CMS serves, its purpose is so intertwined with the purpose of the organization that the more integrated it becomes, the more able it is to serve the goals of the organization.

A CMS becomes integrated into the organization in the following ways:

- It organizes many departments and job types into an overall system.
- It harvests information from throughout the organization.
- It unifies the organization's communications to itself (an intranet site, for example) and the outside world (an Internet site, for example).
- It ties into the organization's existing information management infrastructure.

Quite often, the need for a CMS is felt most acutely by one part of the organization. The group that feels the pain organizes a team to confront the problem. In my travels, I have seen these originating groups:

- **A business unit** (or department) that needs a much fuller Web site than it now has. The business unit has diverse content and core functionality that it needs to bring to its audiences.
- **The organization's corporate communications** or editorial group that finds it cannot keep up with the explosive growth of content that must be created and delivered.
- **The organization's IT group** that needs to unify all of the divergent requests it receives for Web pages into a single, manageable system.
- **The organization's marketing department** that wants to both unify the organization's appearance to the world and better target market segments.

If all goes well, the originating group brings in all other groups and they coalesce into a tight team that serves all the groups' needs. More often than not, however, this does not happen. Rather, the originating group avoids the hassle of dealing with other groups' budgets and politics and decides to go it alone. The result is, at best, inefficiency and redundant efforts. At worst, the result is massive infighting and paralysis while the organization decides who owns the effort.

I have seen organizations waste a tremendous amount of time and effort sorting out ownership of the new function of content management. The problem is that each group has a fair claim to ownership based on the part of the problem that they own. The business unit owns the content, the editorial group owns the management process, the marketing group owns the recipients of the content and the messages they receive, and the IT group owns the computer system that makes it all happen. Any group on its own can only duplicate the expertise already in another group. If a member group does not have the experience or skill to participate, then the group needs to be built up, not avoided.

The deadly forces of bad interpersonal dynamics and poor communication notwithstanding, the solution is obvious. Each group needs to understand the contribution of the others and make the best use of it. If the CMS effort is to remain in a single group, then that group must reach out to the others to be sure they are included. If the effort is to lie outside of any group (a solution I favor), then the purpose of the outside group is to marshal the resources of the other groups toward the goal of easily collected, well-managed, and fully deployed content.

For good or ill, in order to do its job, a CMS must integrate and unify the organization (see Figure 12-1).

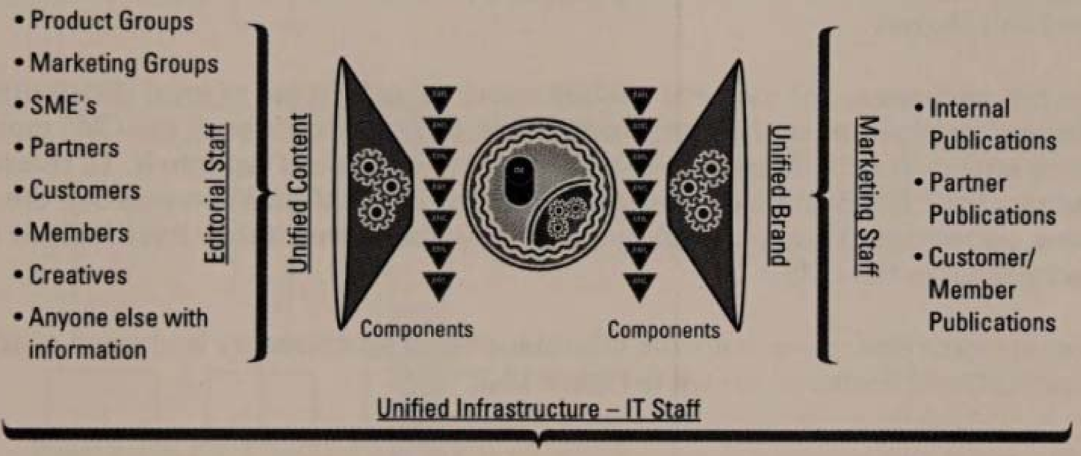

• Product Groups
• Marketing Groups
• SME's
• Partners
• Customers
• Members
• Creatives
• Anyone else with information
Editorial Staff
Unified Content
Components
Components
Unified Brand
Marketing Staff
• Internal Publications
• Partner Publications
• Customer/ Member Publications
Unified Infrastructure – IT Staff

Figure 12-1: To do its job, a CMS must integrate many of the parts of an organization.

To serve the whole enterprise, the CMS must accept a wide variety of information from a cross-section of the organization. To crunch all of this disparate information into a single system requires that you develop a single editorial approach to the organization's information (preferably with a unified editorial staff). If you are lucky, the organization will embrace this approach and will eventually create information in a unified way. If you are unlucky, you will be converting and reworking content for the indefinite future.

To allow information to flow into the CMS from throughout the organization and then from the CMS out to a variety of delivery platforms requires a unified infrastructure. CMS systems generally assume you will have a unified Web infrastructure for collecting and distributing information. Whatever infrastructure you create may be one of the few that spans the whole organization. If you are lucky, the infrastructure will be embraced and maintained by your existing IT group.

To allow diverse information sources to coexist in the same publications requires that you create a single brand (or small family of related brands) for your organization. Broadly speaking, a CMS must integrate and unify marketing approaches into a single approach with many channels. If you are lucky, your organization will be firmly behind this effort and will see it as a great step forward. If you are unlucky, you will be stuck trying to unify groups that would rather go it alone.

The challenge of a CMS is to be able to unify the organization enough that it is possible to have a single system collect, manage, and publish its information.

Tracking Information Flow in the Organization

All organizations create content. To create a successful CMS, you must be able to separate the content that has value beyond the context in which it was created from the content that serves one specific purpose and does not need to be managed and shared.

It is not uncommon for valuable content never to make it out of local distribution within a small segment of the organization. In order to do its best, the CMS must reach into each area where valuable information exists and tap into it. To reach in and tap, the CMS must be designed to target the right information sources, creators, and accumulators, and easily be used by them even if they have little or no background in the CMS.

In a typical organization, valuable information and functionality is created at all organizational levels, as shown in Figure 12-2.

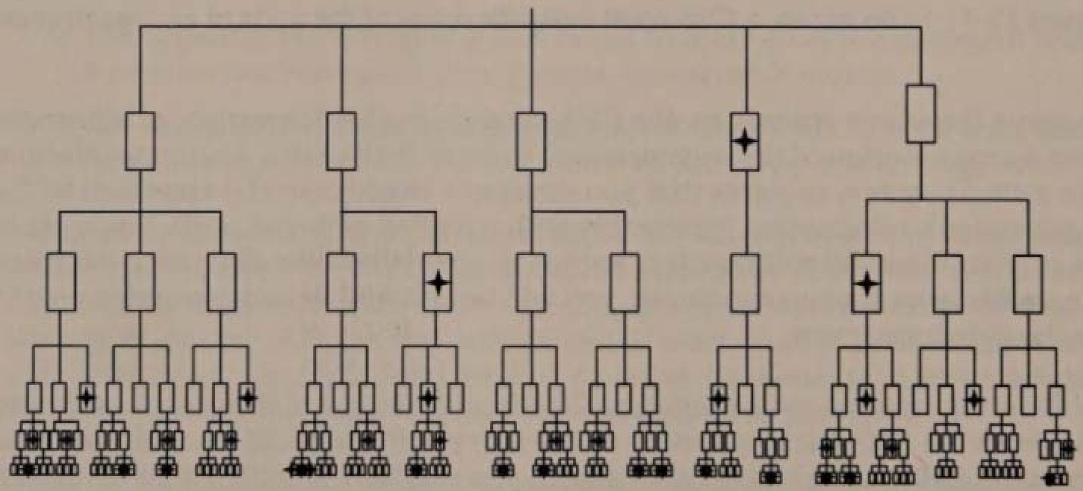


Figure 12-2: Valuable information and functionality (represented by the stars) is created throughout the organization.

After it's created, information can flow within the organization in a number of ways.

Some information is created by an individual, who then passes it on to others higher up in the organization in the form of reports, analyses, presentations, and recommendations. The creator of this sort of information usually thinks of it as meeting a particular deliverable for a particular person, as shown in Figure 12-3. Often, the creator does not consider her information to be valuable beyond its original purpose (whether or not this is true).

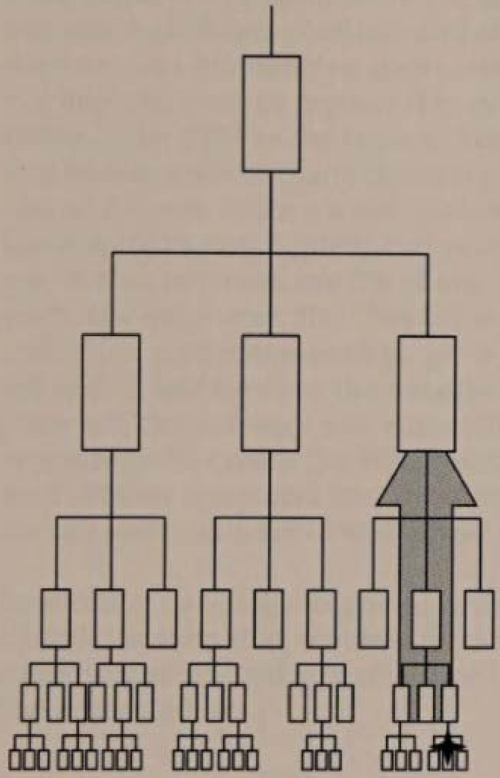


Figure 12-3: Some information flows upward in the organization.

Other times, an individual creates information and distributes it to her team. Information such as plans, policies, new initiatives, and background is developed to give context to employees, and often this information is not considered by the creator to be generally useful (whether or not this is true), as shown in Figure 12-4.

Finally, some information is created by an individual who then distributes it to peers. Information such as discipline-specific analyses, tips and tricks, and training resources is developed and distributed to help people of a certain type do their job more effectively, as shown in Figure 12-5. Once again, this information often is not considered generally useful but may indeed be.

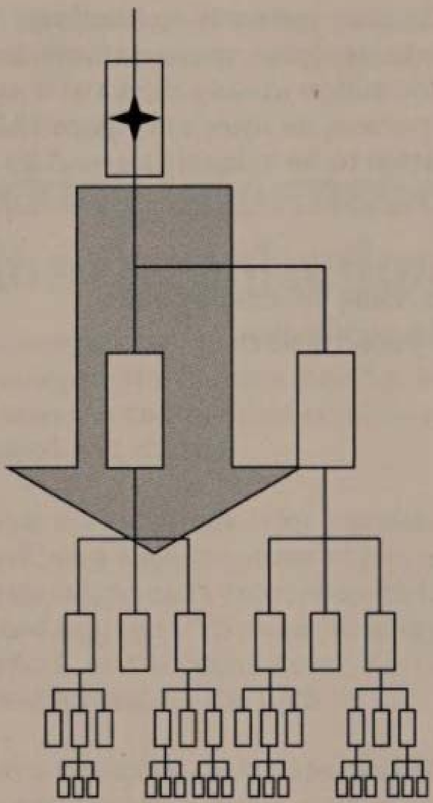


Figure 12-4: Some information flows downward in the organization.

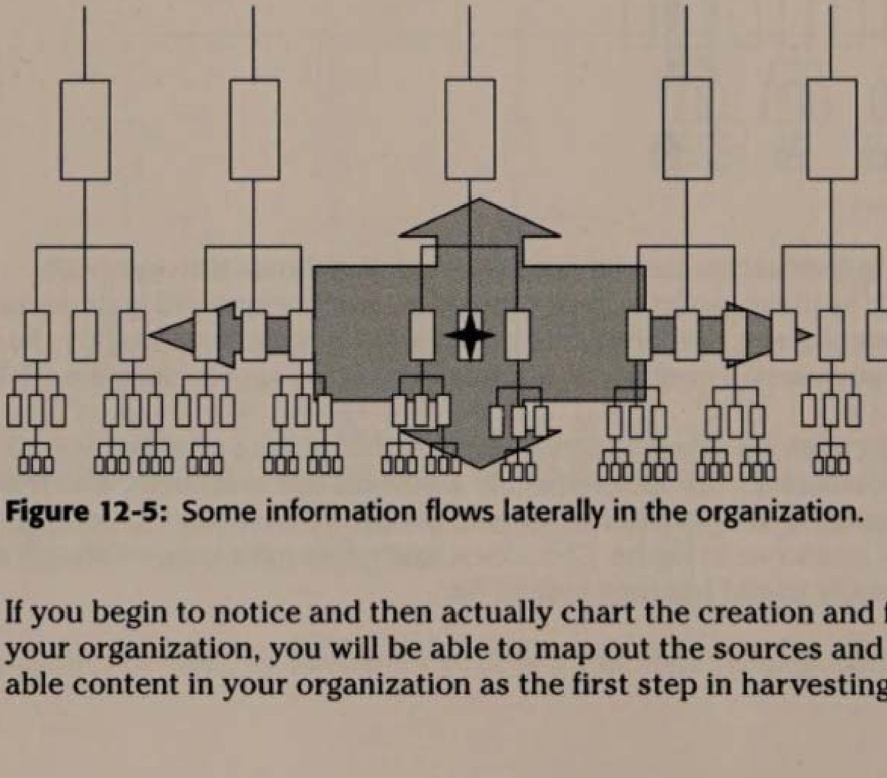


Figure 12-5: Some information flows laterally in the organization.

If you begin to notice and then actually chart the creation and flow of information in your organization, you will be able to map out the sources and reservoirs of valuable content in your organization as the first step in harvesting it.

All organizations have systems that capture, store, and give access to data. A CMS rarely replaces these data systems. Rather, it brings the same degree of management to content that the organization is accustomed to applying to data. This is not to say that there is no interaction. Especially in the publication system, there is likely to be significant interaction.

Consider a typical staff management data system. Such a system has data about each employee or member of the organization, including name, ID, address, compensation package, position, and so on. The human resources and administration departments are likely to access the system via a specialized application. You are not likely to want to replace this system with a CMS. You may, however, want to connect the CMS to the system. For example, you might want to create a printed and Web-based company directory. You can base such a directory off the data system as follows: Once a week, software that is part of the CMS repository contacts the employee data system and retrieves data records for new or changed employees. It also retrieves the IDs of any employees that have left the organization. For each new employee, the CMS software adds a work item. The work item tells a CMS collection staff member to go get a picture and a memorable quote for the new people and to add them to the data that has already been retrieved. For employees that have left, the software can automatically delete their information from the CMS repository. To create the Web-based and printed directories, the CMS publication staff creates templates that draw the employee information out of the repository and format it as a set of Web pages and as a book.

In addition to being integrated with other data systems, the CMS itself is an organizational system that requires the same sort of administration and maintenance as other organizational systems. The CMS is one in the overall set of systems that run the organization.

Understanding your information

People will not naturally seek out the CMS and contribute to it. Rather, the CMS team must seek out contributors and go to them. To begin, you must define what makes information valuable to your organization. Some of this valuable information comes from outside the organization (industry news, for example) and some comes from within the organization (case studies, for example). For the information originating inside the organization, you must discover the following:

- Where in the organization is it created?
- What particular people or job descriptions create the information?
- What is its initial purpose?
- To whom is it distributed?

- Are there places (directory locations or file cabinets, for example) where the information naturally collects?
- Are there particular people or job descriptions that collect and save this information out of interest or job function?
- In what formats is it produced?

The main obstacles to harvesting information from the organization are finding it, understanding how to remove it from its original context, and any adverse attitudes in the content creators.

Understanding your functionality

All organizations create and distribute functionality (even if only internally and on paper). Just as the CMS must reach into the organization to tap into the valuable content that exists in the organization, so the CMS must reach into the organization to tap into the valuable functionality (like the functionality that lets someone update their own HR records, or the functionality that allows customers to see their past transactions). And just as with content, you first should learn about your functionality and then plan for its harvest. The main obstacles to harvesting functionality are making it suitable for electronic distribution, tying into existing data management processes, and any adverse attitudes in your programmers and administrators.

Functionality, like content, might exist throughout the organization. To understand the functionality in your organization, you might start with these questions:

- Where in the organization is it created and maintained?
- Is it suitable for being segmented and delivered electronically on a Web site? If not, what would need to be added or changed?
- Who is responsible for programming the functionality? Was it developed in-house or is it part of a commercial package?
- Who is responsible for managing the data behind the functionality?
- Can it be accessed by code executed on a Web server?
- Who is the current audience for this functionality?
- What knowledge or experience is required for someone to use the functionality?
- Is there other functionality to which the functionality is related or on which it depends?

Understanding Organizational Roles

Each of the major groups in your organization has something different to add to a CMS initiative, as the following list indicates:

- **Business units** create the valuable content and functionality that the organization must deliver. These groups are, or should be, designed to produce important information effectively.
- **Editorial teams** should aim toward creating unified content across the organization.
- **Marketing teams** should create a definitive audience analysis and unified messaging in all publications.
- **The IT group** should implement and maintain the CMS infrastructure and, possibly, be responsible for functionality.

Business units generate value

The valuable content and functionality that the organization delivers comes from business units. These groups are, or should be, designed to produce important information effectively. In addition, business functionality (ordering, getting support, making deals, and so on) originates and terminates in a business group. Business units should rightly be expected to create the value that the organization needs to deliver.

On the other hand, business units should not necessarily be expected to know how to gather, target, or deliver this value. Often, the valuable information the unit produces remains lodged on some individual's computer and never sees wide distribution. In addition, much of the valuable information is produced for an internal audience. Product specifications, for example, are extremely valuable information that is often inaccessible to the outside world because of their style or format.

Editorial teams unify content

All organizations create some sort of written communication. Organizations employ technical writers to create product documentation, marketing writers to create brochures and press releases, and subject matter experts to create white papers and other background documents. Beyond these writers (and other content producers, such as graphic artists) is a group of editors who ensure that the output of the various writers is aligned and consistent. Consistency is the currency of content management. Without a strictly consistent approach to content structuring and tagging, the CMS cannot manage or publish content effectively. If your organization has editors, they should mediate between the business units and the central repository. Editors within the business units forge consistency within the content of the units. Editors outside any business unit forge consistency among all of the organization's content. When you implement a CMS, these two types of editors should form a common understanding and a common set of rules to make the editorial processes flow smoothly between the business units and the larger CMS effort.

On the other hand, editorial organizations rarely have been called upon to perform the very granular metadata tagging needed by a CMS. Thus, editorial skills in the organization must be augmented by the metatorial skills needed to develop and apply the level of metadata consistency needed by a CMS.

Marketing teams direct and unify publications

An organization's marketing group is responsible for understanding and controlling the way the organization presents itself to the outside world. Two of the main ways that marketers present the organization is through brand and messaging. The brand is the identity of the organization. It is a set of traits and associations that encompass how the organization wants to be seen by the world. Messaging consists of the small set of clear messages that the organization wants to deliver to the world (for example, "we are the place for information on the third world," or "buy our cars and you will have lots of admirers"). Because of this responsibility, the marketing group is often in control of all the organization's communication, including its Web site and other publications. What the marketing group brings to these publications is clear, directed communication about who the organization is and what they most want the world to know. In terms of content management, the marketing group has an even more central role — they know the audiences. The marketing group knows what kind of people the organization is trying to serve and what they want. This is the key knowledge that enables the organization to identify and target valuable content and functionality.

On the other hand, the marketing group often knows little about the technology and process behind designing and building a CMS. The analysis and development skills needed to create a CMS are often beyond the desire and ability of a marketing group.

IT groups build and maintain infrastructure

An organization's IT group keeps the computer hardware and software running and performing. Because a CMS is a set of interlocking software and hardware components, the role of the IT group is clear — to install and administer the system. In addition, the IT group has the knowledge and experience (or should have) to integrate the CMS into the array of other organizational systems (catalog and staff data systems, for example) that the CMS needs in order to provide content and, especially, functionality. In one sense, the CMS is just another enterprise information system and can be maintained by the same group and in the same way that other systems are.

On the other hand, just as you would not ask an IT professional to enter invoices into the organization's accounts receivable system, so you would not ask her to create content or publications. The natural place for the IT group is not in the content but in the system behind the content.

The groups I have discussed, as well as others, are not candidates for sole and exclusive ownership of the CMS. They *are* integral players in the overall success of the CMS effort and each has a different contribution.

Exploring Organizational Models

For simplicity, I present content management systems as single systems. Although it is surely most efficient to have a single system throughout the organization, it is by no means mandatory. If a single, monolithic CMS represents too much of a hurdle too soon for your organization, you can create a more or less loosely coupled system of relatively independent but interacting systems.

A Good Idea Well Presented

One of the first things I ask new clients is, "Can a good idea that is well presented and well defended be adopted in your organization?" I've received many strange looks when I've asked this question and more than a few non-responses from people who would rather not openly discuss the issue. But overall, this question has led to some of the best organizational-dynamics conversations I have ever had.

The point of the question is to understand the mechanism by which the organization adopts a new idea. In the ideal world, the governors of the organization would forthrightly discuss the merits of the idea, weighing every pro and con, and would adopt it or deny it based on its merits alone. Conversely, in the most cynical world, the merit of the idea would not count for anything. Adoption or denial would be granted solely on the basis of how the idea would help or hinder the personal agendas of the individual governors. If it would help enough powerful people, the idea would be adopted. Those whom it would ignore or even subvert would oppose the idea to whatever extent they could.

The real world is somewhere between those two extremes. Just because a CMS is a good idea, it does not mean that it will be adopted. On the other hand, just because the CEO favors a CMS, it does not mean that it *should* be adopted. You need to be sure that you have a good idea, that it is well presented, and that you can defend it against all reasoned attacks. In addition, you need to have an idea that has political support and can be defended against interpersonal attacks.

If, in the end, good ideas win out in your organization, you are pretty safe. Content management is a good idea.

It's never a good idea to force a CMS on the organization. You are too much in need of the organization's active and creative participation ever to risk that sort of alienation. Rather, you need to figure out how the CMS can bring value to the groups it will encompass. You need to be clearly convinced that the CMS brings value to a group greater than the cost they will bear. Then you can present your case to the group and hope for the best. Be prepared for groups not to be convinced immediately. They may have a lot of personal and financial investment in the way they do things now. Think as objectively as possible. Is the problem that they just don't want to change? Is it that you have not presented your case well? Or do you just not have a compelling case?

For any of these reasons, you may need to fall back to a looser connection between the CMS and a particular group. There are a number of ways to loosen the connection.

Collection variations

On the collection side, the "normal" way to proceed is by having contributors submit content directly to the central CMS repository (see Figure 12-6).

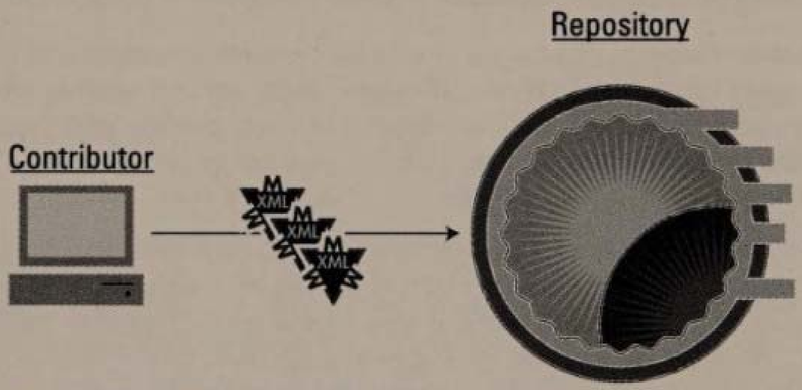


Figure 12-6: The usual collection mode

It is not always possible, however, to immediately encompass every group's collection process. Suppose that a group is already creating a set of print publications that you will not soon be able to duplicate with your CMS. Still, you need the content in those print publications for the Web site that the CMS will produce. Rather than taking on a task you know you can't do, or worse, forcing them to "dumb-down" their publications so you can create them, you can get the group to behave like a syndication source, as shown in Figure 12-7.

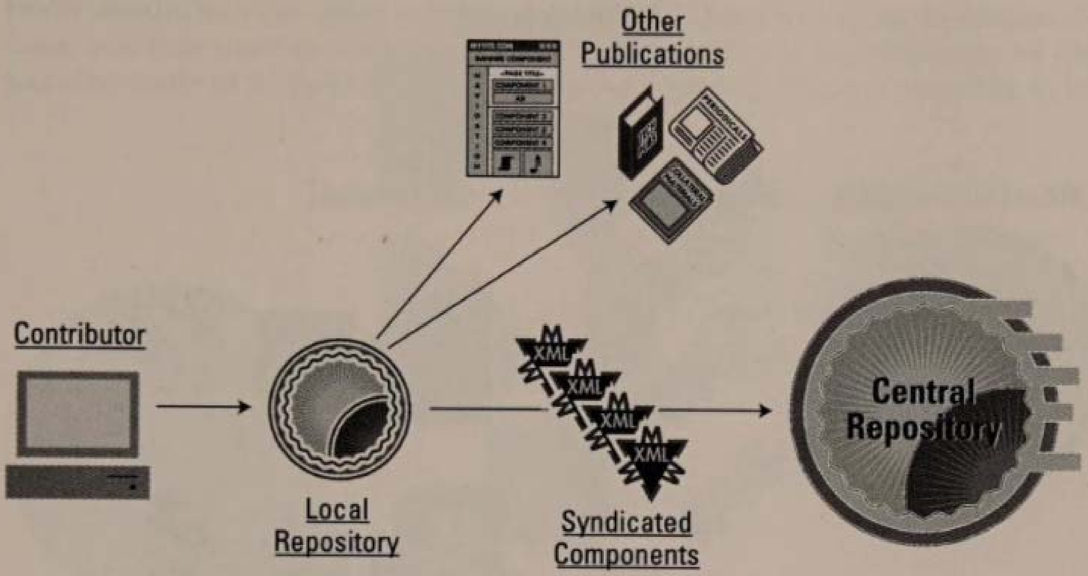


Figure 12-7: You can treat other groups in your organization as syndication sources.

Your main task in this scenario is to get the other group to begin to think of its own system as a CMS. If they can figure out how to separate their own content from its publications, they can figure out how to give clean input to your CMS. Even if they never fully automate their process, you still will have accomplished two major tasks:

- The group will be on the road to becoming fully part of the CMS. As they get better at separating their content from their own publications, they will make their own compromises in their publications, and be learning the art of CM along the way.
- You will get a good content source without having to participate in its creation. Given that you can get the group to give you good stuff (valid XML, for example), you will have all you need to easily incorporate their content in your repository.

What you lose in this approach is time and control of the master source. You probably will not receive content from this group until it has already been released in their publications. In addition, can you be sure that they will not make significant changes to the content after they deliver it to you?

Publishing variations

On the publishing side, the usual way to proceed is to have the CMS create all of the publications, as illustrated in Figure 12-8.

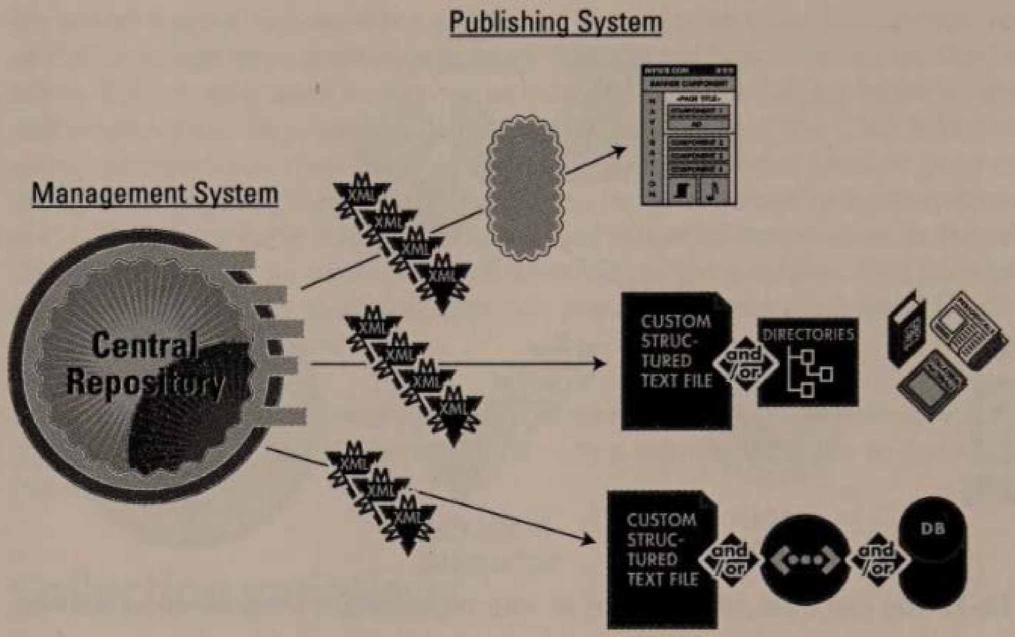

Publishing System
Management System
Central Repository
XML
CUSTOM STRUCTURED TEXT FILE
and/or
DIRECTORIES
CUSTOM STRUCTURED TEXT FILE
and/or
and/or
DB

Figure 12-8: The usual case, where the CMS creates all publications

But it is not always practical or advisable to have the CMS control all publications. Suppose that you have a group that has already set up a dynamic Web presence with a set of content databases and a complete Web infrastructure. Rather than migrating them to the CMS publication system and Web platform, you can simply fill their databases full of content from your repository (see Figure 12-9).

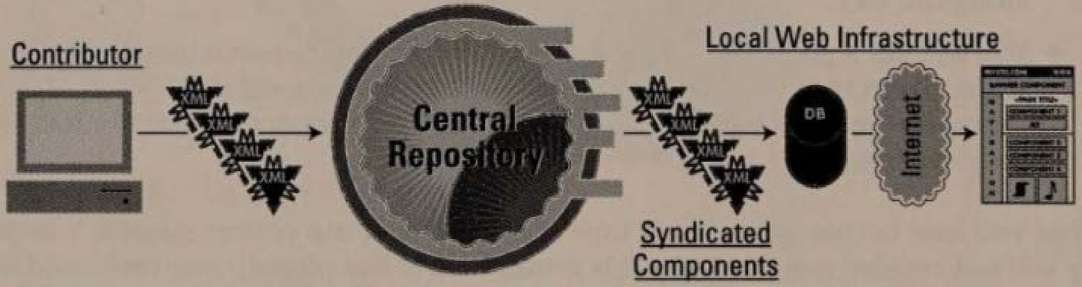

Contributor
XML
Central Repository
Local Web Infrastructure
DB
Internet
Syndicated Components

Figure 12-9: Using the tools you have for syndication, you can fill a Web database with content.

Using this method, you can embrace all of the collection and management of content destined for the site without disrupting the site itself. In effect, you are syndicating content to the Web database. Although this may require some programming on your part, it will probably be a lot less work than re-creating the entire site from your CMS.

Of course, you may want to re-create some of the site. For example, maybe the site design is out of line with the other publications you are creating, or maybe the site really should include other content that it is not currently set up to handle. In this case, you may want to consider having the CMS produce not only data for the site, but also some of its page templates and even some of its pages (see Figure 12-10).

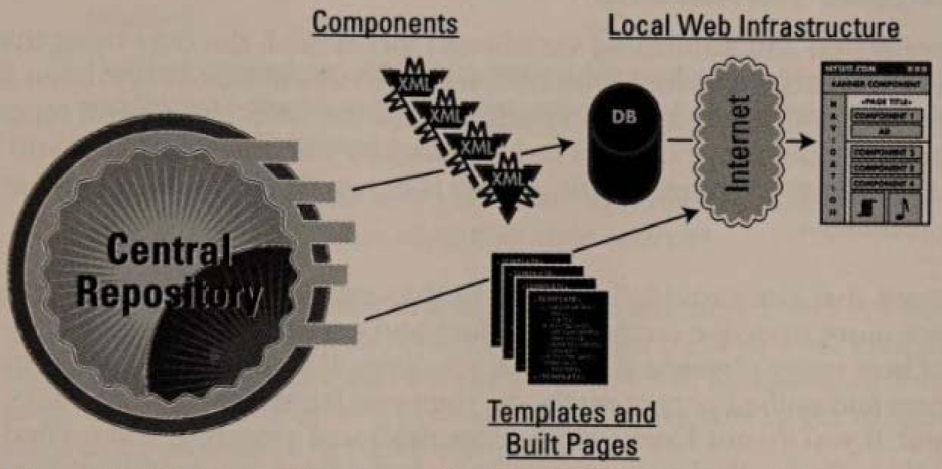

Components
Local Web Infrastructure
XML
XML
XML
XML
DB
Internet
Central Repository
Templates and Built Pages

Figure 12-10: In addition to syndicating data, you can supply page templates and even complete pages to an external system.

This scenario is a bit tricky. To produce page templates for an existing Web site, you have to know a lot about the code and conventions of that site. In effect, your CMS would be managing much of the publishing of the external site. To produce pages, you have to make sure that they fit into the site. Both the look and feel as well as the navigation must match the way the site is constructed. Of course, if you are producing templates for the site, both of these variables are already in your control.

Finally, you can simply treat any system external to the CMS as a syndication target, as shown in Figure 12-11.

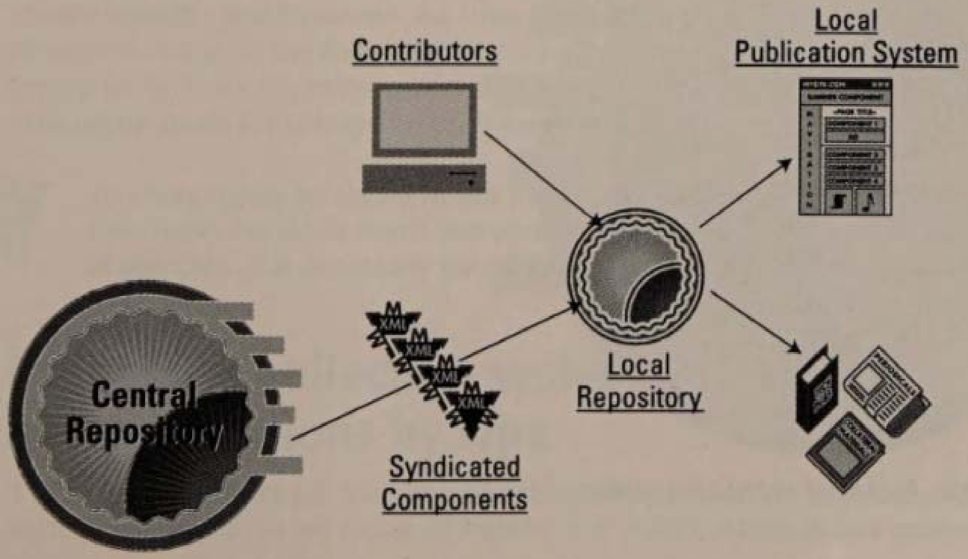

Contributors
Local Publication System
Central Repository
XML
XML
XML
XML
Local Repository
Syndicated Components

Figure 12-11: You can always just syndicate to an external system.

The external system controls its own publications and may have other content that you do not control.

Management variations

In all of the collection and publishing variations I mentioned, the only thing that was absolutely under the control of the CMS was the central repository. Even in the case of the external collection system syndicating to the CMS, the central repository can contain the definitive content to be shared by the organization. If you back off this, and the definitive content to be shared is in two or more places, you are biting off some problems.

It is not so much that you shouldn't have more than one database, but that you shouldn't have more than one content structure and access methodology. Regardless of how many physical databases you have, if you have a single organizational structure and unified access methods, then you have a single repository. On the other hand, if you do not have a single organizational structure and unified access, even if you have all of your content in a single database, you do not have a single repository.

That said, if you can manage to have a single database for all your content, you will spare yourself the unenviable task of distributed data management. At the very least, you must have a single place where you can tell people to go to get the blessed version of a particular kind of content.

Optimally, you want to create a central repository that really unifies your content and content access (see Figure 12-12).

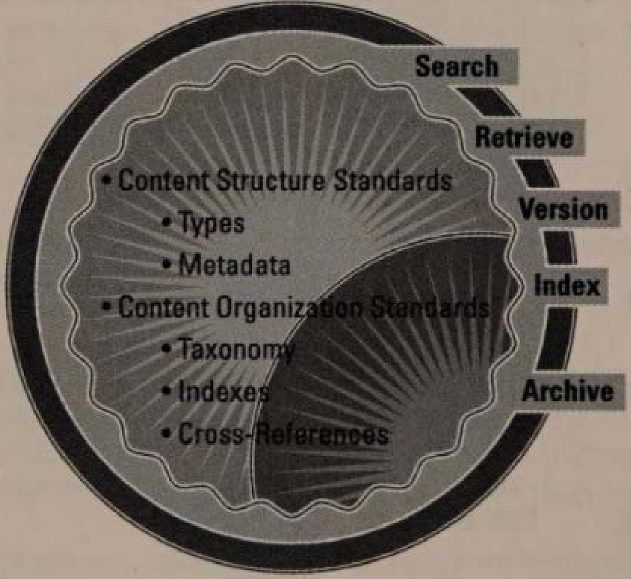


Figure 12-12: A central repository unites content structure and access.

To function as a central repository, your system must extend the following qualities over all of your content:

- **Unified content structure.** The way content is chunked and tagged (your components and their elements) needs to be standardized across all data sources.
- **Unified organization.** The hierarchies and other organizational schemes you use to categorize and get to your content need to extend to any place where the content is stored.
- **Unified access.** The way you query and make use of the content you retrieve needs to be the same across all data sources.

My best advice concerning management variations is to try not to vary too much. The headaches you will have — trying to keep track of what content is in what structure, organization, and access system — will be unending. You will also inevitably find yourself in a position of not being able to get to the content you need in a publication or particular page.

Using Functional Collection and Publishing

By far, your best bet is to have a single central repository and management system that extends across all of your needs. That's not the case with collection and publication systems. As long as all efforts use the same small set of interfaces to the management system, you should be free to allow contributors and publishers to do their own thing — within limits.

In fact, for a system with real enterprise reach, you need to release control of specific collection and publication activities and focus instead on serving general collection and publication needs. At first, you are most likely to create the specific collection systems that gather the content you have identified. Likewise, you will create specific publications. As time goes on, however, other content and publications will come to the fore that you also need to include. Rather than continually trying to do the collection and publication for the organization, you can start providing the tools for groups to do it for themselves.

Note

You may rightly be saying at this point that it will be hard enough to do this in your own team, let alone teach everyone else. This will be true for the first few versions of the CMS, but eventually you will be ready.

Organizing collection systems and publications by type

A good architecture for taking a distributed approach to collection and publication activities is to focus on types of inputs and publications (see Figure 12-13).

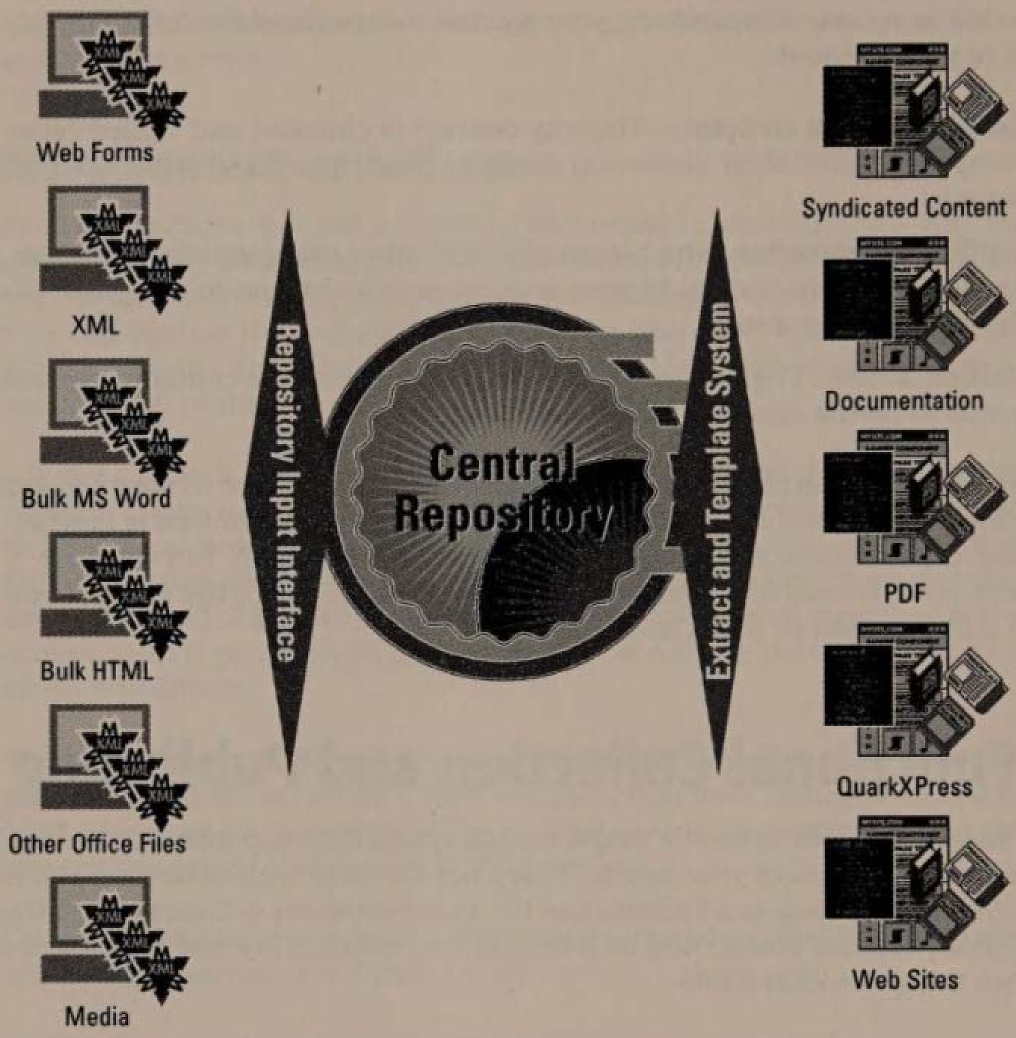

Web Forms
XML
Bulk MS Word
Bulk HTML
Other Office Files
Media
Repository Input Interface
Central Repository
Extract and Template System
Syndicated Content
Documentation
PDF
QuarkXPress
Web Sites

Figure 12-13: In the long run, you can focus on types of input and output, not particular components and publications.

After you have developed a process for bringing in Microsoft Word files, for example, you need not develop another process just because another group also has Word files. Instead, you can take the process you developed for group A and extend it so that group A *and* group B can both use it. In the end, your process will be general enough that group Z can train on it and adopt it without much intervention on your part at all.

Likewise, for publications you can begin to class your publications by type and extend your publication methods so that they can be used by others with the same types of needs.

When you have reached the state where you can route most new content and publications through standard collection and publishing interfaces, you will have a CMS with a very compelling value proposition for your organization. Your proposition would run something like the following:

> You can do your own thing and continue to face all of the problems that we have over the last two years, or you can sign up with us (the CMS team). With a short training session and an agreement to adhere to some standards, you can be up and running very quickly. Your contributors will have well-tested and full-featured tools for creating your content, and your publications will have all the bells and whistles of the best publications our organization now offers.

Creating functional teams

In addition to organizing your CMS functionally, by type of input and output, you can organize your team functionally by dividing it into central teams and a variable number of independent collection and publishing teams. Dividing your team in this way offers a number of advantages:

- **You can distribute the cost** of the system, and especially new additions to it, across the organization. Each group that wants to join the CMS initiative can field and pay for their own teams.
- **You can give a fair measure of independence** to participating groups. As long as they stay within the CMS guidelines, they should be free to organize their effort as they see fit.
- **You make your initiative more scalable** by creating a small central group that maintains standards and starts other teams and any number of independently operating peripheral groups. As the effort grows, the central group will grow only very slowly. The peripheral groups provide most of the new staff that are required to collect or publish more content.
- **You make a place for the people who know the content** best to be in control of its creation and for the people who know the audiences best to prepare publications for them.

The set of groups I show in Figure 12-14 represents one way to organize your CMS modularly so that you can adapt as your CMS matures (I'll be able to write much more about mature content management systems when some actually start to exist).

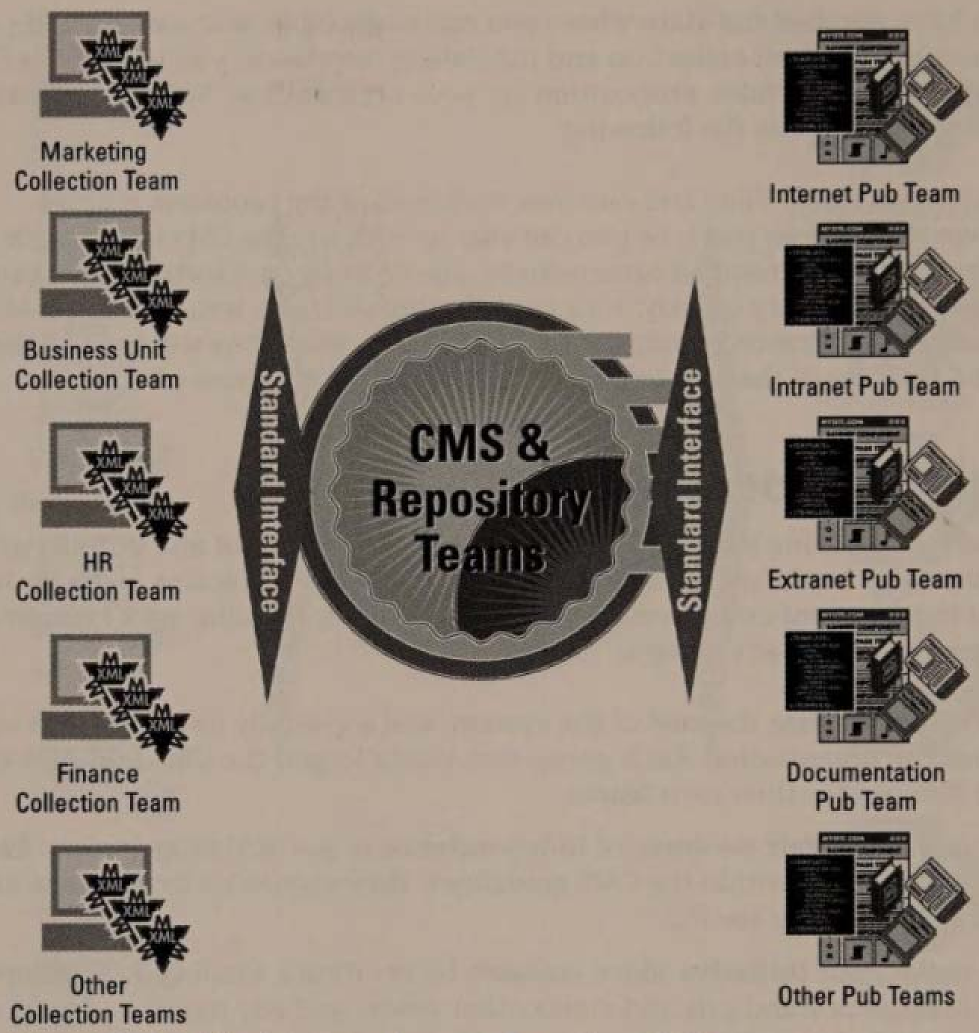

Marketing Collection Team
Business Unit Collection Team
HR Collection Team
Finance Collection Team
Other Collection Teams
Standard Interface
CMS & Repository Teams
Standard Interface
Internet Pub Team
Intranet Pub Team
Extranet Pub Team
Documentation Pub Team
Other Pub Teams
XML

Figure 12-14: The functional teams of a mature CMS

The CMS steering committee

The steering committee is the ongoing remnant of the mandate and requirements processes. They are the people in the organization that have a continuing interest in the system and enough authority, expertise, or both, to determine the direction of the CMS. The steering committee can have representatives from the CMS project sponsors, members of significant contribution and publication groups, and CMS analysts and other project team members with continuing knowledge and responsibility for the system. The steering committee can do the following:

- **Set CMS standards,** including content modeling trade-offs, when to make a custom collection or publishing process into a standard process, and when and how to extend current standards to include new groups.
- **Work with new teams** to determine whether they should become part of the CMS initiative, and, if so, when and how to integrate them into the current process.

- **Resolve content-sharing conflicts** between groups with competing needs.
- **Serve as the ultimate supervisors** of the CMS and repository teams.

The CMS team

In the first versions of the CMS, the CMS team is all there really is. As time goes on and the rest of the organization begins to participate, however, the CMS team can take more of an advisory role, helping new teams decide how best tie into the existing effort, including the following:

- **Readiness assessments** that help the new team to decide what they have so far, what they still need, and how they will fit into the existing standards.
- **Mandate assessments** that help create team mandates and fit the new team into the existing CMS mandate (or modify the CMS mandate to fit the new team).
- **Logical design** of the new team's content and publications that leverages as heavily as possible from existing collection, publication, staffing, and process models.
- **Help with implementation,** including assistance with specifications, development, and testing.
- **Help with deployment** that leverages as much of the existing infrastructure as possible.
- **Help with maintenance** that fits the new team into the existing maintenance procedures or modifies those procedures as needed.

The repository team

The repository team members are the keepers of the system and its standards. They are responsible for the following:

- **The CMS system,** including any ongoing installation, maintenance, or enterprise integration issues.
- **Ongoing training** on, and updates to, the CMS content schemas, including new and updated component classes and elements.
- **Standard collection interfaces,** including any Web-based form interfaces and batch upload systems.
- **Standard publication interfaces,** including the templating, personalization, and integration systems.

Collection teams

As groups with content to contribute come into the picture, you can constitute collection teams whose purpose is to gather and submit a particular kind of information to the CMS. Collection teams are responsible for these tasks:

- **Figuring out how their project's goals** and requirements fit into those of the wider CMS.
- **Performing a logical analysis of their collection needs,** including defining audiences, authors, sources, components, staff, and workflow.
- **Designing any additional conversion processes** needed for information from acquisition sources.
- **Implementing, deploying, and maintaining** any additional software and hardware that is particular to their systems.

Publishing teams

As more groups arrive with needs for publications, you can create publication teams that design and maintain the new publications. If you are making the best reuse of the content in your organization, there can be far more publication teams than collection teams. There are only so many sources of information in the organization, but an unlimited number of ways to use that information. Publication teams are responsible for the following tasks:

- **Figuring out how their project's goals** and requirements fit into those of the wider CMS.
- **Performing a logical analysis of the publication** needs, including defining audiences, components, templates, personalization, staff, and workflow.
- **Doing publication design,** including page layout and content, look-and-feel, editorial and graphic style guidelines, and user interface.
- **Implementing, deploying, and maintaining** any additional software and hardware that is particular to their systems.

Identifying Your CMS Hurdles

As much as an organization may need the services of a CMS, it might not be easy to agree on and implement a CMS. As a colleague likes to say, "when you pour milk on oatmeal, all of the lumps come to the top." Likewise, when you install a system that touches as much of the organization as a CMS does, all of the disorganization and local tensions come to the fore. In addition, the CMS requires a level of coordination and effort that many organizations are hard-pressed to muster. Following are some of the hurdles an organization may need to overcome to be successful when implementing a CMS:

- **Who owns the system?** The organization's departments must be able to work together to plan and implement the CMS, regardless of which group might initiate or lead the effort. It will require contribution from a wide range of groups. Organizations that are mired in inter-group rivalries risk failure in implementing a CMS.
- **What is our content model?** The major part of a successful CMS is a single, comprehensive scheme that says what all of your content is, what each part consists of, and how each type relates to the others. Without such a scheme, you have little chance of successfully sharing content across groups and publications. The key to creating a content scheme is constructive debate and compromise. No one person or group holds all of the scheme; rather, it is agreed to and shared across the organization. The organization's departments must be able to work together to create this scheme. In addition, the content must be untangled and organized into a tight system. If the departments can't cooperate or if the content itself is too scattered and disorganized, the CMS project is at risk.
- **How do we create general content?** Most content creators are accustomed to writing with a very particular publication in mind. For the CMS to succeed, authors have to think more widely about a set of target audiences and publications. If authors cannot break through to a less-narrow definition of their job, they will not effectively serve the CMS.
- **How do we create automatic publications?** Most publication specialists (print producers, Webmasters, and so on) are accustomed to creating publications directly. They lay out a publication and populate it with content by hand. They can change and tweak the layout and content until it satisfies them. In a CMS, publication teams do not create publications. Rather, they create the templates that create the publication. This level of abstraction is sometimes misunderstood or even resisted by publication specialists who prefer to create products, not systems.
- **Will the CMS stifle creativity in our authors and publications?** Many authors and publication specialists think that by putting together a CMS, they will no longer be able to express their creativity and will instead become cogs in a system, endlessly producing the same product over and over. Although a CMS does require that content be standardized and publications be created automatically, it does not require that creative efforts cease. Quite the contrary, a CMS requires an enormous amount of creativity to figure out how a base of content can be modeled and then successfully delivered to a potentially diverse set of audiences. The creativity in a CMS is simply at a different level. Rather than being creative at the level of individual content components and publication pages, you must be creative at the level of the types of components and the relationships between components and the ways that they can be delivered in publications. Far from being a static system, where the same activities happen over and over, the CMS requires constant feedback and refinement to get better at meeting the organization's goals. If authors and publication teams cannot rise to this new level of creativity, the CMS will not succeed.

- **Don't we already have a mandate?** Your project team may want to believe that it already has a clear mandate and that no further probing is needed. If this is so, then it ought to be easy for them to list the appropriate sponsors to write a complete mandate statement (like the one detailed in Chapter 14, "Securing a Project Mandate," section "The project mandate") and have the sponsors sign off on it. It is also possible that they are leery of losing control by bringing in others who might outrank them, are afraid of the controversies they know exist, don't believe that the sponsors have anything to add to the project, or just don't want to be bothered initiating a big process. In any case, proof of a mandate ought to be required before you skip the step.
- **How can we do this project when our project team has no power?** This is a real problem, if it is true. True or not, many project teams think this way and silently agree either to proceed covertly or just to give up trying to accomplish anything. Either the team really is disempowered and needs new leadership or a new sponsor, or it just feels that way and needs a renewed sense of itself. One sure way to find out is to really try to get started with a readiness assessment (as outlined in Chapter 13, "Getting Ready for a CMS," in the section "The readiness assessment"). Either your team will start to feel its oats, or it will find exactly where the brick walls are.

In addition to these existential questions, there are a number of simpler problems that you might encounter, such as the following:

- **The project has no team.** You will not need a team to start, but before long, you will not be getting much sleep without one.
- **There is a division between the team and sponsor.** This can be a killer if left untreated. I've seen more than one project drop dead because of an unresolved but underground conflict between the team and its sponsor.
- **The team wants to do it alone.** It is undoubtedly more trouble at the start to solicit a lot of input, but you will pay in the end if you do not.
- **The team is not interdisciplinary.** You can start from one department, but you will eventually need the entire range of skills represented on your team. All of the CMS jobs and skills are listed in Chapter 11, "Staffing a CMS."
- **The sponsor is not available.** If you can't get her attention, ask yourself if she is the right sponsor. Then decide exactly what you need from her and ask her very specifically. If you still get no response, find another sponsor.
- **Sponsor does not understand or accept her responsibility.** A lack of understanding is easy to deal with — educate her. Not accepting the sponsor role is a bit more difficult. Does she not want the role, not have the time, or not know how to fulfill the role? Keep in the back of your mind that whatever department you and she happen to be in now, this will eventually be an enterprise endeavor. Keep your eye out for potential sponsors who get that idea, have a sincere interest in the issue, and have the wherewithal to help make it happen. If you are lucky enough to find such a person, gravitate toward her, and maybe she will adopt you and your crazy ideas.

- **The organization underestimates the magnitude of the problem.** I hope that after reading this far, you are not underestimating the size of your project — but others might. Maybe your organization is not ready for a large-scale solution if it is unaware of a large-scale problem. Maybe you just haven't found enough pain in the organization yet (as described in Chapter 13, "Getting Ready for a CMS," in the section "Look for pain in the organization").
- **There is a rush to buy something.** I've seen this one a lot. The organization feels that by making a product purchase, they have done something tangible. Having bought other enterprise software packages, they might also have more of an idea how to accomplish this than how to spec-out a CMS. On the one hand, after you have bought a product, your life simplifies to just how to make a piece of software function. On the other hand, if you don't take the time upfront to figure out what you want, the system might never be able to do what you end up wanting it to do, or you might never actually even figure out what you want to do in your rush to make something happen.
- **The team wants to move forward** even though they have no consensus and still have lots of issues to resolve. Direct that frustration toward getting closure on the key issues. If you must move forward, work in the areas that bring the issues into highest focus. If you are being stonewalled by a group that will not send you content samples, for example, make some quick prototypes of their pages and tell them that you will hold them to the component structure you came up with unless you are otherwise notified.

I hope that by giving you this long list of problems (which I ended long before I could have), I have inoculated you against them. Just like a vaccine, maybe if I give you a little bit of the disease now, you will not catch it later.

Summary

Your organization can be your biggest asset or an albatross around your neck. If you can figure out how to fit the CMS naturally into the organization, and if the organization will let you do so, it can become a part of the fabric of the enterprise.

- Content management systems directly serve the organization's need to produce and distribute value.
- Information now flows in the organization through well-worn channels. Learn where these channels are and how to tap into them.
- Each group has a part to play in the overall system and none has the right to own the system to the exclusion of others.
- If you organize functionally and do not force every contributing group into the same collection or publishing model, you will have a better chance of acceptance and scalability.

- Any number of factors can slow you down or stop you. If you are aware of those factors from the start, you might be able to head them off before they present a problem.

In the next chapter, I describe some techniques and deliverables you can use to prepare yourself and the organization for a CMS.

✦ ✦ ✦

CHAPTER 13

Getting Ready for a CMS

◆ ◆ ◆ ◆

In This Chapter

How a CMS project is similar to and differs from a usual software or system development project

How to measure information pain, and other assessment techniques

The readiness assessment and other preliminary deliverables

◆ ◆ ◆ ◆

Your organization creates and distributes content today. Before planning an entirely new process, you are well advised to study the current process. Before designing your new system, come to understand the ways your organization creates and publishes information and functionality, and what constraints they will put on the system you want to create. Your goal here is to work outward from your CMS project team, through to the sponsors of the project within your organization, to the audiences you hope to reach, and to the contributing groups in your organization, in an effort to find their needs, constraints, assumptions, and blind spots.

In this chapter, I'll provide an overview of the CMS project process and discuss how you might go about getting yourself and your organization ready for a CMS.

Understanding the CMS Project Process

Any large development/integration project has these broad phases:

1. Business justification
2. Requirements gathering
3. Design
4. Implementation
5. Deployment
6. Maintenance

The process I propose for doing a CMS project is quite a bit like the general process, as shown in Figure 13-1.

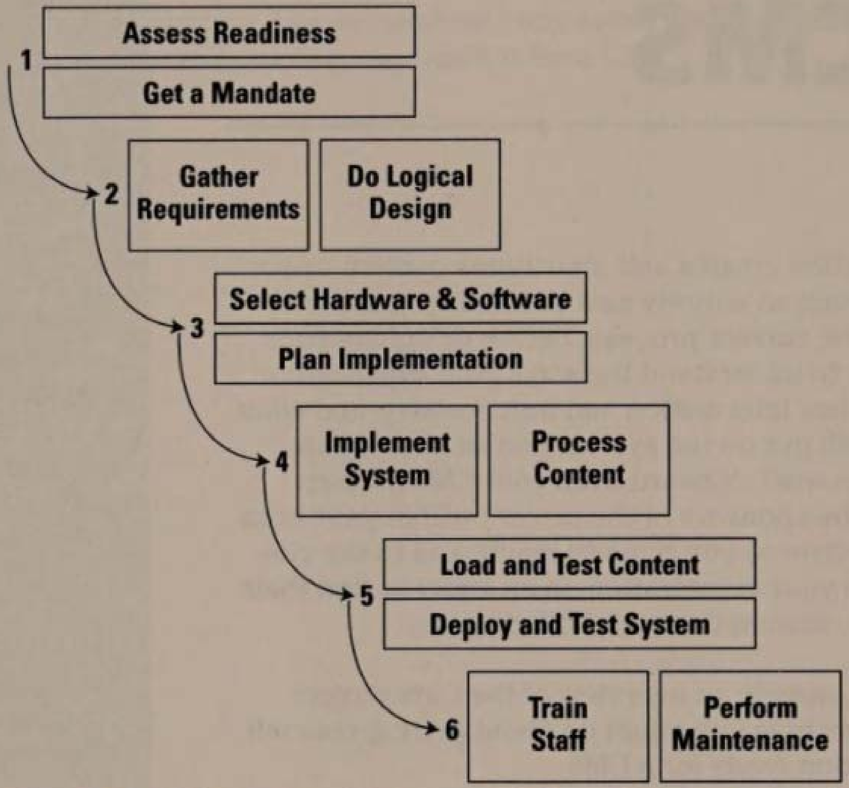

1
Assess Readiness
Get a Mandate
2
Gather Requirements
Do Logical Design
3
Select Hardware & Software
Plan Implementation
4
Implement System
Process Content
5
Load and Test Content
Deploy and Test System
6
Train Staff
Perform Maintenance

Figure 13-1: The CMS project process follows the same general process as any other large development project.

Even though I use some different names, the ideas are generally the same, as follows:

1. **Business justification:** This step is taken care of in my readiness assessment and mandate processes. In these two processes, you decide what the organization has accomplished so far and then build consensus around a plan of attack. The readiness assessment is discussed later in this chapter. The mandate process is discussed in Chapter 14, "Securing a Project Mandate."
2. **Requirements gathering:** This is a specific step in my CMS process, but it is not as extensive as the standard process. I favor a short requirements-gathering phase, followed directly by what I call logical design. During logical design, you continue to gather requirements of a sort, but your real task is to fashion those requirements into a clear idea of what your system must accomplish. I cover requirements gathering and logical analysis and design in Chapter 15, "Doing Requirements and Logical Design."

3. **Design:** In my process, this step begins with logical design, but cannot conclude until you have selected a particular CMS system. In system selection, you use the requirements you have completed and part of the design to create an evaluation process for selecting a CMS product (or, possibly, concluding that you want to build your own). Before you select a system, you have to do enough requirements gathering and design to know what you want. Before you complete the design, you have to know what product you will be using to account for the product in the design. Between design and implementation, then, is a system selection stage that overlaps them both. You can find more on system selection in Chapter 16, "Selecting Hardware and Software."
4. **Implementation:** In my model, this step includes the last part of what is often categorized as design — specifications. I call specifications the physical design of the system and include it as the first part of implementation. Following physical design, CMS implementation proceeds as usual — with a lot of programming. In addition to programming, though, in a CMS project, a lot of content processing might need to be done. To be accomplished by the time the system is ready, this preparation work needs to be started as soon as possible. See Chapter 17, "Implementing the System," for more on implementation.
5. **Deployment:** As with other systems, during this step you install the system in its "production" environment and test it there. In a CMS project, however, loading and testing the content and publications that the CMS produces is also part of deployment.
6. **Maintenance:** In my model, as in other enterprise systems, a CMS project never ends — it just goes into maintenance mode. This is doubly true of a CMS, where you will be adding content continually and will also want to restructure the repository and publications on a regular basis. Deployment and maintenance are covered in Chapter 18, "Rolling Out the System."

The major difference between the CMS process I use and the standard software development cycle is that in the CMS process, you do enough design to fully understand what you want to have happen and to be able to select a product (or choose to build your own). Then you select a system that can achieve the design you have made. Next, you complete design by developing the specifications for your system. After that, you are back on the usual implementation, deployment, and maintenance mill. One other significant feature of a CMS project, over and above the usual kind, is the potential size of the content processing portion of implementation. In the standard development project, moving data into the system is not nearly the task that it can be in a CMS. Of course, a CMS project is a large-scale systems integration project, so it shares a lot with other projects where software must be purchased and integrated into the organization.

Techniques for Getting the Job Done

To get ready for a CMS, you can start within whatever project team you happen to have. Working outward, you can survey the organization for signs of pain related to poor or nonexistent information processes and catalog them for later use. It is most important to try to uncover and document whatever CMS mandate might exist in your organization, to see just how much of a project is currently expected and by whom. Your biggest task is to catalog the current assumptions in the organization on the following issues, which are of most importance to a CMS:

- Audiences
- Publications
- Content
- Infrastructure systems

Having assessed all of that, you will be ready to go from what exists in the organization to what needs to exist to successfully implement a CMS.

Start with the project team

Some individual or group must have been given the task of figuring this all out. I state it so vaguely because that is about how precisely most organizations have defined the problem at the start. Generally, there is a loose consensus that something must be done, but when you ask, "Done about what?" the answers vary. The problems most often cited are the ones I have listed elsewhere. In most minds, they boil down to this statement: "It is too expensive and ineffective to handle our Web presence in the ad hoc way we have handled it so far." Somewhat more enlightened minds see the duplication between the Web and other publications as an unnecessary evil. Very few can succinctly define the problem or frame a comprehensive solution.

Even in the fog, everyone seems to have a gut sense that there must be a better way to handle the large and growing body of content that the organization is amassing. Few know to apply the words "content management" to the problem or solution.

In sum, a person or small group is directed by someone to figure it out. Let's call this group the project team. Unfortunately, this someone who directs the team to figure it out (the project initiator) is often not someone who has the perspective to see the forest for the trees. As often as not, the project initiator has a particular problem to solve (like "I can't get my changed pages on the site fast enough"). In addition, the initiator might or might not have the authority, skill, or desire to pursue the kind of big solution that I present here.

But at least he or she had the foresight to get the process started. What the team will soon need is a solid set of sponsors that do have the perspective to see the entire problem, the foresight to envision a whole solution, the authority to mandate a solution, the skill to negotiate throughout the organization, and the desire to transcend organizational politics and do something new and risky but replete with reward.

One way or another, then, a team is created to figure out how to more effectively create a Web site and maybe other publications as well. Job one for this team is to get their bearings in the organization and chart a course toward a solution that will stick. If they are a savvy team, they will not rush toward a quick-fix solution. Rather, they will hang back a bit (despite the pressure to produce) and try to define the problem, realistically assess the progress the organization has made, build consensus around an appropriate solution, and then begin to build a system. Admittedly, the savviest team in the world might not be able to resist the pressure of near-sighted goals and deadlines, but the savvier they are, the more the team will realize that in the end, their own success will be judged not on how quickly they acted, but on how well their solution worked.

Still, an enormous amount of finesse and fortitude can be called for if you are being asked to do X and you believe that you need to do 100 times X. Of course, I am painting the darkest picture here, a picture of sponsors not up to the job and teams with myopic mandates. I have seen too many of these sorts of situations not to acknowledge them and provide some moral support to those who face them. I have also seen sponsors who are up to the task and teams whose mandate is broad indeed.

At any rate, I'll assume that the team has management that is enlightened enough to expect and respect a fair amount of up-front due diligence. This due diligence amounts to understanding what the organization has accomplished already; gaining general agreement on the goals and success metrics of the project; compiling the requirements for audiences, publications, content and infrastructure; and educating the organization to ensure that there is enough of a shared vocabulary and concept set that you can communicate and agree.

The core tasks of the project team show directly the skills the team needs to have in order to succeed. The following list overviews these skills which you will need to have in at least one member of your team:

- **Analyze and assess your organization.** For this task, you need a business analyst. This person is responsible for answering questions like, What groups need to be involved? Whom do I contact to get permission to . . . ? Who are the key supporters we need to have on our side? Who needs what education? How do we maneuver through the organization's bureaucracy? How will we build consensus around a mandate? How do we align and extend organizational goals? How do we measure success? Someone from management is a logical choice for this job, but it must be someone who can manage "up" and not down. It must also be someone who is quite analytic and strategic in her approach.

- **Understand your audiences.** You need an audience analyst. This person is responsible for figuring out who the appropriate audiences are, what they want, what you want from them, how finely to divide them, what good analyses exist in the organization, how to align your system to current marketing approaches, what data you need to collect on audience members, and how to collect it. Someone from marketing or public relations is a logical choice for this job, but it must be someone who is less focused on campaigns and more focused on analysis. Someone from an editorial background is also a possibility, but in this case, you need to be sure the person is comfortable being quite quantitative in her approach to audience analysis.
- **Understand the publications** that your organization will be creating from the system. Thus, you need a publication analyst. Even if your first plan is only to create a Web site, you should still have your eyes open for the ways in which the Web site needs to share with other publication efforts. This person is responsible for finding out what publications exist now, how they share information, how they are produced, how often they are produced, what their audiences are, how they are distributed, and how they can be deconstructed into pieces that a CMS can produce and coalesce. Someone from the current Web effort, or another major publication group, is a logical choice for this job, but that person must be able to climb out of the publication she has been creating and look beyond the particulars of a publication, to how publications in general can be created.
- **Understand the content** that the system will manage. This task requires a content analyst. This person is responsible for finding out what kinds of content you have, what you need, how it serves audiences and publications, how it can be divided into content classes, how each class can be reduced to a set of content elements, and how those elements can be fashioned into a metadata framework for tagging content. She is responsible for understanding how information is produced in the organization and where it can be found. She will assess the amount of work it will take to write or acquire content and how much will be needed to start and run the system. Someone from an editorial background is a logical choice for this job, but it must be someone who is less focused on creative writing and more on mechanics. Someone with a library background is another good choice, but the librarian must be able to understand the creation process as well as the cataloging process.
- **Understand the computer infrastructure** upon which the system will run. You therefore need a technology analyst. This person is responsible for understanding all of the systems in the organization that will interact with the CMS. She is also responsible for understanding any constraints or technical requirements the organization might have. Later, she will be central in the selection of any new hardware or software you will need to build the system. She must be able to understand and piece together every piece of technology used, from the first authoring application to the final piece of JavaScript in an end user's browser. Someone from a development or IT systems background is a logical choice for this job, but she must have a head for content as well as data. A technologist who is comfortable ignoring the human parts of the system will not do.

You will need to bring all of the diverse perspectives together and unite them around a clear vision of how they all fit together. Thus, you will need a project leader who can form and articulate a clear plan of attack that takes advantage of all the skills in the team. This person needs to have a little bit of the skills of each team member but also a wider perspective of the entire system and process. Because she will often be improvising (for lack of any established process), she needs to be a strong communicator and a forceful director. Finally, she needs to embody an information perspective. That is, rather than focusing too much on any of the team member skills, she brings them to bear on the central problem of the entire project—what is the right system for us to best collect, manage, and publish the information and functionality that our audiences want and from which our organization can benefit.

Cross-Reference

I present a more comprehensive look at the jobs and skills required for a CMS in Chapter 11, "Staffing a CMS."

After the system is in development, of course, you will need other skills. But the skills you need later are just a deeper and more technical version of the skills just discussed.

Look for pain in the organization

A good way to begin any conversations when examining your organization's CMS needs is to ask what information problems the person is facing. If you ask (and even if you don't), you probably will hear a plethora of issues, dilemmas, worries, and horror stories that surround the core issues of content management. Certainly, if you do not hear a lot of woe, you should question the need for a CMS! Be sensitive not only to the types of problems, but also to the actual stories that people tell. These stories will come in very handy when educating the organization, bring a measure of "ground truth" to the abstract discussions you will find yourself in, and keep your team on track toward solving the problems that are most pressing. As you hear these stories, do the following:

- **Record and categorize them.** Distribute them freely to let everyone know where the problems are.
- **Try to find the common themes.** These are the major issues for your organization to resolve.

Try to rate the severity of the problems. See if you can put some numbers around the comments you hear, especially concerning the problems of a lot of information, slowness of the system, people complaining, and unsatisfied customers.

Assess the current mandate

If you are extremely lucky, there is a one-page, crystal-clear description of what the organization expects to accomplish with a CMS. The page has a paragraph that anyone can understand that says why this system is necessary and what it will do. It then goes on to list the major goals of the system. Finally, it has a shortlist of the criteria by which the system's success will be measured.

More likely, you have had a set of discussions with the project initiator, where you learned of her goals and requirements. You might have an e-mail message or a memo that reiterates the discussions. Maybe that is all there is, but maybe not. Where did the initiator get the idea? Were others involved? Is there some wider initiative of which this is a part? If you dig around a bit (gently, of course), you might find that a number of overlapping movements have led to your project and that it serves a number of organizational goals. If others are unaware of the connections, you have material for an education plan. Most likely, there is a loose agreement (and some disagreement) on the issues surrounding a CMS. Where exactly is the agreement and who agrees? Where is the disagreement and what are the divergent opinions?

In addition to the text of a mandate (or lack of one), you should explore its scope. Given your current understanding of how far-reaching your system will need to be, are the right people involved? Are other initiatives out there that are not connected to this one that maybe should be?

Assess the organization's assumptions

For each of the kinds of analyses you must do, there is a set of relevant documents and people in the organization that you have to find, understand, and present back to the organization.

Audience assumptions

You would be exceedingly lucky to have a good audience analysis in your hands at the start of your assessment. On the other hand, you would be in a very strange organization indeed if there were not a tremendous amount of raw material that you could use to begin a good audience analysis. Any marketing or public relations group worth its budget will have a lot of information about the organization's customers, constituents, members, or whatever else audiences may be called. If some or all of your audiences are internal staff, then the human resources department is the place to go for the divisions and groupings that matter.

I'll stress again that this is raw material, but at this point, raw material is all you need. Your purpose is not to complete an audience analysis, but to see what the organization has to offer when you do one later. Editorial groups are also storehouses of audience analysis. Although the amount of written material might be sparse, you will find no lack of people who have thought about audiences and have a very clear idea of who they address in their work. Finally, the people in your organization that are in the most contact with audiences will have a lot of good, practical input for you to put in the hopper with the more abstract definitions you will get from people who think about but do not actually talk to audience members. As you work to uncover information about audiences, keep an eye out for key members of audience groups. Is there someone out there who typifies a particular audience? Would that person be willing to talk to you, or even consult with the project team, concerning what is best for her peer group?

Publication assumptions

All organizations create some set of publications (even if it is only internal information). To implement a CMS successfully, you must decide which publications ought to be part of the system and then how to subsume them into a CMS, where they can share content, functionality, and design.

To begin this process, you can simply catalog the publications your organization produces that might be part of the system. It is quite likely that each major publication your organization creates comes from a different publication group. Each group is focused on a particular author group, content set, and publication format. One group, for example, might produce case studies and industry intelligence for sales support. Another group might produce documentation for post-sales training. A third group might produce an internal newsletter that highlights recent advances and best practices. Talk to the people in each of these groups to find out what content is in their publication, how often it is produced, who receives it, and how it is delivered.

Just as important as understanding the structure of current publications is understanding the structure and attitudes of publication groups. Find out how the groups are staffed, who does the writing and other content creation, and who does the layout and other production tasks. How do these groups feel about the idea of a CMS? Do they understand it? Do they support it? How do they think it will impact their jobs? This sort of information will be crucial to you later as you develop an approach to implementing the system.

Content assumptions

In the end, you will have to be very specific about the information you include in your CMS. To start, however, you can cast a much wider net and find out what kinds of information are out there in the organization and what sorts of information people are assuming will be included. As with audience information, it is unlikely that anyone has created the definitive guide (eventually, you will), but there should be an enormous amount of raw material in the organization. Rather than trying to amass the whole bulk of information that might be included, focus on collecting representative samples and any catalogs that might exist. Catalogs can be anything from official content inventories to the TOCs and indexes of existing publications to a listing of the files in a directory.

Tip

It is almost lost wisdom that, at the DOS prompt, you can navigate to a directory and use the command `dir /s > filelist.txt` to save into that directory a file called `filelist.txt`. The saved text file lists names and details about all files and subdirectories in that directory. (Or, to just get a list of filenames with paths, you can add the switch `/b`.)

Without a CMS, content and publications are pretty much the same thing. That is, the publications contain the content that you will collect and manage. A group creating industry analyses now, for example, will be creating the content and simultaneously publishing it (as, say, printed white papers).

Just as important as creating a general content inventory is discovering the structure and attitudes of the content groups. What staff and workflow do they use now? Do they use any automation systems, templates, or style sheets? What kind of standards do they adhere to? Is there an editorial guide? What are their assumptions about their audiences? How do these groups feel about the idea of a CMS? Do they understand it? Do they support it? How do they think it will affect their jobs? This sort of information will be crucial to you later as you develop an approach to implementing the system.

System assumptions

Eventually, you will need a thoroughgoing analysis and specification for the technological structure of the CMS and for how it will interact with other organizational systems. To start, however, a simple description of the requirements people have on their minds and a listing of existing systems will do.

Especially in a large organization, there are likely to be a variety of conflicting requirements that technical staff would like to set for the CMS. Some groups already might have begun working with a particular CMS and will want the entire organization simply to adopt theirs; other groups might have a strong preference for one development environment over another. Still others will have requirements based on a particular sort of content or publication that they want to create. People concerned with the organization's infrastructure (the IT group, for example) will want any new systems to tie into the existing network and Web systems.

Avoid adding to the debate at this point and simply record and categorize what you hear. Later, you can spend the time it will take to reconcile these various opinions into the best fit for the entire effort. From the very start, your best position is not as a combatant in the war of requirements, but as a mediator and neutral provider of unbiased information.

For each system that you come across that might have to interact with the CMS, find out how it is able to communicate with other applications. For example, is it able to exchange data using XML? How does one connect to it? Is it accessible from a Web server outside the firewall?

Of particular interest is the Web infrastructure your organization uses. On what platform does your Web presence run? What databases and application server software (if any) are in use? What methods are there for moving information to the Web server (or servers)? How is content replicated and distributed worldwide (if it is)?

Similarly, study the distribution of information in non-Web publications. How must print files be delivered to the printer? How are mass e-mail messages and faxes distributed today? How are publications that come in file form distributed internally?

Finally, gather what information you can on collection tools and processes now in use. What authoring tools do groups use? Have they created any productivity tools? What development skills do they possess and how have they been applied?

Taking Stock of the Deliverables

As you prepare your organization to begin a CMS project, you can prepare the following tangible outputs to document your progress and provide a starting place for the rest of the process:

- **A readiness assessment** gauges the organization's progress toward a CMS in a number of key dimensions.
- **A document inventory** catalogs all of the plans, memos, decisions, and other artifacts that you can uncover about past and current CMS-related initiatives.
- **A state-of-the-system report** summarizes your findings in an easily understood and consumed form.
- **An education plan** identifies the people in the organization who need to know more about content management and how and what you will attempt to teach them.
- **A preliminary project plan** charts out the major phases of the coming project and broadly estimates the amount of effort you expect each one to take.

The readiness assessment

To my mind, the best place to start a CMS project is by getting a firm feel for what the organization has accomplished so far. Such an assessment will give the project team an immediate, action-oriented task. Go through the whole organization and find out what has been done and what the current assumptions are.

In the process of assessing the current situation, the team will become acquainted with all of the players and all of the significant documents. The people with whom you interact will get the chance to assess you informally and see that you are ready and interested in what they have to offer. Conversely, you can assess the various organizational contributors and decide what offers are worth following up on. If you do this first job well, you will build an enormous amount of brand equity for your CMS project team within the organization and will initiate just the relationships you will need to continue and complete the project. Of course, the main reason for this task is to uncover a lot of great information to be used in the coming project. I call this process a readiness assessment.

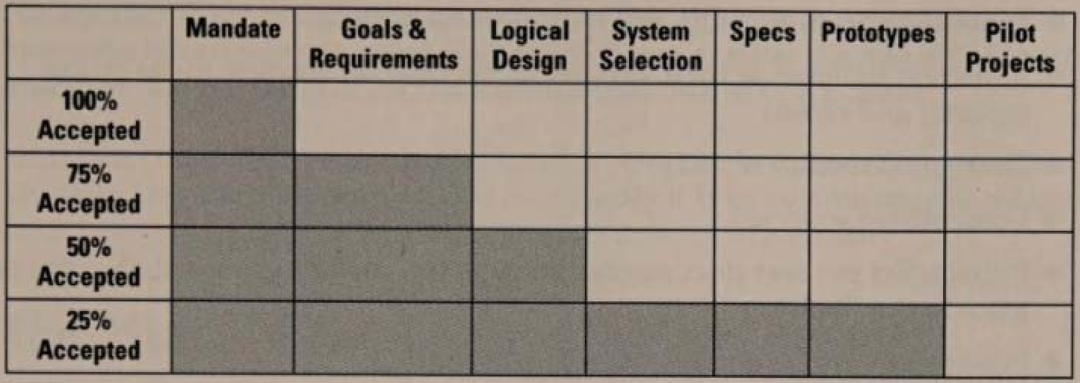
Both before you begin, and then again, when your readiness assessment is complete, you may want to apply the complexity equation that I presented in Chapter 8, "Knowing When You Need a CMS," to assure yourself and the organization that a CMS is indeed warranted.

The following parts of the readiness assessment should be no surprise to you if you have read the first part of this chapter:

- **What mandate exists** in the organization for the project now?
- **What audiences** does the organization expect to serve and how well-developed is the organization's approach to these people?
- **What publications** does the organization expect to create and how suitable are they to a CMS approach?
- **What content** does the organization think it needs to deliver, and how well-known are its quantity, structure, and use?
- **What system** does the organization need or expect to be under the CMS? Are there well-formed requirements or a variety of opinions?

If you do no more than ask yourself these questions, you will be ahead of most teams. If you go out to the organization and collect and organize opinions and documents on all of these subjects, you will have the start of a great project. And, in fact, that is a good method for doing the assessment. Start with the project team. What do you know and believe about these subjects? Who knows what you do not? What documents have you seen that mention any of these things? Who would know if there are other documents? Who in the organization cares about any of these subjects? What kind of people would care about them? Begin with what you know and then follow the trail you began to uncover the people and policies that bear on your subject matter. Try to get beyond what the project team assumes or has seen. In so doing, you will gain valuable experience in how to engage the wider organization in the system you will later need them to support and to which you will need them to contribute.

You can also look at readiness from more of a project-completion standpoint, focusing on how far the organization has gotten in defining the mandate, project requirements, publications, and constructing a system, as shown in Figure 13-2.



	Mandate	Goals & Requirements	Logical Design	System Selection	Specs	Prototypes	Pilot Projects
100% Accepted	▨						
75% Accepted	▨	▨					
50% Accepted	▨	▨	▨				
25% Accepted	▨	▨	▨	▨	▨	▨	

Figure 13-2: This chart shows at a glance how far the organization has gotten toward a CMS.

This chart combines general project milestones with general levels of completeness. The project milestones are organized more or less in sequence so that early milestones are toward the left. At the start of the project, you hope for a profile somewhat like the one shown. The most work has been accomplished on the early milestones and the later ones have not been started.

Unfortunately, as often as not, the profile is much less regular. It is common, for example, for the organization to have gotten quite far in system selection before even beginning on a mandate. A chart like this can help you get an early snapshot of the areas that are overdeveloped and those where the most early effort is needed.

Document inventory and analysis

Some organizations have a tremendous amount of documentation hanging around from previous electronic publication efforts. Other organizations have very little. In either case, it is worth your while to become the central repository for all such documents, to become very familiar with their contents, and to know how they relate to the current effort. As you make your way through the organization, look for these sorts of documents:

- Requirements documents
- Specifications
- Lists of content in spreadsheets and documents
- Site maps
- Policy statements and other memos that show the intent of the organization
- Presentations and demonstrations of concepts
- Vendor evaluations

- Proof-of-concept projects and their documentation
- Audience analyses in their various forms (market studies, focus group reports, and so on)
- Strategic direction reports
- Competitive analyses
- Publication process documentation from the various groups that might participate in the CMS
- Publication design documents and illustrations
- Marketing strategy documents
- Usability studies and user interface documentation
- Lists of organizational resources, products, assets, or any other documentation that might help you create metadata lists later on (typically, at least some sorts of product or service taxonomies exist)
- Tables of contents and indexes from any potential publications or content sources
- Any other artifact that looks like it might bear on the CMS

I'd be very surprised if you did not come up with a wealth of information if you really look. A thorough document inventory provides you with an enormous resource. First, it gets you familiar with where the organization has been, who proposed what, and what the outcome was. You will have a tremendous advantage over most people you will encounter, who will have vague, secondhand knowledge of what has been tried but nonetheless will be willing to base strong opinions on it. The more obvious benefits are that you will be able to leverage a lot of this information toward your effort, will meet all the right people in your search for these documents, and will not have to repeat the mistakes of others.

The effort of a document inventory also pays for itself in the feeling you get that you know what went wrong and what went right in the past and so can better design the future.

I suggest that you *ultimately* organize your inventory based on the part of the CMS that each document bears on, but that you start much more simply. For example, you might first start simply by gathering and perusing each document. Classify them in a spreadsheet by source, filename, and date. Add a note column for contents.

As you make your way through the mounds of information that you collect, you can begin to categorize them by the readiness categories: mandate, audiences, content, publications, or system infrastructure. Alternatively, you can use the project completion categories I outlined in the preceding section. I don't think it is worth the effort to go much further in categorizing these documents. For one thing, it is enough of a struggle getting them to fit into the few categories that I mentioned.

For another thing, one document will often totally cross your boundaries. If you can spare the effort, you can pivot your spreadsheet and list the major CMS categories and then cite the parts of each document that bear on each issue.

Your team ought to become pretty familiar with the key documents that you unearth. You might be surprised at how handy it is to be able to refer off the top of your head to the right official document to underscore a point.

The "State-of-the-Content System" Report

After a week or two of research, you ought to be able to say something definitive about the current state of content management in your organization. You can make your research public in a "State-of-the-Content System" report. The point of the report is not to show solutions, but rather to do the following:

- **Educate.** Simply by laying out the organization's knowledge using the categories and terms of your analysis, you can do a lot of education. Understand that this report is the first that many people will have ever seen of a content management approach. Be sure it helps to frame the issues from a content management perspective and shows why these categories are helpful in starting to frame a solution.
- **Provide the first stage of consensus.** Send the report first to the people with whom you worked to gather information. Do they agree with the way you represented them? Did you miss anything? What did they think of the stuff others provided that is similar to or different from their opinions and information? By doing this, you can gauge the intensity of the coming process. If people are so polarized that they cannot even agree on what the situation is now, you have a long way to go to get them to agree to what should be! On the other hand, if there is no profound disagreement in the report or if those in disagreement see some validity in each other's positions, you are set to drive to consensus on a solution.
- **Position the CMS project team as the central point of contact and expertise** on the system. It is not unlikely that, even while you conduct your research, other groups and individuals are questioning your authority to be leading this effort. Or, more benignly, they are simply wondering what you are up to and how much they should support you. If your first report is authoritative, well constructed, complete, and compelling, you at least will have seized the initiative and, at most, secured it.
- **Provide a platform on which to build the rest of the project.** The information you gathered ought to provide a bounty of ideas and discussions for how to move forward. You ought to have discovered the low-hanging fruit as well as the key stumbling blocks to moving forward. You should have ample input for the next stage of the project: building a project mandate.

- **Relate the project to the existing efforts.** You can include enough of your document inventory in the report to show that you have done your homework and that the assessment you have made relates directly to the organization's key policy and design documents.

You can choose your own format for the report. You will be well served, however, if the document contains some explanation of the process and its goals, as well as a section for each of the parts of the analysis.

The education plan

Your "State-of-the-Content System" report can help educate people concerning the current situation in your organization. Given the analysis you have performed, you can also take a more direct approach to educating your organization on the ideas and methods of content management. Your analysis ought to have shown you who the concerned parties are and what they know. If you can identify the gaps in knowledge in your organization, you can put together a formal or informal plan to fill them.

This may seem like an optional move, but I don't believe it is. In an area as new as content management, the lack of knowledge often presents itself as a lack of agreement. I've seen many hour-long arguments end with "we are just using different words for the same thing." More important, without a shared vocabulary and understanding of concepts, you cannot create consensus. It is not that everyone needs to become a content management expert, but rather that they must have the same basic idea of what content management is and what, for example, a publication is.

The analysis you performed ought to give you all you need to design an education plan. Following are some suggestions for how to go about it:

- **Include the people that do get it in your plan.** It's really helpful to have these people assist you in some way to educate their peers. This need not be a big-time commitment. Rather, what you want is their moral support to help explain, uphold, and proliferate the set of common words and concepts.
- **Find as many ways to educate as possible.** Anything from one-on-one meetings to mass e-mail messages can help get the organization wrapped around the new concepts. In general, face-to-face meetings are better for education, but do not hesitate to create short white papers, memos, and reports to spread the word quickly and effectively.
- **Don't imagine that you have all the right words and answers.** Content management is not a done deal. Even this book is just a bid by me to help define the content management playing field. It is much more important to reach consensus on some set of words than to have them be the words you would prefer. If there is anything that you should insist on, it is that the words your organization adopts to describe the problem and the solution do more to clarify than muddy the waters.

- **Find the right message for the right person.** For each of the significant players in your organization, decide what exactly they do not understand and what the right approach to teaching them is.
- **Do not confuse education with propaganda**. If your organization perceives that you are using an education banner to slyly convince them to follow your agenda, they will neither learn nor be convinced. Instead, create a separate forum, where the merits of particular approaches can be debated. It is a big enough task simply to get the parts of the organization to agree to a definition of the problem without adding the mistrust involved by pushing for a particular solution.

A preliminary project plan

Regardless of how much pressure you are feeling, this early in your project, you are not in a good position to put together a project plan to which you can be held. Still, you *are* in a position to put together the form of the plan, note any of the expectations people have for cost or schedule, and get the project plan process rolling.

Cross-Reference

You can, and should, revisit the project plan many times. The next time you visit, you can add a lot of reality to it. See Chapter 15, "Doing Requirements and Logical Design," in the section "A revised project plan," for details.

If you follow the process I lay out, your project will go through the stages I laid out earlier in this chapter (see Figure 13-1).

You might want to engage a project manager and use an official project-planning tool, or you might want to be a bit more informal. In either case, consider including the following information in your plan for each stage of the project:

- **Completion:** Include any assumptions you have found for how complete people think that the stage is, plus your own estimate of the same.
- **Cost and time:** Include any assumptions you have found for the cost and schedule that is expected for this stage. Include your own estimates of the same or your reasons for believing that it is too early to say. If it is too early, indicate what needs to be known to better fix the price or timeframe.
- **Deliverables:** Again, include others' assumptions and your own ideas. I have listed deliverables for all stages in other chapters. Use these or create your own.
- **Staffing:** Include whatever anyone has suggested about how to staff the effort. In addition, don't forget to include your guesses as well (my guesses are detailed in Chapter 11, "Staffing a CMS").

Tip

If you do use a formal approach to project planning, be sure you clearly communicate that this is a preliminary plan. As any graphic designer will tell you, "If your preliminary effort looks too professional and clean, people will mistake it for what you really intend to do!"

A risk assessment

At this point in the project, you will have proceeded far enough to start assessing potential risks. Admittedly, these risks will mostly be the risks that bad organizational processes introduce, but they are project risks nonetheless.

Tip You may not want to call people risks, even though they often are. Try phrasing your risk in a general enough way that it can't be assigned to a particular person.

If you don't have your own favorite format for risks, a simple table like this one will probably do:

Risk	*Impact*	*Probability*	*Score*	*Mitigation*

where

- **The risk column** names the risk.
- **The impact column** rates how bad the consequences will be if the risk actually occurs. You can choose whatever scale will give you enough of a range to differentiate the various risks.
- **The probability column** rates how likely the risk is to occur. A scale of 0 to 100 percent is reasonable here.
- **The score column** multiplies the impact by the probability to come up with an overall score for the weight of this risk.
- **The mitigation column** explains what you are doing to make sure the risk does not occur and what you will do to recover if it does occur.

Cross-Reference I suspect that you can discover any number of risks for yourself, but if you need a start, you can look in Chapter 12, "Working within the Organization," in the section "Identifying Your CMS Hurdles," to get some ideas.

Summary

The effort you put in up front to understand where your organization has been, who the players have been, and what they have accomplished so far will be handsomely rewarded throughout the rest of your project. In fact, whether or not there *is* a project might depend largely on how well you can fit it into these parent efforts. Even if you do so more casually than I describe, don't neglect to do the following:

- Assess the knowledge of your project team (if you have one).
- Find the major pain spots in the organization.
- Discover what sort of mandate exists for your effort.
- Assess the assumptions that the organization has now for the CMS audiences, publications, content, and systems.
- Develop a knowledge base of relevant documents, be able to say how far the organization has gotten in any previous CMS-like efforts, know who needs more education and how you will help them get it, and be able to roughly estimate the project before you.

If you do all of this, then the only reasons you still might not get a project going are a dysfunctional organization or the fact that a CMS is simply not needed in your organization yet.

In the next chapter, I talk a lot more about dysfunction and function in the organization as you seek to build consensus around a workable mandate for a CMS project.

◆ ◆ ◆

CHAPTER 14

Securing a Project Mandate

◆ ◆ ◆ ◆

In This Chapter

What it means to have consensus on a CMS project

How to recognize and understand project sponsors

The project statement and other mandate process deliverables

◆ ◆ ◆ ◆

A typical content management project begins with wide agreement that there is a problem — too much information to manage informally. There is tacit approval from organizational decision makers that some solution must be found, and there are the beginnings of a content management project team. On the other hand, the project begins with little or no understanding (let alone agreement) about what the solution will be, what parts of the organization will be affected, how long the project will take, or how much it will cost. Without compelling answers to these and many other questions, the project will flounder from the start.

In this chapter, I discuss how you might work outward from your team to both your organizational sponsors and to your audiences to drive toward consensus and a workable project.

What to Agree On

The place to start a CMS project (or any other major organizational initiative) is by building consensus around some fundamental issues. You build consensus among the three *major* stakeholders in the project: the organizational sponsors, the project team, and the final judges of the system (your audiences). The internal stakeholders (sponsors and the project team) must agree on the following issues:

- **What exactly is our information management problem?** Where do we feel pain, and what are the most significant changes that need to be made? How big is the problem?

- **Where are we now** in our efforts to address these problems? What steps already have been taken, and what documentation of current approaches and analysis has there been?
- **What group of people will be responsible** for creating a solution? Who will lead the group, and what will the responsibility of each person be?
- **What is the organization's project mandate** and upon what criteria will success be judged?
- **Who are the intended audiences** for the publications that the system produces?
- **Are there organizational standards** for development or deployment that must be used?

All three sponsor groups must agree on these issues:

- **What sorts of information and functionality** are most important to capture and distribute?
- **What sorts of publications** will be needed to distribute content most efficiently and effectively?

Maybe you are lucky and during your readiness assessment you uncovered the perfect mandate for your initiative. More likely, you uncovered areas of agreement and areas of disagreement from which you must build an overall consensus for your project. To build consensus, you must find and coordinate a set of sponsors.

Techniques for Getting the Job Done

Negotiating your way to a happily agreeing organization may be tough. It will take skills beyond technical expertise or any knowledge of content management processes. To my mind, the best way to approach the task is to learn about the people who will make or break your initiative — your sponsors. If you go into the consensus-building phase of the project armed with all the knowledge you gained in the readiness assessment, fortified with a thorough understanding of the people you must get to agree, and reinforced with a fair but strict consensus process, you might just make it out of this phase with your pride and project intact.

Recognize sponsors

A sponsor is someone whose support you need to make the project successful. The ultimate test of a sponsor is to ask, "Without this person's support, will the project be seriously compromised?" Conversely, you can ask, "With this person's support, will the project be seriously enhanced?" Either kind of person is a sponsor. Sponsors can come from a number of areas, as follows:

- **Organizational executives.** These are the classical sponsor types. They are people who have sway in the organization, promote ideas they believe in, and can get budget allotted to projects that they, well, sponsor.
- **Key influencers.** These are people whose opinion is often sought to test whether an idea is worth pursuing. They might or might not have budget or authority, but they do have respect and credibility in the organization.
- **Key outsiders.** These people usually are from important audience segments, such as customers, partners, members, or other pivotal constituencies. More than just being part of the audience segment, to truly be a sponsor, they must speak for the audience or have some specific influence.
- **The person who initiated the project.** Undoubtedly, this person feels like a sponsor but might or might not in fact be someone who can make or break the project.

Most sponsors are from within the organization, but some might not be. In fact, the presence of an outsider in the sponsor group can help enormously to create a mood of solidarity and agreement. Most sponsors are decision makers, but some might not be. They might have influence, but no direct power. The presence of influencers in the sponsor group helps to deepen the discussion, especially if the influencers are known experts in some aspect of the system.

Regardless of where a sponsor comes from, she must be someone who can truly offer support. This means that sponsors are both in a position to support and want to do so. I have often seen groups seek weak support from a more powerful sponsor rather than strong support from a less powerful sponsor. The road to completion of a CMS is long and can be quite rocky. Without a personal interest and commitment from your sponsor, you might not make it. As you seek sponsors, then, try to assess the degree to which the person can commit for the long term to your goals.

Another useful way to think about a sponsor is as someone who will help you get beyond your organization. That is, a good sponsor will help you avoid the red tape that might be involved in getting an initiative off the ground. Without such support, your project can get mired in a morass of meetings that never manage to merge the multitude of opinions and make measurable progress toward real milestones.

Every sponsor comes with her own set of attitudes and interests. Some will want to exert direct control, while others will want only quarterly reports. Some will have a lot of background in what you are doing, while others will have no clue. Regardless of background, however, a good sponsor will understand what is your responsibility and what is hers. Possibly the best statement I have heard from a sponsor went something like this:

> "I don't understand what you are doing nearly as well as you do. On the other hand, I know our business and I know a good decision when I see one. I'll question you mercilessly until I understand the significant facts and figures. I'll let you know where we agree and disagree. Where we agree, I'll support you wherever I can. Where we disagree, we have some more work to do until we agree."

You and the sponsor must agree that your job is to know what to do and hers is to know how to get it done in the organization or with an important audience group. If you can get to this accommodation with your sponsors (and they with each other), you will have cleared the first major hurdle on the way to success.

Learn about your sponsors

A good sponsor will help your project because she believes in its goals and in your ability to achieve them. Still, if you recognize the needs and motivations of your sponsors, you will be more likely to craft a project that helps them achieve their own goals. In addition, if you are not lucky enough to handpick your own sponsors, you will do well to try to understand the perspective of the project sponsors so that you can best navigate among their dispositions and desires. Following are some of the questions you might want to answer about your sponsors (whether or not you choose to write down the answers is a different question):

- **Why is she a sponsor?** If this is simply part of her overall job, is it a part she particularly cares about? If she does not seem to care a lot, will this mean that she will default to accept or not accept what you propose to her?
- **What will make her successful?** How does a CMS relate to the criteria by which she is judged in her job? What risks to her success does your project represent? In many organizations, the way each person is judged is public. If you can get this information about your sponsors, you can use it to craft your approach to your project and to them.
- **What will compete for the attention and support of your sponsor?** Is this the smallest, largest, or one of many initiatives in which the sponsor is involved? If this is a big initiative, expect it to receive a lot more scrutiny and attention, and remember that it will be much more important to plan your communication with this sponsor.
- **What alignments does your sponsor have?** Are there particular groups in the organization to which she is particularly close? Are there individuals whose opinion she particularly respects or disrespects? How are these groups or people disposed toward your project?
- **What stake does she have in your success?** How noticeable will it be to her peers and supervisors if you succeed or fail? Similarly, who is watching the sponsor and what does she want or need from the project?
- **How well does the sponsor understand the organization's problems?** Are her assumptions about what is wrong with the status quo the same as what you have discovered? What education might she (or you) need before you see things the same way? How will you deliver education to this sponsor? Are e-mail messages and memos enough, or will you need something more?
- **How well does the sponsor understand content management?** Does she get it enough to understand the words and concepts you use? Does she get it in a way that you don't? Can you help her understand how to apply her implicit understanding of what goes on in the organization to your set of words and processes?

- **What specific reviews and approvals does she require?** What sorts of communications and project policy statements does she think she should a) not see, b) review, c) approve, or d) write? How formal should each review be?
- **How does she relate to your audiences?** Some of the worst sponsors I have seen erroneously have believed that they could speak directly for an audience. They used this position to squeeze the project team from both sides (using their purported knowledge of the audience to back up any statement of opinion that they made). Not that sponsors can't understand audience needs — quite the contrary. Many have had a lot of experience with audiences and can add to the discussion. It is the specific responsibility of sponsors, however, to speak for the organization and it is only an added benefit if they can speak for an audience. Conversely, if the sponsors do not understand the needs of an audience, they must certainly be made aware of them. A recognized but unmet need of an audience is an argument that a sponsor will have difficulty ignoring.

Taking Stock of the Deliverables

Your mandate-building process will consist of a lot of talk and very little action. Your job is to keep the talk productive and leading toward an agreed-upon set of goals and success measurements. You can produce the following tangible deliverables to help the process along:

- **Sponsor profiles** that list exactly what each sponsor wants and needs
- **An issues hierarchy** that fits all of the contending needs of all of the sponsors into a single framework
- **Notes and minutes** that mark progress and help you stay on track
- **A mandate process** that encourages debate, then progresses toward consensus, and ends in acknowledged closure
- **A project mandate** that captures the agreement you reached and presents it to the organization

Sponsor profiles

To really study your sponsors, you can create written profiles that answer all of the learning questions I listed earlier in this chapter. Now, if you are truly a consensus builder, you immediately will be questioning the wisdom of this endeavor. Taking notes on people and creating files on them is not the way to gain their confidence and trust. So, if this seems just too risky or Machiavellian, skip it and content yourself with thinking about the questions and discussing the answers you find with the sponsor and your team.

On the other hand, if you can get past the feeling that you are acting like the next J. Edgar Hoover, this approach can be of immense value. If you act covertly and sequester the information you collect, you will indeed be doing espionage. If you are open about your motives and up front about your techniques, however, you are just being conscientious. Moreover, if you have the sponsors participate in filling out their own profiles and openly discuss the profiles in group meetings, the profiling process can be as innocuous as any survey.

The point of profiling is to be explicit about how you will approach each sponsor. The more open you can be with this information, the better for everyone. In fact, the degree to which you feel like you can be open with your profiles indicates the degree of general openness in the organization1 If you can manage to create a climate in which it is all right to share your approach to sponsors openly, you will be creating just the right climate for honest debate and real agreement.

Still, you can only be an idealist for so long. To my mind, it is always best to lead with complete openness and retreat as required to retain your personal comfort and your job.

One way or another, you should have in mind a profile for each sponsor so you can think explicitly about how to get each one to agree.

A hierarchy of issues

As you leap or crawl toward consensus, you will hear a blizzard of seemingly unrelated opinions, recommendations, and dictates. These will become a tangled web that will confuse and then capture the group, unless you control them. The best way I know of to control blizzards of information is to organize them (no surprise there). And, of course, western culture's greatest organizational tool is the outline, or hierarchy.

Note If you want to be really anal about it, you can produce indexes of the issues and cross-reference them as well.

By getting all of the statements of what you should do in one outline, you have done all you need to keep them from getting the better of the group. Your outline can do the following:

- **Assure the group** that all issues are captured and available. People then do not have to keep in their heads all of what they have heard, and they ought not get the feeling that "there are big issues out there that we are forgetting about."
- **Contain the issues** by putting them in the wider project context. By organizing the issues into an outline, contrasting opinions will come together, similar issues will merge, and sub-issues will be clearly seen as part of wider concerns. Hierarchies are syntheses of the items they contain. That is just what you want to accomplish.

- **Center yourself** as the one who is keeping track of the issues and bringing back potential resolutions to the group. The outline can be your main tool for showing progress, or lack of it, to the group. It also provides you with the perfect platform for offering options and potential solutions.
- **Provide an ongoing working draft** of the final project goals. If you drive your outline toward a small number of top headings, you can be simultaneously driving your group toward a small number of top-line goals.

Take care that you don't get too far ahead of the group. The work you do on the goals hierarchy should be logical and reasonable extensions of what the group has said or meant, not your own opinions and ideas. You must maintain a careful balance between synthesizing the ideas of others and inserting ideas of your own. If you are too far on the side of your own ideas, you may lose credibility as a facilitator of the process and be seen as trying to drive rather than direct.

If you keep the balance and keep driving the outline toward a final resolution to which everyone can agree, you will have a great tool for keeping the process under control and progressing. You will also know when you are finished.

Notes and minutes

Groups of highly placed individuals have exceedingly short memories. They are apt to repeat themselves, or change their minds without remembering that they used to think differently. In addition, during the mandate process, there will be a lot of facts to record and organize. For these reasons, and to stay on top of the issues, you are well advised to take good notes.

It is probably not a good idea to have the same person writing the notes as facilitates group meetings. The facilitator needs to be actively listening, responding, and directing. The note taker needs to be faithfully recording. In taking notes, be sure to listen for and record the following things:

- **Whether an issue raised is an opinion,** fact, recommendation, or directive. Later, it will be important to know how strongly stated the issue was and in what context any solution was offered.
- **Disagreements.** Note the disagreeing parties and the exact point of contention. Sometimes merely writing these down miraculously cures them.
- **Agreements.** Note the issue as well as the agreement that was reached.

I think you can figure out how to take notes at least as well as I can (someone who never takes notes). My main point is just to make sure that you capture an ongoing record of what was suggested, what agreements were reached, and what still needs work. This record will help you construct an issues hierarchy and will ensure that there is something to go back to when people forget.

The mandate process

Assuming that you have done all of your readiness assessment homework, you will have a good idea of what the organization's main information problems are. Assuming that you have managed to get around in the organization to talk with all concerned parties, you will have a good idea of whose opinion counts. From this beginning, your next task is to formulate a solid mandate for the system that you will create.

I've never seen an organization that has no established rules for how to get something done. Whether these rules are explicit or in the collective consciousness of the staff, rules of engagement exist. Thus, I cannot presume to provide you with some process that you can use as such. Instead, I'll propose general considerations that whatever specific process you use must take into account.

A lot of people will want to be heard on the issues, but only the small group of sponsors will get to decide. You might consider providing a voice for the larger group by soliciting their input and cataloging it for presentation to the sponsor group. I have often created simple Web-based forms for this purpose. The forms ask questions such as "Rate these audiences in order of importance to the organization," or "Which of these publications is most critical for us to deliver electronically?"

You probably already will have talked to many of these people and will have asked them some of these same questions. You haven't asked them to officially register their opinion on specific issues, however. By doing so, you are officially including them in the process. This is not only a goodwill gesture. As your sponsor process unfolds, it is really helpful to be able to canvass a wide range of opinion on a matter of dispute. There is nothing like getting hard facts back when you hear a lot of what seems to be second-guessing the needs of the organization.

It is helpful to categorize your opinion-givers by the kinds of opinions they are most qualified to give. You might guess that audience, publication, content, and system are the categories I would use. These categories will help you keep the number of questions you ask any one person to a minimum and will ensure that you get back the highest quality responses.

You need to establish some procedure for sponsors to process a possibly large amount of information, provide input, review the input of others, and come to agreement. I have used a variety of methods to accomplish this, all of which shared these common qualities:

- **Setting the rules of the process before you begin.** Some rules I have found useful are a commitment to attend meetings and answer all e-mail messages, and a commitment to end with a single agreed-upon statement.
- **The process ought to offer a degree of anonymity in input.** That is, you should offer ways for people to offer opinions without a name attached. This ensures that not only the dominant or highest-ranking people get to decide. I have often used tools, for example, that enable anonymous ranking and commenting on alternatives (MeetingWorks, `www.meetingworks.com`, is a good example).

- **The process ought to include bottom-up and top-down decision-making.** For example, you can ask, "What are our goals?" and keep merging them up to a single project statement (bottom-up). Then you can ask, "What is our project statement?" and, from that, begin to derive goals (top-down). I have found that whichever way you start, it is a good idea to validate your results by going the other direction.
- **Present the facts.** Make sure that any facts you have found (things such as how many we have, how long it takes, and who is currently responsible) are presented and agreed upon by the sponsors before a discussion begins. Then be sure to refer to these agreements concerning the facts as people begin to get creative in the heat of debate.
- **Present the opinions you have collected *as* opinions.** Be especially sure to outline contrasting opinions. Put the sponsors in the position of reconciling approaches and opinions, as opposed to presenting and defending them.
- **Make sure you cover all angles.** Dominant players will tend to drive the discussions toward their areas of interest. Remember that in the end, you need to have covered all four topic areas (audience, publications, content, and systems).
- **Make sure you play angles off each other.** It is not enough to dive into each of the four topic areas separately. Ask questions like "Which content is most important to each audience?" and "To which publications does this content naturally lend itself?"
- **Be prepared to break and educate.** Note when the discussion would be helped by a quick discussion of vocabulary or concepts. Be aware of when disagreements are based on dissimilar terms. You are not only building consensus on the solution but also on the definition of the problems.
- **Stay out of the fray.** It rarely serves you to jump into the debate and offer opinions. For one thing, it puts you on a winning or losing side. More important, while you are in the fray, there might be no one watching the process to ensure that all voices are heard and a single group opinion emerges. If you think that you must contribute, make sure someone else is there to facilitate the meeting. Consider using a professional facilitator if you think that the group will be particularly contentious or if you think that you want to be part of the discussion. If you do intend to participate, remember that later you will want the support of the people with whom you are debating.
- **Be prepared to break for more facts or wider opinions.** As a tactic to break up contentious discussions, as well as a way to quietly deflate unfounded opinions and facts, create a method for tabling an issue in order to canvass opinions for others not currently in the discussion. If you have previously created a method to get quick turnaround on questions from the wider group of concerned parties, this would be the time to use it.

- **Reward conflict and consensus.** In this process, you need to walk a fine line between too much and too little contention. Often, groups will deadlock with conflicting opinions. But just as often, they will seem to have agreed when in fact, those who did not agree were not paying attention, did not understand that to which they were agreeing, or did not want to speak up. Pay attention to both situations and test a weak consensus by restating it plainly and getting each person specifically to agree.
- **If you can't solve the problem,** solve the problem of how to solve the problem. Don't let a deadlock go for long without backing up one step. Divert the discussion from the problem to the process for resolving the problem. Go back as far as necessary in order to find agreement. Then start working back toward the deadlock one step at a time, trying to build on the agreement you reached.
- **Note the dynamics.** Use every clue at your disposal to understand how people are disposed to the process and each other. Note body language, attitudes of boredom, frustration, apathy, or anything other than honest interest and engagement. It is up to you to decide what to do about dynamics (good or bad), but even if you don't know what to do, be sure to pay attention.
- **Use a variety of venues.** When working with sponsors, it is useful to have a bag of discussion tricks up your sleeve to deal with their chronic lack of time and focus. Big group meetings have the advantage of everyone meeting face-to-face to work through issues. (We have found that the statement "We will not leave this room until we have agreed about X" is particularly helpful if you find that the group meanders and fails to decide.) Your process can drag on for weeks or months, however, as you try to find a time that works for everyone. Consider smaller group meetings, gathering opinions via e-mail, one-on-one meetings between warring parties, and any other gatherings that will help you hammer out consensus. I have found that bringing the entire sponsor group together at significant times (the start, major decisions, and the end) is very helpful in building group identity and ensuring that every sponsor gets the same message at the same time.

However you design your mandate process, the result ought to be a strong consensus. The consensus specifically should not be on the details of the system but on the concept, goals, and success metrics of the system. You are finished when all of the sponsors exit the mandate process with the same idea of what you are doing, what organizational purpose it serves, and how they will measure your success.

Of course, some groups run out of time or sponsor patience before they reach this nirvana. If you find yourself in this position, pull back toward a mandate to which you can stick. The worst outcome from a mandate process is a public stalemate. If the process ends and it is clear to everyone that major rifts prevented a unified statement, you are in a poor position to move forward. If you choose to move forward anyway, beware. However you choose to proceed will be in opposition to someone. More important, the organization will know that you failed in your first major test.

It is not enough for a room full of sponsors to agree on the solution; the rest of the concerned parties must agree as well. Given that you have strong sponsors, a strong statement, and have solicited input at all significant junctions, it will be difficult for others to find fault. Still, unless you publicize the results of the mandate process, how will they know with what they are agreeing? Moreover, your mandate is your calling card to any group that you call upon for assistance. It should be right out in front of your group as you make your next moves. Finally, your mandate is your contract with the organization about what you will do. It behooves you to make sure that anyone involved in the project understands the ground rules that the mandate represents. Don't be afraid to ask questions of your team, such as "Which goal does that help us meet?" or "How does that support our mandate?"

In short, your mandate is the rule of law behind your project. It ought to keep the organization from changing the rules, and it ought to keep your project team from changing the rules. This does not mean that you are locked in to any specific course of action. Quite the contrary, a well-crafted mandate specifies outcomes, not actions. It leaves open your options on how to achieve the outcomes. Rather than restricting action, then, the mandate motivates it. It serves as the context in which your decisions must live. When someone comes to you with a proposal for an approach or a technology, the mandate does not say yes or no. It does say, however, "Whatever you do, make sure it results in X."

The project mandate

When politicians win an election by a large majority, the first thing they do is claim a mandate. They mean to say that the people are united behind them and in overwhelming support of their initiatives. It is less important, in the present context, whether politicians actually have a mandate, and more important that the politician needs to claim one. A mandate is a powerful tool to keep skeptics and critics at bay and forge ahead, especially when times get rough.

In politics, a mandate is often a vague and vacuous statement, such as "People believe in my positions." Many content management projects are begun with a similarly vague statement, such as "Everyone agrees that we need a CMS to better deal with our information." If you expect your mandate to stick, or provide any measure of support, you must be much more precise and clear. I can point to the following three specific elements that a mandate must have to stand up over time:

- **A project statement** that precisely and clearly defines what you are trying to accomplish with the CMS
- **A small set of goals** that the system will be expected to accomplish
- **A way to measure** the system's progress against each of the goals

The project statement

To function as a mandate, a statement must be short and simple enough to be learned and recalled, but powerful enough to continue to exert influence over the project. Over the course of a project, staff and sponsors must be able to restate the mandate over and over to different concerned people in order to explain and justify the project. If it is not crisp and convincing, it will not last through the first telling. On the other hand, if it is clear and compelling, it will require little extra effort to show anyone interested the scope or importance of the CMS. A good project statement has the following attributes:

- No more than a few sentences
- Phrased in plain language that anyone can understand
- Captures the essence of what you are trying to accomplish
- Can be used as a yardstick for the remainder of the project to determine whether something or someone fits with the project

The project statement is a proposed solution to the pain you found in the organization, as well as a way to coalesce the audience, publication, content, and system assumptions that you uncovered in your readiness assessment into a simple paragraph to which everyone can agree.

Suppose that you decide that the main problem your organization is experiencing is the inability to get product information in a timely way to sales people who need it. The organization experiences this as a high training cost for sales staff (because it takes a long time for them to learn where the information is and how to get to it). Your project statement might read as follows:

> We will reduce training costs by building a complete Web-based information base of our product line — SBase. SBase has all of our product information stored logically and presented in a way that the sales team can understand it and get to it quickly.

Anyone to whom you said this (inside or outside the organization) would understand what you are doing and why. In a few words, it gives all of the most salient facts: reduce training costs, Web-based, all product information, understandable and accessible to sales staff. The paragraph does one additional thing: It names the system. This may seem like a small move, or even an unnecessary one, but it is very important. If you can create a name for your system that is memorable and descriptive, you will go a long way toward creating first, consciousness, then understanding, and finally, acceptance of the system.

To keep the example simple, I purposely made this statement limited in scope. This system has only one major content type, one publication, and one audience. Not a particularly sophisticated project. Nevertheless, it is a start. It might very well represent the project that all sponsors agreed was important enough to fund. It might be the point of most pain and what the organization felt needed its limited resources most. It might represent the core information that later will be leveraged from sales staff to customers. The point here is that the statement must reflect what the organization can really accomplish.

You can pretty easily imagine how the statement might be extended for a wider mandate — a broader content base, more audiences, more publications. The real trick is to keep the statement short and sharp, given a wide mandate.

System goals and measurement

A project statement includes a top-level view of the goals of the system. It answers the question, "Why are we doing this?" This goal level is good for getting people to understand the project but is not concrete enough to use as a measure of the project's success. To get more concrete, you can create a list of the concrete outcomes you expect from your project and how you intend to measure those outcomes. The goals of using SBase might be to help the sales group as follows:

- **Reduce training costs.** We note that today it requires an average of eight weeks before a newly hired sales representative is ready to take an incoming call. Our system will be successful if this time drops to seven weeks after the system has been in place for six months.
- **Reduce sales closing time.** We note that today our average time to close a sale is 17 days. We will consider the system a success if we note any drop in this closing time after the system has been in place for six months.
- **Create a single source for all product information.** Our analysis found a total of nine different directories, Web sites, and individuals that are used today by salespeople to get product information. Our system will be a success if a similar analysis six months after the launch of the system shows that the nine information locations are no longer used.
- **Create a logical organization for all product information.** Our analysis shows that the organization has four major competing schemes for organizing product information. Our system will be a success if we are able to create a single product information outline, index, and cross-reference structure that is deemed better than the other four schemes by the users of the other four schemes. In addition, we will consider the system successful if it takes no longer than two minutes on average for a contributor to correctly categorize a new piece of product information.
- **Provide quick access to product information.** We will consider the system a success if it takes no longer than five minutes on average for a salesperson to find product information or discover that what they are looking for is not on the site.

Note that each goal has a statement, a benchmark of today's metric (if there is one), and a quantitative measure of success. The statements try to squeeze all of the vagueness out of the goal in order to provide an objective and very tangible standard by which you can be judged. Many project teams would cringe at being held to such tangible standards. There is no room to fudge on success. I'll assume that your attitude is not to hide behind vagueness, but to have a bright-line test for success to which you can hold the organization as you make your way toward success.

This method of goal setting does require a bit of extra effort to track the success metrics that you are using to judge your project. But it is effort well spent. In the preceding example, the training time and the sale closing time ought to be measured regardless of the CMS. The alternative product-information sources will be good to know anyway, because they are your competition. If you do not get agreement on your organizational scheme, or if it takes too long to categorize new information, your system not only did not succeed, but it will fail. Finally, the average time to find information is, for this kind of site, the key metric you need to know to gauge how well the site is performing. Good metrics are far from extraneous; they are the key gauges on your management console.

Summary

If you go forward into a CMS project without securing a project mandate, you might get very lucky and feel no effect. On the other hand, your entire project might fall apart because of forces you never even knew were active. On the positive side, a mandate causes the entire organization to side with your effort. Even if you don't absolutely need support, don't you want it? The process of securing a mandate does the following:

- Requires you to recognize and understand the potential sponsors for the CMS project
- Consists of a guided process for garnering support and agreement by your sponsors
- Is aided by creating sponsor profiles, an issues hierarchy, and by taking careful notes
- Results in a project statement, a set of project goals, and an agreed-upon way to measure progress against those goals

In the next chapter, I describe requirements gathering and logical design, in which you use your mandate to conceive and design a CMS.

✦ ✦ ✦

CHAPTER 15

Doing Requirements and Logical Design

◆ ◆ ◆ ◆

In This Chapter

A simpler sort of requirements gathering process than you might be used to

The concept of logical design

Techniques for doing requirements and logical design

The types of deliverables that result from requirements and logical design

The people you may want to include in your requirements gathering and logical design process

◆ ◆ ◆ ◆

After you have secured a project mandate, you can begin to collect all of the information you need to design the system with your sponsors and other stakeholders in mind. You can begin by gathering your organization's requirements for content, publications, and CMS infrastructure. From there, you can create a product-independent (or logical) design for your CMS that defines exactly how you intend to collect, manage, and publish information.

In this chapter, I'll lay out some project techniques and deliverables you can use to collect and organize requirements and construct a logical design.

Note This chapter covers the project process behind requirements and design. The actual requirements and design considerations you need are covered in the chapters in Part IV, "The Logical Design of a CMS."

What Are Requirements?

Requirements are the specific qualities, features, and objects that the system will ultimately include. In a CMS project, there are three sorts of requirements to be collected:

- **Content requirements,** which specify the kinds of content to be managed and how it must be gathered and organized
- **Publication requirements,** which specify the kinds and structure of outputs you want the CMS to produce, including how you want to target content to your chosen audiences
- **CMS requirements,** which specify how the CMS hardware and software are required to operate

It is likely that you already have a lot of this sort of information hanging around. By the end of this stage of your project, however, you need to have these requirements fully nailed down.

I think the word "requirements" says it all. You will be required by your organization to follow these guidelines in designing and constructing your system. If you take this seriously, it can work very well for you. In particular, you will be very careful about what you call a requirement. You will be sure that 1) the statement is of enough importance to be called a requirement, and 2) you are happy to be held to it. In addition, as you collect and organize requirements, you can get the organization to help you by underscoring the importance and durability of requirements.

What Is Logical Design?

At the end of requirements gathering, you will have a head (and a hard drive) full of other people's dictates to you about how the CMS should be and why it is being created. Next, it is your turn to respond. In logical design, you decide how the CMS will meet all of the goals and requirements that have been set out for it.

I use the term logical design to contrast this process with the later process of physical design. In physical design, you will put together the specifications and plans that say how the CMS will actually be constructed from hardware and software. In logical design, you put together the system outside the constraints of a particular platform. Another term for logical design might be platform-independent design. So why would you need to do such a thing? Here are a few good reasons:

- There is much more to a CMS than hardware and software. Content, for example, is neither hardware nor software.
- You should understand what you *want* to happen before you decide *how* you will make it happen. There is nothing more time-consuming and potentially career adverse than creating a system, only to find out later that it can't do what you never took the time to figure out that you wanted it to do.
- You need information to help you decide what sort of hardware and software you will need to build your CMS. The best tool you have for deciding what sort of platform you need is to know what you want it to do. Even if you already have a CMS in place, before you start pushing a bunch of buttons, you should have a clear idea of the results you intend.

Your Logical Design

Logical design specifies what you want to have happen in a system. Physical design refers to the particular parts you bring together to make the system happen. There are a number of other places where the same relationship between physical and logical design occurs, as the following examples illustrate:

- A computer itself has a physical design, with chips, disk drives, and a lot of wire. It also has a logical design. In response to a given set of inputs (ones and zeros that are presented to the computer's central processing unit), the computer is designed to perform a particular action and produce a certain output. The computer's physical design can run software with its own organization and internal logic.
- A supermarket has a physical design, with shelves, floors, ceilings, and lots of products. It also has a logical design. It is a system for moving products from loading dock to shopping basket. The physical design of the supermarket can "run" promotions that move your body and eyes to the products the "supermarketeers" most want you to buy.
- Your brain has a physical design, with neurons, hormones, and a lot of connections. It also has a logical design. Groups of neurons form functional areas that take care of particular functions. Groups of functional areas gang together to perform more complex functions. At an even higher level, the physical design of the brain can "run" consciousness and thought processes that have their own organization and logic.

In all cases, the physical design allows but does not determine the logical design. The logical design works within the broad constraints of the physical design. It can't do something outside the bounds of the physical design. On the other hand, it can do things that are beyond the physical design. My word processor, for example, enables me to write. Writing is not something that is part of the physical design of the computer. In a CMS, the hardware and software that make up the system allow but do not determine the kinds of content and publications you will create.

There is not one physical and one logical design. Each shelving unit in a supermarket and each neuron also have a physical and a logical design. In fact, in interesting systems such as computers and brains, there are myriad levels of designs. In a CMS, the physical design itself is composed of software and hardware with their own internal logical and physical design.

The logical design is the real reason for the system. The physical design of the computer, supermarket, or (arguably) the brain is there so that there is someplace for the logical design to exist. The same is true for a CMS. You want the logical design of contributors, components, and publications. You need the physical design of computers and networks that enable it to happen.

Logical design is like a big puzzle. The pieces are bits of information you collect or derive from the requirements related to authors, sources, components, publications, and so on. You have to fit them all together so that they make a complete and sensible picture of the collection, management, and publishing process you intend to create. If a piece does not fit, you must rearrange the others until it does, or get rid of the outlier.

Logical design results in process, relationship, and structure, as follows:

- **You design the processes** by which you intend to collect, manage, and publish content. For example, you must figure out how a particular kind of content can be collected at the right rate to meet the publication cycles of all the publications in which it will appear.
- **You work through the relationships** between the significant players in the system until they all support and enhance each other. For example, you must figure out how you can get your authors to create content components in such a way that they can be targeted to your various audiences, using publication templates.
- **You design a content structure** that fully represents the information you need to distribute and adds enough metadata to it to enable you to automate its management. For example, you need to create a set of content component classes that organize the information you want to deliver, and augment it with management data you can use to automate its delivery.

By the end of the process, you might long for the days when everyone was telling you what to do. Or, if you're like me, this might be your favorite part — figuring and finessing and finding a way to make very human factors fit into a logical and very effective framework.

Techniques for Getting the Job Done

The techniques in the following sections help you understand what I mean by requirements and how you might go about gathering them, including how to approach your organization most effectively to get them to give you the requirements you need. They also help you learn and practice some techniques for logical design, including how to go from business goals to a system, how to iterate a number of times through the process, how to triangulate on the various aspects of design, and how to manage the enormous amount of detail you will collect.

Gathering requirements

The requirements process I will lay out is quite a bit simpler than the one most software developers use. Rather than asking an exhaustive set of questions that will lead directly to a design, I recommend that you ask simple questions that provide you with the overall direction you will need in order to dig further to come to the more fundamental questions and answers.

The process I present is predicated on the assumption that the people you will approach for requirements will have neither the time nor the insight to give you real answers. They do, however, have the ability and the authority to determine the broad outlines of the system.

The requirements process

Your requirements process needs to result in an agreement on content, publication, and system requirements. The point here is to get the people in the organization to tell you what they really need. You are not trying to get them to learn all of the terminology and process that you will need to turn requirements into action. Nor are you trying to get them to work any harder than absolutely necessary. So, your best approach, I believe, is to ask a set of simple yet comprehensive questions that require only short answers.

Before diving into the more precise set of questions to ask, let me propose an overall process for gathering requirements: canvass widely, catalog everything, and get consent.

Canvass widely

Cast a net as wide as is feasible to get input from a wide base of contributors. With any luck, through your previous work in assessing the organization and building a mandate, you have already isolated the list of potential contributors. The list ought to be narrower than the one you used originally to find out what was going on in the organization. It should include only those people who are in a position to "require" something of the system. On the other hand, the list should be wider than the people who participated in the mandate process. They are probably not the only ones with the knowledge and authority to state what the CMS must accomplish.

Tip

You can certainly get the process off to a good start by providing a list of the potential requirements that you already have amassed from your initial communication. It also helps to ask different people for different types of requirements, based on what they are qualified to comment on.

Catalog everything

Your position in the requirements process is scribe and librarian. The most help you can offer is to keep the list of potential requirements from getting out of control and becoming confusing. The work you do to synthesize people's input into an easy-to-read and easy-to-understand document is crucial. When someone does not see her words in the document you produce, you should make it clear that her requirements are being superseded by larger and more central concerns rather than ignored. There is a subtle art to combining requirements in a way that no one feels slighted but that inferior input is weighted less than superior input.

Tip A technique I have found useful here is not to wait until I am finished to get feedback. Rather, I go back and forth with a number of contributors, passing small pieces of the requirements between us as I work on the final draft. This technique has the effect of giving work to people in bite-sized chunks (that is, the size of chunk that can get done in an e-mail exchange) and building consensus before the draft is even circulated.

Get consent

Formal approval of the requirements is a must. When someone comes along weeks (or even months) later with a "must-have" requirement, you need to be able to point to the formally approved requirements. This may not stop you from having to include the rogue requirement, but at least it will make the "rogue requiree" think twice. In addition, the formality of official consent helps people take their part seriously. It may be that by now people are tired of all the work that they have had to do to get this project started. You can assure them that this is the last big push for them, and smile inwardly that you have exhausted them into giving you the space you need to do the tasks ahead.

In addition to approval on the list of requirements, you should push for consent on their priorities. Unless I am sorely misinformed, your list of requirements will be longer than you are comfortable accomplishing in the limited time you have for the project. In addition, it is likely that the list you have includes items that are there because you stopped arguing or simply because someone was willing to stand up for them. By having priorities, you can let the organization decide the relative importance of the requirements and prepare the organization for not getting all required items on the first go-round.

Tip You can get your contributors to take the first cut at prioritization by asking them to mark requirements as "must have," "should have," or "could have."

In gathering requirements, your goal is to get people to think out of the box to tell you what is really needed, not just what they already have or are already doing. You want people to give you the constraints without which the CMS will not work, rather than tell you about their favorite white papers, Web sites, and development platforms. There is a fine and not-so-visible line between a desire and a constraint (especially if the "desiree" is a person with sway!). If there are disputes in the organization, especially in the platform questions, they will show up in your requirements. If you are lucky, the support and consensus that you forged in the mandate process can carry you through any difficulties here — but don't count on it.

Note I have seen more than one content management initiative stall as all of the organization's disputes were visited on the content management requirements gathering process. A software developer I once knew captured the problem with requirements and the organization most elegantly. He said, "When you pour milk on oatmeal, all of the lumps come to the surface."

Content requirements

For clarity, I will start with content requirements. Content is a good place for you to start if your organization has its business goals solidified. If not, you may get stuck by starting with content. Without a solid marketing plan, you may not know who your customers or other audiences are, or which are most important, making it extremely difficult to define and prioritize content. Without clear business goals, the simple question, "What content should we deliver?" may open a can of worms. The result may be a CMS effort that either forces those plans to be solidified, protracting the process, or an effort that pushes on without clear goals and ends up with a half-baked system.

If you find that you are not making much headway with content, try starting with the questions about publications and working back to content.

The questions behind the content requirements are quite simple:

- What information do we need to deliver?
- For each type of information, who in the organization is needed to create this information or how will we otherwise acquire it?

From these humble beginnings, you can begin to construct the entire collection system. The answers to the second question serve as much to make the respondent think as they do to provide you with information. If the respondent has a clear idea of how you will come by the information she suggests, great. If she does not, maybe question two will cause her to think a bit more before suggesting that you collect such information. You can also use the answers to question two as an aid to prioritization. If no one knows how to get a hold of a certain kind of information, then maybe the cost of collecting it is higher than the value that the content provides.

System requirements

System requirements are the set of technological constraints that the organization would like to put on the CMS.

There are a few more questions to ask to gather system requirements, and they are more specific than the content and publication questions. Following are such types of questions:

- **System integration:** With what existing systems will the CMS have to integrate? Existing systems include those connected to the management system, as well as those that provide information to the publishing system. In Chapter 7, "Introducing the Major Parts of a CMS," there is more information on management and publishing system connections.
- **Hardware and software:** What company standards for hardware and software apply to the CMS? If, for example, all of your graphic artists use Macintosh computers, you need to know that so you can make sure your collection software works on their computers.

Content requirements

For clarity, I will start with content requirements. Content is a good place for you to start if your organization has its business goals solidified. If not, you may get stuck by starting with content. Without a solid marketing plan, you may not know who your customers or other audiences are, or which are most important, making it extremely difficult to define and prioritize content. Without clear business goals, the simple question, "What content should we deliver?" may open a can of worms. The result may be a CMS effort that either forces those plans to be solidified, protracting the process, or an effort that pushes on without clear goals and ends up with a half-baked system.

If you find that you are not making much headway with content, try starting with the questions about publications and working back to content.

The questions behind the content requirements are quite simple:

- What information do we need to deliver?
- For each type of information, who in the organization is needed to create this information or how will we otherwise acquire it?

From these humble beginnings, you can begin to construct the entire collection system. The answers to the second question serve as much to make the respondent think as they do to provide you with information. If the respondent has a clear idea of how you will come by the information she suggests, great. If she does not, maybe question two will cause her to think a bit more before suggesting that you collect such information. You can also use the answers to question two as an aid to prioritization. If no one knows how to get a hold of a certain kind of information, then maybe the cost of collecting it is higher than the value that the content provides.

System requirements

System requirements are the set of technological constraints that the organization would like to put on the CMS.

There are a few more questions to ask to gather system requirements, and they are more specific than the content and publication questions. Following are such types of questions:

- **System integration:** With what existing systems will the CMS have to integrate? Existing systems include those connected to the management system, as well as those that provide information to the publishing system. In Chapter 7, "Introducing the Major Parts of a CMS," there is more information on management and publishing system connections.
- **Hardware and software:** What company standards for hardware and software apply to the CMS? If, for example, all of your graphic artists use Macintosh computers, you need to know that so you can make sure your collection software works on their computers.

- **Development environments:** What requirement or preference does the organization have for software development environments? The question is not "What platform do you prefer?" but "Has the organization or any department mandated the use of a particular platform?" This phrasing helps avoid getting back a lot of personal preferences (which, for some strange reason, you do not need any great process to find out).
- **Deployment platforms:** What requirement or preference does the organization have for publication deployment platforms? The obvious answers are about Web servers, Web application servers, databases, firewalls, and server farms. You will find this sort of information easiest to collect. Either the organization has all of that set up and prefers to continue using it, or it does not and it will be part of your job to figure all that out. Less obvious are the deployment platforms for non-Web publications. For example, what infrastructure exists or is preferred for print publications? Is there a group that really has this nailed down? What e-mail platforms are in use that can do bulk mailings (if that is part of your publication set)?
- **Performance:** What requirements are there for scaling and performance of the management or publication systems? What you are most likely to get are the current statistics for your organization's Web initiatives. What you might probe a bit deeper for, though, is how far the current systems can go before they begin to degrade. It is quite likely that you will push your systems much harder with a CMS behind them than they are being pushed now.
- **Deployment:** Are there standard rollout procedures? If you are in a large organization that has existed for more than a year or two, it is beyond doubt that you have rolled out an enterprise system before. If this rollout was done well, it will have left a written legacy of good procedures and guidelines. If you can find this legacy, you will save a tremendous amount of time and effort.
- **Maintenance:** Are there system maintenance standards? Your organization probably maintains enterprise systems now. What standards are used to measure the quality of maintenance agreements and support procedures? As with rollout standards, you can save a lot of time if you can find an existing process into which you can tie the CMS.
- **Localization:** Are there localization standards? If your organization is in more than one culture, you have no doubt already come across the issues associated with multiple languages and localization of content (localization includes, for example, making sure half-naked bodies do not show up on screens in cultures where they offend). I have seen few organizations that have dealt with this problem in a comprehensive and well-documented way. If your organization has done so, in addition to having a great career opportunity in localization, you will have a great start on this difficult issue.

Publication requirements

The questions to ask about publications are a bit more complex, but ought not to cause any huge workload for the respondents. They are the following:

- Who are the top three audiences we are trying to serve with our information?

- What categories do the top three audiences fall into?
- What other audiences would it be nice to serve?
- How do we provide information to the top three audiences now?
- How would these audiences prefer to have information provided?

By starting with the audiences, you put the respondent in the right frame of mind. Rather than telling you directly what publications they think the organization needs to create, they tell you how audiences prefer to get information. The questions nudge the respondent to begin to prioritize audiences and set them all at the same level. All respondents choose three audiences (an arbitrary low number that you might want to change). That means that they will all have to choose the most important ones to list. In addition, by asking them to choose three audiences, you force the respondents to give you audiences that are at the same level. One can't (or at least has no excuse to) say, "sales engineers, sales reps, and inside sales," while another says, "staff, partners, and customers."

In your questioning process, there is purposely a slot for people to tell you what you do now, but no slot for them to tell you what they think you should do. Rather, there is a slot for saying what the audiences would like. Too often, people will try to fit the audiences into their preconceived notions (or, more to the point, their preexisting outputs) of the right set of publications. This questioning is a not-so-subtle attempt to get people to break those molds.

As with the content requirements, these questions are designed to make the respondents think as well as simply provide information.

Approaching the organization

Sometimes the organization itself can stand in the way of completing your requirements gathering in the following ways:

- **It's difficult to establish ownership of many of the requirements.** Who is the right person to say which database you must use? Who can decide whether a particular type of content is a "must have" or a "nice to have?"
- **People don't have time to answer your queries.** This is especially true for questions that require real effort such as, "Can you sample those 1,000 files and tell me how consistently formatted they are?"
- **People will speak with authority when they have none.** When you ask a question, you may get an answer whether or not the respondent has the knowledge or position to properly answer.
- **People cannot (or will not) deal with your collection methods.** My teams have tried everything from Web-based forms and e-mail messages, to Microsoft Word templates and personal phone calls, and still people cannot understand how to respond in the appropriate way. The good angel on my right shoulder says, "It's not easy enough." The bad angel on my left shoulder says, "Lazy #$%#!"

- **People don't know how to answer your questions,** so they will stall or try to divert you by heaping information on you (such as those people who say, "I don't know what is important so I'll just give you all of it and let you decide").
- **CMS requirements tend to cross department lines,** so you end up chasing facts around all of the loops and down all of the blind alleys that your organization might have in its structure (for example, being bounced around from person to person until you are back at the person where you started).

So, what do you do? First, a small amount of common sense and organizational savvy can go a long way. Note when you are getting heaped on or put off, and then stop wasting effort. Use your mandate and the support of your sponsors to cut red tape and hold people to the responsibilities of their job descriptions. Note whose interests are being served by holding you back or stalling the process, and act on that rather than on the frustration you feel toward a particular person.

More generally, you can create a plan of attack that identifies organizational contacts and their responsibilities (the plan of attack deliverable is detailed later in this chapter).

Above all, realize that people are unlikely to understand what you are doing and why you are asking the questions you are. With patience, an attitude of respect, a desire to teach them as well as learn from them, and a good sense of humor, you can build alliances while you collect requirements that will last you through the rest of the project.

If you have built your readiness assessment as I outlined, you have no doubt already interacted with many of the people to whom you are turning now. If you performed a mandate process anything like the one I outlined, you will have all the organizational support and muscle you need to get back good, timely information.

Finally, use your mandate as a tool to set the context of the discussion. Rather than saying, "What content should we include?" say, "What content best fits the mandate we received?" Keep referring back to the mandate until your respondent (and you, for that matter) cannot help but take it seriously into account.

If all of this good advice fails you and you are still left on Thursday night with not enough information for the Friday deadline, take heart. I have been there, everyone I know has been there, and now you are there too! Welcome to the Frustrated Analysts Club!

Doing logical design

Logical design is the design of the CMS outside of the particular hardware and software you will use to implement the CMS. This is not to say that it cannot help you choose the right system or that it will not be used by the system you end up installing. It surely can and will. A logical design specifies what is to be done, and a particular CMS dictates how it is to be done.

Here are the aspects of logical design that you will have to take into account:

- **Audience analysis:** This details the kinds of people you will serve. In Chapter 21, "Cataloging Audiences," you will find the full description of audience analysis.
- **Publication design:** Here you define what content and navigation each publication will include and exactly how they will automatically be built and personalized by the CMS. In Chapter 22, "Designing Publications," you will find the full description of publication design.
- **Component design:** Here you determine the complete set of content components you will manage and exactly how each one will be constructed. In Chapter 23, "Designing Content Components," you will find the full description of content design.
- **Author analysis:** Here you figure out whom you need as content authors and how you will serve them. In Chapter 24, "Accounting for Authors," you will find the full description of author analysis.
- **Source analysis:** Here you decide where you will acquire information and how you will process it to make it ready for the CMS. In Chapter 25, "Accounting for Acquisition Sources," you will find the full description of source analysis.
- **Access structure design:** Here you define the hierarchies and other access structures you will need to keep your content organized in its repository and to produce the navigation you will need in publications. In Chapter 26, "Designing Content Access Structures," you will find the full description of access structure analysis.
- **Workflow and staffing design:** These specify the kinds and numbers of jobs and tasks that will be needed to start up and run the CMS. In Chapter 27, "Designing Workflow and Staffing Models," you will find the full description of workflow and staffing analysis.

These various analyses and designs do not stand alone. Rather, they overlap and inform each other in a large number of ways.

Cross-Reference In Chapter 19, "The Wheel of Content Management," you will find an overview of all of these analyses and a discussion of how they interact.

Why do logical design?

Most people underestimate the amount of information you need to accumulate to fully define a large CMS. In fact, many organizations skip the stage of defining their requirements for the CMS entirely and simply begin using whatever software they have purchased, trying to make it produce pages like the ones they have now.

Contrary to what you might expect me to say, I don't believe that this is a bad approach. Given that you have a long time to play with a system, experiment with what it can do, and determine what it requires of your organization, this method can give you a very good start. It can teach you all the basics of content management and give you practical experience with a real tool.

On the other hand, there is a big leap from a system that can produce some pages like the ones you have now, to one that can organize your entire production and distribution process. You can't expect the latter process just to happen by turning on and using even the best CMS.

Other organizations rush to buy a tool and then begin an analysis. Again, this can work. By immediately imposing the constraints of a particular product on the project process, you get to use a lot of the methods and tools that the CMS product company might provide. In addition, you can show immediate results and hit a big milestone quickly. On the other hand, if you buy before you study, you risk the very real possibility of having purchased a (possibly very expensive) system, only to find after using it for a while that it was the wrong one for your needs. Finally, by choosing a tool before you really know what you want to do, you may end up not doing something that the tool does not readily do simply because it is off the tool's radar screen (regardless of how much value it can have for your organization).

If you have followed the steps I have laid out so far, you already have spent a lot of time working through the issues and should have a lot of detail about what people want and require of the system. You might think that what you collected during the readiness assessment and mandate process ought to be enough to define the system. Maybe it is, but probably it is not. What you have collected so far certainly establishes what people want, but does it define what your organization needs? At the very least, you will have to organize and augment these wants and get the organization's departments to agree that they are complete. More likely, you have a good start and a lot of general statements but nothing specific enough on which to base a system design.

If you take the trouble to do a complete logical design, it will help you significantly in the following ways:

- **System selection:** In logical design, you say exactly what you want your CMS to do. If you already have a CMS package in mind, your planning team can use the design to decide the degree of fit between the selected system and the design. If you have not yet selected a system, the design will give you a clear picture of the system you want. You can compare your picture to the one presented to you by the product companies you review. You will also be able to ask intelligent questions of the CMS product vendors and not be overly swayed by slick presentations. Finally, if you choose to develop a custom system, the logical design provides great input for your software requirements specification.
- **System implementation:** To implement a CMS, you create a set of specifications. The specifications combine what you want to do with how you will do it. The logical design is what you want to do. The features and functions of the CMS you will build or buy are how you will do it. If you do not get a clear idea of what you want to do, the features and functions of the CMS you use will decide for you. A logical design puts reality into your specifications by making sure that the features and functions it defines are seen in light of the exact content and publications you want the system to manage.

- **System rollout and maintenance:** The logical design should provide just the right starting point for many of your system deployment deliverables. For example, the logical design describes in detail all of the staff that will contribute to the CMS. These are the same people to whom the system must be deployed. In addition, much of the documentation you will need to train users and maintain the system can come directly from the logical design deliverables.

From business to system

I'll assume that you are starting this phase of your project with a hierarchy of business goals, a mandate statement, and a set of requirements (see Chapter 14, "Securing a Project Mandate," section "Taking Stock of the Deliverables," for more information on coming up with these items). You can use these goals and requirements to make your way into a design. Figure 15-1 illustrates the hierarchy of business goals — a mandate statement is at the tip of an organized set of business goals; a content domain statement is at the same tip of an organized set of publications and content components.

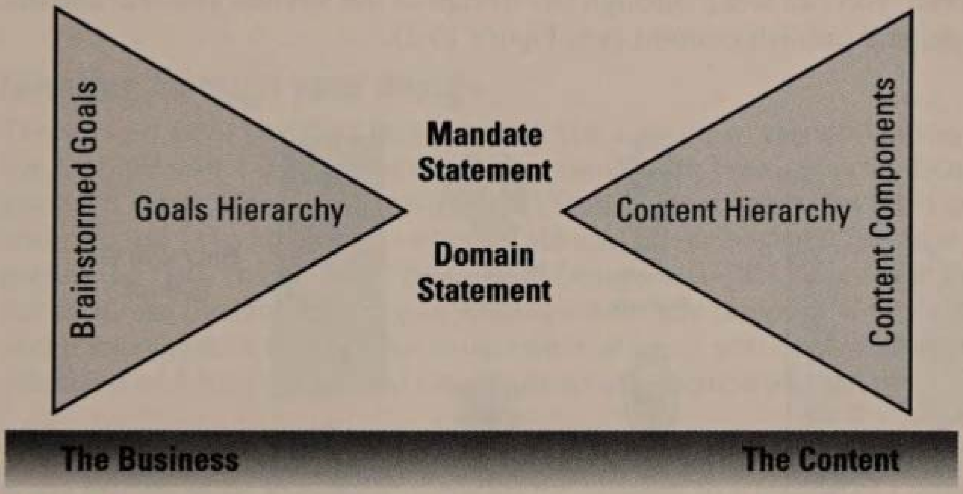


Figure 15-1: The hierarchy of business goals

Your mandate ought to lend itself to rephrasing as a content domain statement by asking the question, "What information and functionality do we need to deliver to meet this aim?" Then, from the single domain statement, you should be able to start constructing an outline of the kinds of content that belong in the domain (the content hierarchy). In this way, you can pivot naturally from the knowledge of the business you have gained in the mandate process to knowledge of the content you will need to provide.

Cross-Reference

In Chapter 26, "Designing Content Access Structures," in the section "The content domain," you can find out more about the content domain statement.

Tip When you construct a content hierarchy, try not to be overly influenced by what you already know about the content that people plan to deliver. If you can work strictly from the mandate statement and then the domain statement, you will have an independent means of evaluating what belongs in the system. Inconsistencies between your outline and what people have asked for in their requirements are a good thing. Use them to rework your content hierarchy, your domain statement, and even your mandate statement (if you dare), or to be able to say with confidence why the proposed requirement does not belong.

With a good content hierarchy in hand, you can begin to decide what sort of content components you will need. This is only a beginning, however, because you will work and rework your list of components throughout the design process. (See Chapter 23, "Designing Content Components," for much more information on content components.) Loosely, each terminal node (the ones with no children) of the hierarchy is a potential component class.

With the preliminary list of component classes, you are ready to begin the real work. From the content you know you have to deliver and the requirements you have gathered, you can work through the design of the system you will use to collect, manage, and publish content (see Figure 15-2).

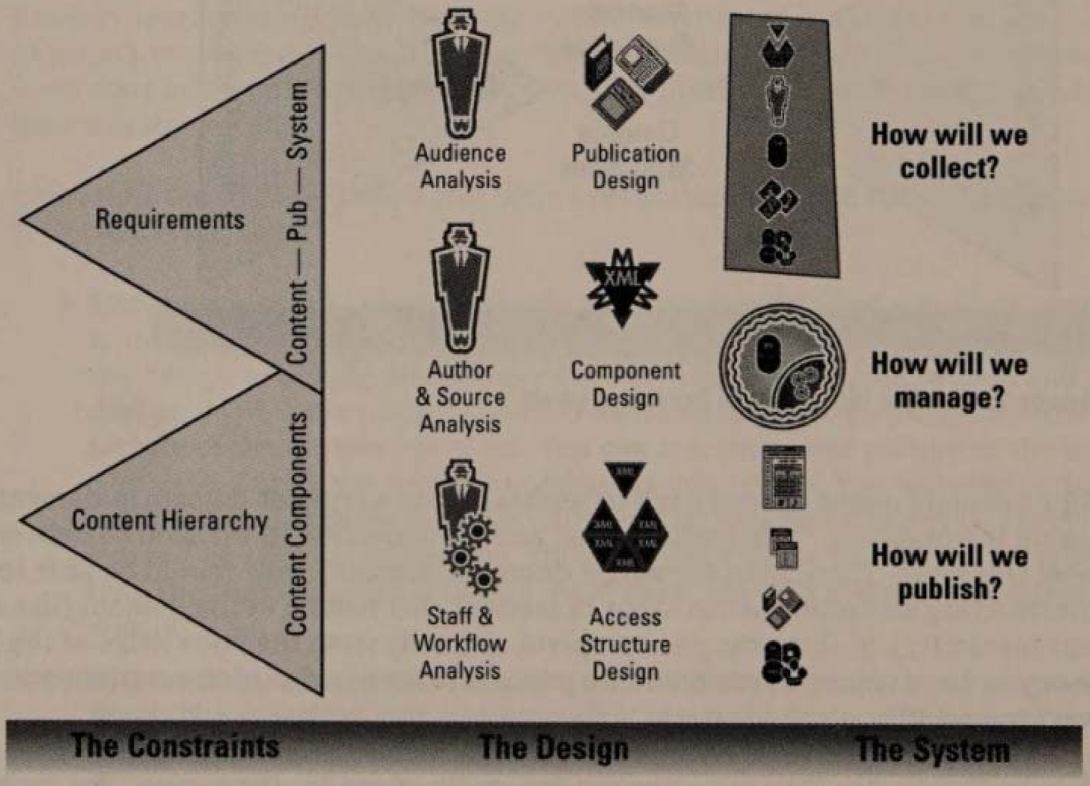

Requirements
Content — Pub — System
Content Hierarchy
Content Components
Audience Analysis
Author & Source Analysis
Staff & Workflow Analysis
Publication Design
Component Design
Access Structure Design
XML
How will we collect?
How will we manage?
How will we publish?
The Constraints
The Design
The System

Figure 15-2: Your requirements and the content hierarchy are the starting place for your design.

Your independently created content hierarchy and the requirements you have gathered are the starting place for design. After you have reconciled these two, you can begin to perform the in-depth logical design of your system. Logical design answers the myriad questions about what you want the CMS to do. You can organize the answers you find into the usual three categories: collect, manage, and publish (for presentation back to the organization).

As you can see, a lot is going on here. You can go from business goals to a preliminary list of component classes, all on a piece of paper in a coffee shop. To go from the preliminary component classes to the final classes and the system you will need for them, however, is another story. You will need to know much more about your audiences, publications, information sources, staffing, workflow, and the access structures you will use to manage components before you can begin to implement the system.

I'm sure you are not surprised to know that there is a lot of work to do. I hope that this overview shows you, however, that the work is neither insurmountable nor unknown. Keep these diagrams in mind (or at hand) as you dive into the details of design.

Iterating through your design

There is no clear dividing line between the amount of categorization you need and the amount you can do. (For example, should you break your audiences into 3 categories, 9 categories, or 27 categories?) Frankly, you can go on forever, collecting and refining your approach to your CMS and never begin to feel that you have completed the task. In Part IV, "The Logical Design of a CMS," I present a comprehensive list of all the design criteria you should eventually uncover. If you simply dive in and try to capture all of these constraints at once, you may get bogged down and confused and then run out of time with an incomplete analysis.

The solution is to pass many times through the same set of constraints, each time going deeper than the last and each time coming to a feeling of completeness at some level. When you run out of time, you may not have all of the constraints worked out, but you will have an even and complete amount of detail in all areas. In other words, approach constraints in a series of deepening iterations. Here is an example of a five-pass process that you can use as a starting place to design your own set of iterations:

- **Pass 1 — Preload:** Review all of the documents and notes you have collected so far and use them to create a general statement for each category of design (audiences, authors, publications, and so on). For example, your analysis so far might lead you to say, "We will have about 30 authors, all of whom will be inside the organization." Use your best judgment to create these statements and note any gaps in your knowledge that cause you to have to guess.

- **Pass 2—First cut:** From the large number of questions that I provide in Part IV, "The Logical Design of a CMS," decide on no more than 3 or 4 questions that you will need answered in each category to feel that you understand the major issues in full but at a high level. For example, you might divide the 30 authors you came up with in Pass 1 by their savvy and their closeness to your team. Take no more than a few days to get good answers to those questions, and then create a short but comprehensive design report. Get immediate feedback on the report to ensure that you are in synch with the project stakeholders.
- **Pass 3—Design detail:** Begin to fill in the detailed design constraints in each category (presented in Part IV, "The Logical Design of a CMS") by prioritizing the questions that are most important to get answered most quickly and then acting on the highest-priority questions first. In this way, your details will come together in waves from most to least importance. You can break this pass into smaller increments, each with its own tight deadline, to further organize your approach. This pass ends when you are at the point of diminishing returns (that is, it is too much effort to drill down any further at this point). You undoubtedly will have important unanswered questions, but you must resolve to push on while you await responses.
- **Pass 4—Triangulation:** One design constraint does not live in isolation from all others. More often than not, you will discover conflicting constraints or ones that force you to reconsider and perhaps deepen your analysis. This is a good thing. It means that you can play one constraint off others to make them all stronger and more consistent. In this pass, you can focus solely on the how the different classes of design constraints affect each other. See the following section for more information on triangulation.
- **Pass 5—Finalization:** In this pass, you go through the entire set of constraints (which might be quite large by now) to finalize them for approval. It is likely, however, that while you are doing this, the project can begin to move forward into design and specification.

Triangulating on constraints

To find the exact location of a point in the landscape, a surveyor triangulates. First, she measures the compass bearing to the point from one location, and then she measures the compass bearing to the point from a different location. After measuring the distance between her two locations, she can draw a triangle that tells her the exact location of the point in question. If she wants to be more precise, or check herself, she can use more than two locations. The more locations from which she takes measurements, the more sure she is that she knows where her target is.

To find the exact composition of the content components you must create and manage, you too can look at them from different perspectives, as shown in Figure 15-3. For example:

1. **You can look at your components from the perspective of the collection requirements:** What kinds of components are your authors able to create? What kinds can you acquire?

2. **You can look at your components through the lens of management requirements:** How will you determine when it is time to archive or delete a component? What indexes and storage requirements will they have?
3. **You can survey your components from the vantage point of your publications:** What kinds of content belong on each page? What sorts do our audiences expect? What will we need to know to personalize?

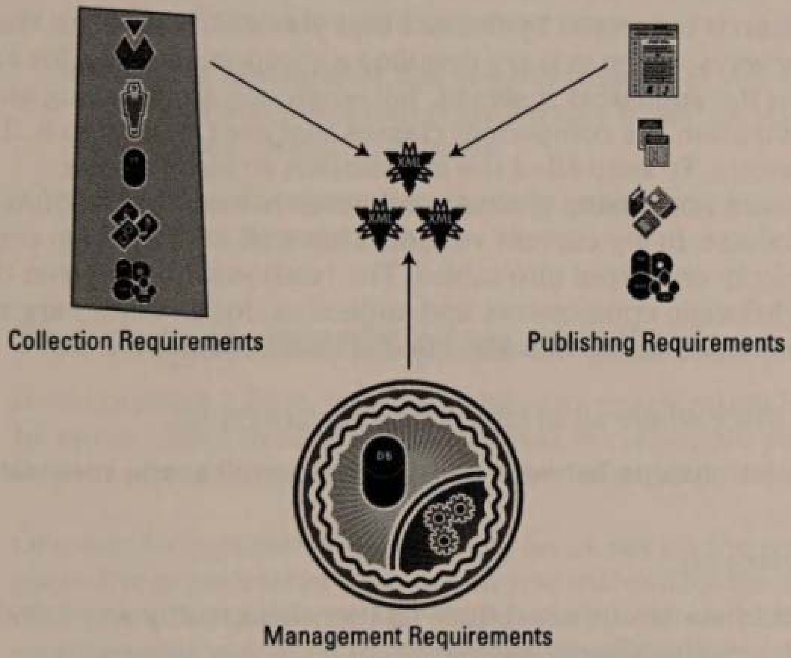


Figure 15-3: The more ways you use to define content components, the more sure you can be about them.

By addressing the same target entity from a number of perspectives, you can be sure that you have accounted for all its parts. Of course, the same notion works for any other part of the system. For example, I could have triangulated on publications by looking at them from the perspective of the components, the audiences, and the sources. Any combination will work.

Triangulation, or cross-correlation, as it is more properly called, is more than just an interesting exercise; it is the way to get at and solidify the requirements and design of your system.

Cross-Reference

In Chapter 19, "The Wheel of Content Management," I provide a more complete picture of the various aspects that must be cross-correlated in a CMS, which I call "entities."

Managing the details

Don't be surprised if you end up with hundreds of pages of data when you have fully fleshed out all of your design. For each component class (and you may have dozens), you might need three or four pages to detail its elements, element values, source information, target, audiences, the publishing templates where it shows up, and the personalizations that access it. For 25 types of components, that is already 100 pages!

The volume of information is increased by the fact that you will need to cut the same information many ways. When you are detailing a component class, for example, you should mention the audiences it serves, but when you are detailing an audience, you need to mention the component classes that are targeted to it. The combinations are numerous. To keep all of the information straight, I have employed a variety of word processing systems and spreadsheets, but nothing has worked as well as a database. In my current version, I have all of the design constraints I can think of nicely organized into tables. The relationships between the classes of constraints (between components and audiences, for example) are specified in database relationships between tables. The database enables me to:

- Have one central place where all of my constraints are typed
- Account for the relationships between constraints as well as the constraints themselves
- Find and sort constraints
- Slice through a set of constraints and their relationships in any way I desire without needing to continually cut and paste
- Produce well-formatted reports

My approach may be overkill for you, especially at the beginning of your project, but keep in mind that you will eventually have all of the same needs that I listed. So, whatever system you use to type in and keep track of constraints, it needs to be robust.

Taking Stock of the Deliverables

You can use the following deliverables to form the basis of your requirements and logical design project:

- **The requirements plan of attack** charts how you will get to all of the people with whom you need to talk in your organization.
- **The requirements document** solidifies and makes official the requirements for the CMS project.

- **The audience analysis** presents your selected set of audiences for comment and acceptance by the organization.
- **The design plan of attack** charts how you will move through the logical design process.
- **The collection, management, and publishing design documents** summarize the results of your extensive logical design process.
- **The localization plan** shows what you consider the most practical approach to localization for your organization to understand and approve.
- **The risk assessment and project plans** that you developed earlier can be updated and deepened considerably based on your requirements gathering and logical design processes.
- **An executive summary** can help you inform and continue to receive support from your organization.

The requirements plan of attack

If you prepare a little, your foray into the organization to collect requirements will be much easier. In addition, in the heat of collection, you will be able to refer back to your plan and not have to remember what to do next.

One way to approach this plan is to focus not on the requirements but on the people in the organization from whom you will collect them. My assumption here is that, given the questions I have laid out, you already know enough about the requirements you need to capture. On the other hand, I assume that you need to know more about how to collect them from your colleagues. To work this through, you can create a spreadsheet that lists your requirements contacts. Before you send out any questions, get together with each contact and define the following:

- **Which questions will you ask?** You might want to number your questions to make it easy to refer to them.
- **What kind of turnaround time** will you expect on answers?
- **What kind of review** of your synthesis will she require?
- **Who would be the right person to contact to escalate** any responses to questions that she is not comfortable answering to someone of more authority?
- **Who are the other people who should answer the same questions**? You can compare her answers to this question to the people you have targeted to be sure that you have not missed anyone and that she is aware of the other people you have chosen for each question. It is better not to surprise her later.

With this task complete, you can create a second spreadsheet that focuses on requirements. For each requirement, you can enter the following information:

- The names of the respondents
- The overall turnaround times for responses
- The review process you will use for the requirement
- The arbitration process for conflicts in answers

The second spreadsheet is a plan of attack for gathering requirements.

Note The length of this process is another reason I boil requirements down to so few questions. You could not practically do this sort of process with dozens of questions.

The requirements document

You need an official result of the requirements process to hang next to your project mandate. When the arbitration is finished on the last question, I suggest that you gather the final results into a document and get specific sign-offs from the following people:

- The respondents to the requirements questions
- Your sponsors or their assigned point persons
- Your project team

I believe that it is worth making this much fuss over the requirements document because it is the last of what you will expect the organization to provide to you for a while. As such, you want to make sure it gets full closure and that the signatories know that their chance for major input is now done. In addition, these requirements are your true marching orders. If the mandate is the goal of the war, the requirements are your orders. What is left, of course, is a battle plan.

Be sure to include in your requirements document clear references to the mandate. You want to be sure yourself first, and then communicate to others, that the requirements derive clearly from the mandate. For your own sake, as well as the effect it will have on others, be sure that this is actually true. Later, it may be too late to reconcile those nagging little inconsistencies between the overall goals and means you have chosen to achieve them.

As you might have guessed, my major organizing principle for all content management materials is collect, manage, and publish. To the extent that you can organize your requirements document with these three categories, it will clearly tie into the later deliverables as I describe them. However, you needn't go too far with this. The requirements document is the pivot point between the world of the organization and that of the CMS. The categories content, system, and publications that I presented earlier map very loosely to collect, manage, and publish but are about halfway between the language of the organization and that of the CMS. They may be a bit clearer to the audience of the requirements document than collect, manage, and publish are.

The design plan of attack

There is a lot to do to fully develop and document a logical design. (It is for good reason that I devote so many chapters to it.) In planning for requirements, you could get away with a couple of spreadsheets. In planning for design, you had better trot out the best project management tools you have. Up until the end of requirements, the project is relatively small. Starting with design, it can begin to explode. It is the number of details and the number of relationships between details that cause such a large expansion of the project workload during logical design.

It behooves you to get organized early in the logical design phase and — stay that way. The result of not being organized is a longer duration for this phase of the project and a lot of missed details and relationships that may surface later in the project and haunt you.

The main things to account for in your design plan of attack are the following:

- **How you will collect and manage** the detailed information you will gather. As I mentioned earlier in this chapter, if you can manage to get a database together, it will repay the effort.
- **Who will help gather** and process the information. Obviously, the same people who provided requirements will be of assistance. Be careful, however, not to count on people with a lot of other responsibility to do substantial work for you. If you don't already have some, this may be the time to get some staff of your own.
- **How you will time the data collection** and processing. As one example in many, it is just as valid to work backward from publications to components as it is to work forward from components to publications. In either case, how will you ensure that in the end, components and publications are completely tied together? I mentioned the techniques of triangulation and iteration earlier to help you find ways to figure this out. In your plan, it is not enough to say that you will iterate, you must say how long each cycle will be and what exactly it will accomplish.
- **What you will deliver.** I'll provide a general framework for the design deliverables, but it is up to you to turn it into a solid plan.
- **How you will deal with overload.** Be prepared at the start for the amount of detail to overwhelm you. As you move forward, how will you reset expectations on required content and publications if you can't do them all? Don't wait until you have run out of time to confront this issue.

I prefer to use a project Gantt chart to show all of the milestones to be accomplished in the design process (see Figure 15-4).

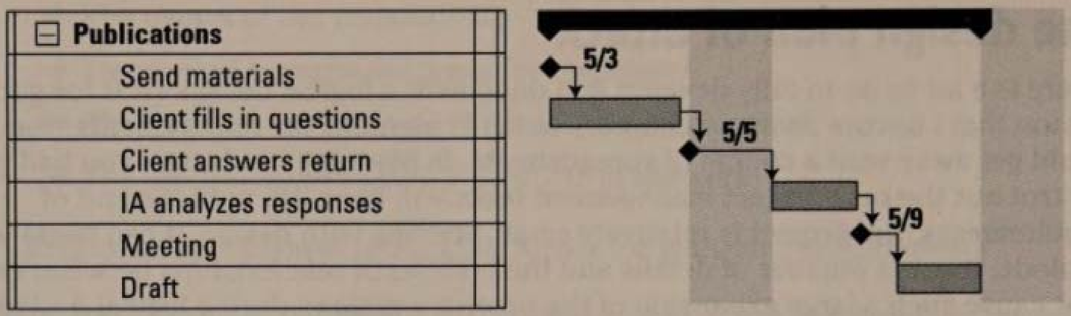


Figure 15-4: A Gantt chart can help you stay organized and on target during design.

Based on what is most convenient for your team and organization, you can create one big logical design document, or a number of smaller ones that fit together as a set. As you might expect, I will organize the information to be presented in three parts: collection design, management design, and publishing design.

The collection design document

To logically design your collection system, you decide and then document how content will move from outside the CMS to the point where it is ready to be used in publications. Following my discussion of what occurs in a collection system, the collection design document will have to cover these topics:

- **Who are our authors** and how will they be tied into the CMS? (More information on this is in Chapter 24, "Accounting for Authors.")
- **What are our acquisition sources** and how will they be tied into the CMS? (More information on this is in Chapter 25, "Accounting for Acquisition Sources.")
- **What components will be created** by which authors and sources and at what rates? (More information is on this in Chapter 23, "Designing Content Components," in the section "Plan.")
- **What conversion processes will we need** to transform the format and structure of the information we receive? (More information on this is in Chapter 23, "Designing Content Components," in the section "Plan.")
- **What staff will be needed** to collect content and what will their tasks and processes be? (More information on this is in Chapter 27, "Designing Workflow and Staffing Models.")

You can use the information provided in the respective chapters that I reference in the preceding list to decide what you need to say about these topics. As far as how to construct the document, consider its audiences. First, the staff it mentions will want to read this document. It should therefore have well-marked sections that divide responsibilities by position name. Next, specification writers will want to read it to understand how to set up the collection systems. Thus, you should make sure that it divides the information by functional units (for example, authoring tools, conversion systems, acquisition procedures, and so on). Finally, it will be used as input to the system selection process, so it should call out any information that should be included in the selection process.

If you are starting to think that you need a CMS just to produce the various cuts on the design of your CMS, you are on the right track.

The management design document

To logically design your management system, you decide on and then document your approach toward content components. The design has the following basic kinds of information:

- **Information about components,** including which ones you will create and how they will be meta-tagged. I call this the content schema.
- **Information about how components will be cataloged and accessed,** including the hierarchies and indexes that the system will use to track and publish components.
- **Information about how the components will be managed in the CMS repository,** including how you will track them through their lifecycles and eventually retire them.

The content schema

A content schema divides content into components and components into elements. The schema specifies the following:

- **The set of components** to be managed.
- **The set of elements** for each component that includes both content and management elements.
- **The allowed values** for each element.
- **The subset of the elements that are shared** between a number of components to keep consistent naming. For example, you might have an element called "Illustration" that is used in any of the components that require a demonstration image.
- **The subset of elements that are universal;** that is, shared between all (or almost all) components that will be used for accessing and managing them. For example, you might decide that every component will have a "Status" element so that you can track its lifecycle stage.

To construct a content schema, you divide and organize your information until every part is accounted for and you can confidently say that you have a workable system of content and management elements and the components of which they are a part.

Cross-Reference For more information on the content schema, refer to Chapter 23, "Designing Content Components," in the section "Analyzing components."

To deliver the schema, you document the components, elements, and values. A spreadsheet or word processing document may suffice. If you use these methods, your problem will be how to cut the same information in different ways. For ease of use later in the process, you will want to deliver the information sorted by component as well as by element. This is cumbersome but possible in a spreadsheet application, but a real pain in a word processor. For that reason, I tend to use a database application where I can switch easily between component and element views of the information.

Just as components are the center of the content management process, so the content schema is the center of the logical design process. This means that it depends a lot on the other parts of the design. Therefore, don't expect the schema to come out quickly and stay the same. Rather, it will come out slowly and evolve throughout the logical design process. It will continue to evolve (though much more slowly) throughout the implementation process. Even after your system is running, you should expect the schema to change as you respond to the changing needs of your information marketplace. At that point, of course, it is maintenance, not design.

The content schema is the starting place for both the schema that you will need to implement your system and the metatorial guide that you will need to produce for system deployment.

Note If you know that you will be using a relational database for your repository, you can deliver the schema to the implementers of the system as a database design. If you will be using an XML system, you can deliver it as an XML DTD or schema. If you do deliver using relational database or XML methods, you will also need to create some sort of document to deliver notes, caveats, or values, and relationships that can't be well represented in these tools.

Component access

A big part of the logical design of the CMS is to figure out exactly how to catalog components so that you can find them and automatically generate the navigation of your publications.

Cross-Reference The concepts and practice of access design are detailed in Chapter 26, "Designing Content Access Structures."

The feature that you cannot do without in this section of your management design document is the component hierarchy. The hierarchy is an outline that categorizes all of the components you will manage. By creating this hierarchy, you are sure that all of your components fit together in a single access system. It is equally important to show how this overall component hierarchy can map to the hierarchies of each of your publications. If your system must produce components in multiple languages, you can show the relationship between the versions in this hierarchy.

You can also include the following sections in your management design document:

- **Indexing terms.** It is less important that you actually come up with an exhaustive set of terms than that you come up with the first list and also decide how the master list of terms will be generated (when and by whom), as well as how to translate the master list into the indexes that you include with any publication that uses an index.
- **Cross-reference strategy.** Here the point is not at all to create any cross-references, but rather to decide on general policies for when they are needed, how they will be created, and how they will be allowed to be represented in each publication.
- **Sequencing strategy.** Again, you will not be in a position to actually create any component sequences, but rather to create the policy for how sequences will be represented.

Component administration

In addition to putting together a component schema, during logical design, you need to decide on and document the staff, processes, and tasks that components will have assigned while they are in the repository. These processes include the following:

- Periodic editorial reviews
- Periodic metatorial reviews
- Archiving procedures
- Versioning procedures
- Deletion procedures

Cross-Reference

You can find more information about these reviews and procedures in Chapter 27, "Designing Workflow and Staffing Models," in the section "Clustering skills and tasks into jobs."

The form this deliverable takes is up to you, but it should match the way you document staffing and workflow in the other parts of the logical design.

An audience analysis

The audience analysis itself is a natural part of logical design (fully covered in Chapter 21, "Cataloging Audiences"). In addition to the data you gather on audiences and how it plays into your other analyses, I think it is a good idea to produce a specific deliverable on audiences for the following reasons:

- **Perspective:** The conclusions you reached on audiences ought to be out in front of the project, giving the rest of the organization the same perspective it can give you. You want the members of your organization to think first about the people served by the CMS and only then about their own perspectives.
- **Purpose:** For many organizations, there is value simply in stating what the core audiences are. If there are disagreements and different audience sets in use throughout the organization, your report may serve as the one point of officially sanctioned segments. Others can adopt it, or at least orient to it, as they work though their own audience analyses.
- **Use:** In addition to the list itself, others in your organization are bound to find the information you have collected about audiences at least interesting, if not incredibly useful. If you do a good job, you will save the organization from the recurring expense of doing the same research over and over again.
- **Feedback:** Of course, the information flows both ways. By publishing and publicizing your audience analysis (under a more enticing name, please), you will elicit feedback from competing initiatives as well as from closet experts who know more than you could ever hope to about some aspect of some audience.

In most organizations, the work you do to analyze audiences will be ahead of the pack. Other analyses are likely to be either less broadly focused or not as thorough. Your analysis needs to be both broad and thorough to fully power a content management initiative. On the other hand, because your success depends so much on the audience analysis you have done, you need to protect it from frivolous attacks from competing schemes. If your analysis is well known and supported, other schemes will have to prove merit, not just support, to challenge it.

In the end, what you want is not to be the one to define the organization's market segmentation, nor be constrained by audience definitions that are inconsistent with your other plans. You want the CMS initiative to be a player in the ongoing determination and refinement of your organization's constituencies. It would be less work for you if someone could simply hand you a good analysis already done. On the other hand, the CMS perspective can deepen and make more tangible the analyses that others have done.

The publication design documents

Publication design is somewhat more involved than collection or management design. Because each publication has its own interface (even books have a paper interface) and visual design qualities, there may be quite a bit to do to fully design even one publication, let alone the variety of publications that a CMS may be called upon to create.

Note It's beyond my expertise to offer process or advice on the aspects of publication design that are outside of the content management arena (graphic design, branding, styling, look and feel, and the like). Instead, I'll focus on parts of publication design that are directly created by the CMS (for example, not the nature of the branding, but how branding elements created outside the CMS will be laid out on a publication page by the CMS).

In publication design, you define and prototype the pages of the publications the CMS will be called upon to produce. The goal of publication design is to create the most compelling publications possible that can still be efficiently produced from the CMS. Ideally, publication design should begin at the same time, or even before, you begin collection and management design. Collection and management design ensure that a viable system can be created behind the publications, and publication design ensures that the system produces compelling output.

The deliverables from the publication design process can be a large number of files rather than the few files that result from a collection or management design process. These deliverables include the following:

- **A publication strategy document,** which specifies how you will deal with your publications in general
- **Page design files,** which define, at increasing levels of detail, how the pages will look and be built
- **Personalization strategy documents** that show how you intend to target and deliver personalized content
- **A publication administration document** that details who will be needed to create publication on a start-up and run basis, what tasks they will have to accomplish, and how frequently they will have to accomplish them

Publication strategy

Although each publication is its own world with its own constraints and design, the set of publications you will create need to hang together as well. To figure out how individual publications fit into the whole, you can create these deliverables:

- **A component-to-publication analysis:** This document tracks each component from its neutral state in the CMS repository to its display in various publications. It ensures that you know how each component will be used in the final publications and that there are no conflicts between them. The core of this document is a large table (word processing, spreadsheet, or, preferably, a database report) that summarizes all of the relationships between publications, pages, and components.

- **A template design document:** This document describes the overall approach you need to take to your publication templates. It includes a discussion of any sharing or commonality there will be between publications, as well as sections for the templates for each separate publication. In the publication-specific sections, the names of the page templates to be used are listed, as are the static text, component, and navigation templates that will be included in each page template. Finally, for each template mentioned in the document, there is some definition of the logic that the template will have to include in order to access and assemble its content properly. (There is more information on template design in Chapter 22, "Designing Publications," in the section "Analyzing Publications.")

Page design

Page design concerns the look, layout, and building of publication pages. The deliverables for this process that have the most bearing on a CMS are as follows:

- **Page wire frames,** which are schematic drawings of pages with boxes where the content will go. The wire frames name the various areas of the page and show their overall layout and size. Because of their simplicity and lack of detail, it is often a good idea to do a set of wire frames as a first step in your publication design, to get an overall view of the pages you intend to create (see Figure 15-5).
- **Page mock-ups,** which are illustrations of completed pages that are produced so that people can get an early view of what a final page might look like. Sometimes called comps, these illustrations enable you to see an average example of the pages in your publication without going though the effort of actually producing them. Be certain, however, that your page mock-ups adequately reflect the diversity of content that may appear on a page of a single type.
- **Page prototypes** are page mock-ups done in the actual publication medium. For example, an HTML prototype would look the same as the corresponding page mock-up. The page mock-up, however, is only an image, whereas the HTML prototype will actually run in a browser, just as the final site will. You can use page prototypes to prove that your designs will work and also to test the variability of content that may find its way onto a page. For example, if you have decided that you can have from 1 to 10 components on a page of a particular type, you can quickly add components to and subtract components from the prototype page to make sure the final page will always look all right. Prototype pages created as part of the logical design process do not generally include any logic or other CMS programming code. They are simply there to make sure that pages can be created in the target environment as desired.

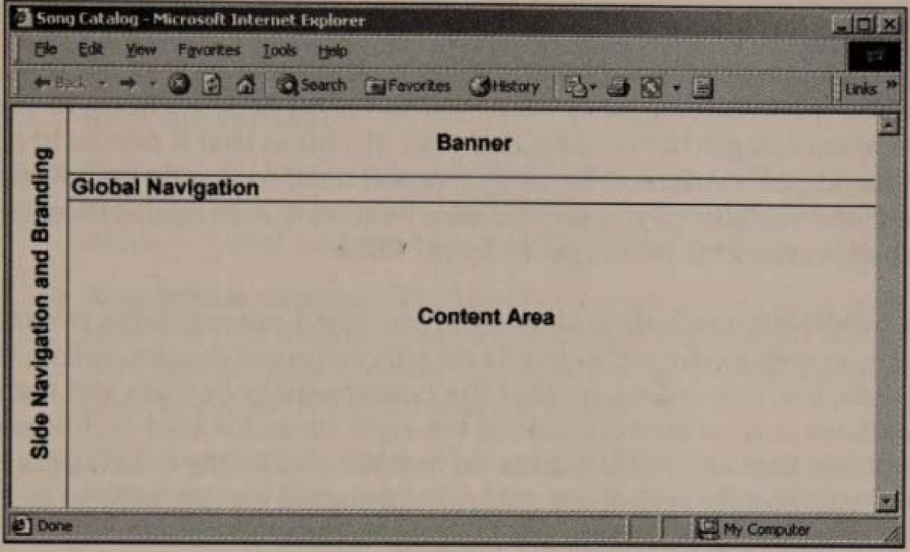

Song Catalog - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Links
Banner
Global Navigation
Content Area
Side Navigation and Branding
Done
My Computer

Figure 15-5: A very simple page wire-frame diagram that shows the locations and sizes of the areas of a Web page in the context of a browser

Cross-Reference Look in Chapter 22, "Designing Publications," in the section "Publications and Templates," for a lot more examples and information on page design.

Personalization strategy

Even if you will have only a couple of audiences for starters and a static Web site as your only publication, it is worth your while to study and document an approach to personalization. This approach can include the following deliverables:

- **A messaging strategy:** This document combines audiences, publications, and components together at a high level. It defines what each audience is supposed to get from each publication and which components and publication design features will be used to communicate with them. It is a high-level precursor to personalization analysis. Particularly for publications targeting external audiences, it provides a marketing context within which to do personalization. For internal publications, your messaging strategy clearly identifies which types of information are appropriate for each audience. (There is more information on messaging analysis in Chapter 22, "Designing Publications," in the section "Messages.")
- **A personalization strategy:** This document goes into detail about exactly how you intend to do personalization in each publication. It specifies a personalization rule for each combination of publication, page, audience, and component. The core of this document is the set of personalization rules. The rules state how to fetch components from the CMS repository based on what you know about the user, plus the values of the component's management elements. The rules also say which elements to display and how to lay out the components on the publication page. (See Chapter 22, "Designing Publications," in the section "Analyzing Personalization," for a more detailed discussion.)

Publication administration

If you have more than one publication depending on the same base of content, it is important that you work through all of the scheduling and dependencies to ensure that each publication can get the content and staff attention that it needs. In addition, as part of the logical design of the CMS, you will need to decide on and document the staffing and workflows you might need. Following are the documents you will produce when working on this aspect of your CMS:

- **A master publication schedule** shows the proposed release dates of each publication, as well as the components on which the publication relies. Using this schedule, you can make sure that the component collection and management workflows you create conclude at the right times for each publication to be released and that two publications do not put conflicting constraints on component completion schedules. (More information on this subject is in Chapter 27, "Designing Workflow and Staffing Models.")
- **A publication staffing and workflow plan** outlines the jobs, tasks, and workflows you expect the publication process to require. (More information on this subject is in Chapter 27, "Designing Workflow and Staffing Models.")

A localization plan

Content and publication localization is difficult. It can add orders of magnitude to the budgets of systems that do not handle it well. Even for the most efficient system, localization can expand schedules and budgets enormously. Expectations for localization tend to be unrealistic, either way too high or way too low. If you plan to localize, it is crucial that you define and set reasonable expectations for what will occur and when.

Cross-Reference

You can find a full account of localization in Chapter 21, "Cataloging Audiences," in the section "Audiences and Localization."

Logical analysis is the right time to create and publish a localization plan. It is late enough in the project that you will have good information to base it on and early enough that you can debate and then finalize a plan in time to implement it.

The core of your localization plan is the set of localities that you plan to support and how you intend to support them. Because localization cuts across all of the parts of logical design and implementation, be sure you cover each locality from all significant angles, as follows:

- **Audiences:** Will you create audience types for each locality or will you create local segments within each audience type? (See Chapter 21, "Cataloging Audiences," in the section "Audiences and locality," for more information.)
- **Publications:** How will navigation differ from locality to locality? Will you replicate templates or will you branch within them? What modifications are needed to your personalization approach to have it work across localities? (See Chapter 22, "Designing Publications," for more information.)

- **Components:** How will translated content be stored? Will you need any extra component elements to account for locality differences? (See Chapter 23, "Designing Content Components," in the section "Locality in components," for more information.)
- **Authors:** Do local authors constitute a different author group or can they be grouped with other authors? (See Chapter 24, "Accounting for Authors," in the section "Local authors," for more information.)
- **Acquisition sources:** What source material originates outside the core region? How will that information be processed and merged with content from the core region? (See Chapter 25, "Accounting for Acquisition Sources," in the section "Local sources," for more information.)
- **Workflow and staffing:** What are the specific tasks involved in localization? How long will they take and how many and what kinds of people will need to be involved? (See Chapter 27, "Designing Workflow and Staffing Models," in the section "Fitting Localization into Your Workflow," for more information.)
- **Access structures:** Will structures be simply translated or will they be localized as well? (See Chapter 26, "Designing Content Access Structures," in the section "Plan," for more information.)
- **Deployment:** How is locally originated content distributed throughout the system? How will local and global content be merged and deployed to the appropriate servers throughout the world? (See Chapter 16, "Selecting Hardware and Software," in the section "Deployment," for more information.)
- **Support:** What support for translation and localization do you expect from the CMS that you choose? (See Chapter 16, "Selecting Hardware and Software," in the section "Localization," for more information.)

A fair accounting on all of the preceding issues ought to give you the information you need to gauge the magnitude of the localization effort before you. Be sure that this analysis makes it out of the depths of logical design and into the hands of your sponsors or other decision makers so that they can either fund localization at the level it deserves or scale back the organization's expectation of how much localization will occur.

A risk assessment plan

As you proceed through logical design, you will make any number of assumptions that might turn out to be untrue. If a bad assumption could cost you significant time or money, it is a risk. In addition, you are bound to want to suggest strategies that might not work. Instead of suggesting them anyway, or worse, not suggesting anything that you think might fail, suggest it, and assign it a risk probability. If the risk is too great, you can drop the idea. If, as is often the case, the risk is close to the reward, or can be mitigated with a little extra effort, it might be worth a try.

At any rate, before you finish logical design, revisit the risk assessment that you created as part of your readiness assessment (which is covered in Chapter 13, "Getting Ready for a CMS," in the section "A risk assessment").

Here is a starter list of risks from requirements and logical design to get you thinking:

- **System requirements:** Do requirements conflict? Were you unable to get them down to a comfortable level?
- **Audiences:** Was there agreement? Are there too many audiences? Too few? Will you be able to get the data on each?
- **Publications:** Are there too many? Are they too complex? Will you be able to make all the integrations required?
- **Content components:** Are there too many classes? Are there more classes to come that you have not yet categorized? Will you be able to gather the metadata you propose to include with each component?
- **Authors:** Do you trust the information you have gathered? Are there too many different types of authors? Are they skilled enough to do their jobs? Are they close enough to you to follow the rules you create?
- **Acquisition sources:** Are there sources out there you could not analyze? Will some take too much effort to convert? Are you dubious about the quality or consistency of some? Are there too many?
- **Access structures:** Can you muster the effort to create the ones you suggested? Are they adequate to make content accessible? Will people really be able to assign components to the right categories? Can publication navigation really be built from the access structures you plan to put in the CMS repository?
- **Workflow and staffing:** Will the system take more effort than you can muster? Are your task time estimates accurate enough to make predictions from them? Can you find skilled people or train them? Are there bottlenecks in your workflows?

A revised project plan

At the end of requirements gathering and logical design, you know everything you need to know about what the CMS needs to accomplish. Although this is not all you need to know to lay out a final project plan, it is enough for a next hard look at the plan. If you followed the process I laid out for getting ready for a CMS (in Chapter 13, "Getting Ready for a CMS," in the section "A preliminary project plan"), then you have already produced a preliminary project plan. You can choose to revise your project plan either after requirements and logical design, or just once after logical design.

After logical design, you should be in a position to revise that plan as follows, based on what you learned:

- **Staffing:** You should have a very good idea of the people and tasks you will need to start and run your CMS. This new information should help you infuse reality into your plan.
- **Quantities:** You now know exactly how much content you will have to process and how many publication pages per month you will have to produce. This ought to help you refine the schedules and budgets for these activities.
- **Level of development effort:** You should have a good idea, from both the functionality components you designed and the system requirements you gathered, of how much software development will be involved in your CMS project. You should be able to do some basic calculations at this point on the amount of time, staff, and budget you will need for programming and other technical services. Of course, it will not be until you create implementation specifications for all of the software development tasks that you will know precisely what it will take, but at this point you should have a good first approximation.
- **Level of infrastructure:** From both the system requirements you gathered and the logical design you did, you should now be able to estimate the kinds and amounts of hardware and software you will need. Of course, until you select a CMS product and complete your implementation specifications, you will not know for sure, but there ought to be enough information available at this point to make reasonable guesses.

At this stage of the project, you should be able to revise a project plan such that it is clearly doable within the time frames and budgets that you have. As likely as not, when you first revise your project plan, you will find that you now exceed even your most liberal time and budget estimates. That is a sign that you have done a complete analysis. If this happens, you can either ask for more time and money or you can cut back.

If you need to cut back, the appropriate thing to do is to start at the mandate. What can you cut from your mandate that, in turn, negates certain requirements, which, in turn, negate components and publications? If you start, as most will, cutting components and publications first, you will have done the following:

- Ruined the coherence and unity of the mandate-requirements-logical design relationship that you worked hard to create.
- Lost a great opportunity to argue for more resources based on what the organization does not get to have for less. It is easy for the organization to let you cut content and publications and still hold you to the original goals (which is, after all, what they really care about). It is difficult for them to let go of the goals they themselves crafted.

Inevitably, you will have to cut back some of what you designed. You can be upset about the cutbacks, or you can take it as a sign that you have been smart and have designed ahead of implementation. It is good to make your current design bigger than what you can accomplish in your next implementation. A bigger design helps you be sure that the implementation is meeting current needs and preparing you for the future.

Tip Don't look at cutbacks as a defeat. They very well might be a good thing for both the project and the organization. If you cut the parts of the system that were the biggest stretch, you will be buying more time to really understand them before you actually have to do them. History is on the side of content management. Eventually, you will need to do all that you have designed and more. You needn't be in a rush to do it all at once, especially if you are really not quite sure how you could do it even if you had all the time in the world.

An executive summary

Sponsors, and anyone outside the project team, are unlikely to want to wade through the mountains of information you generate from the logical design. For these folks, it is best to create an executive summary that is fewer than 10 pages long. Here is a list of the sections you might consider including:

- A reiteration of the mandate and major goals of the project
- A summary of the requirements you gathered
- An overview of authors and sources
- A shortened, annotated version of the component hierarchy
- The messaging and personalization strategy
- A few choice page mock-ups to show what the final publications will look like
- An overview of the major staff positions and workflows you worked out

Note If you don't know what some of the items in the preceding list are, read on. They are all covered in the chapters that follow.

In addition, you might consider including the updated project plan (or some shortened version of it) as an appendix to the report.

Taking Stock of Your Staffing Needs

Table 15-1 lists the sorts of people you might want to include in your requirements gathering and logical design process (see Chapter 11, "Staffing a CMS," for the full descriptions of these job titles).

Table 15-1
Staffing Needs

Task	*Leader*	*Participants*	*Notes*
Requirements plan of attack	Content Manager or Business Analyst	All other analysts as needed	Content manager will at least want to supervise this activity.
Requirements document	Business Analyst	All other analysts as needed	Content manager may want to participate too.
Design plan of attack	Analyst or Business Analyst	All other analysts as needed	Content manager will probably supervise this activity.
Executive summary	Content Manager or Business Analyst	All other analysts as needed	Content manager might be best suited to create and present this document.
Audience analysis	Business Analyst	Publications analyst, outside marketing staff, and all other analysts as needed	The business analyst is in the best position to lead this effort, but will need to rely heavily on the other people in the organization who specialize in marketing or in serving a particular audience.
Collection design document	Content Analyst and Software Analyst	There should be strong input from the version analyst and input from all other analysts as needed. Leader will bring in content contibutors and others from the organization as required.	If practical, the two analysts should collaborate, because it touches each of their disciplines separately. If this is impractical, then one should own the document and the other contribute a lot to it.
Management design document	Content Analyst and Deployment Analyst	All other analysts as needed	If practical, the two analysts should collaborate, because it touches each of their disciplines separately. If this is impractical, then one should own the document and the other contribute a lot to it.
Publication strategy documents	Publications Analyst	There should be strong input from the Content Analyst and other analysts as needed.	The Software Analyst might need to participate in the templating section.

Continued

Table 15-1 *(Continued)*

Task	*Leader*	*Participants*	*Notes*
Page design	Publications Analyst	Content Analyst, Publication Designer, page developer, and UI Specialist	The Publications Analyst should direct the effort, but the other staff members are the ones to create the deliverables. The Content Analyst should consult heavily on the parts of the analysis that concern the content component. The Publication Analyst or Publication Designer will need to bring others in for specific input.
Personalization strategy	Publication Analyst	Content Analyst and Software Analyst	The Publication Analyst should lead the effort, because the personalizations affect the structure of the publications. However, the Content Analyst will need to consult heavily on the access and retrieval of the content, and the Software Analyst might have to consult on the personalization rules and their implementation. Input from marketing groups also might be needed for the messaging analysis.
Publication Administration	Publication Analyst	Deployment Analyst and other analysts as needed	The Deployment Analyst can aid in understanding the role of infrastructure staff in the publishing workflows and tasks.
Revised project plan	Content Maanger	All other analysts as well as a Project Manager as needed	The analysts might be responsible for each part of the plan, but the Content Manager is in the best position to own the plan, make the tough compromises, and get continuing support for the plan.

Summary

Requirements and logical design are the beginning of the meat of a CMS project. My version of requirements has you ask the organization for the most general but the most important content, publication, and infrastructure needs.

My version of logical design has you work from these general requirements to a very specific design for what you want the system to accomplish:

- In doing requirements gathering, use common sense and organizational savvy.
- In doing logical design, be sure to iterate a number of times through the process and find a way to manage the tremendous amount of detail you will amass.
- Prepare a plan of attack for requirements and logical design before starting out on them.
- Prepare analysis documents for collection, management, and publishing (or along any other outline that covers all of the logical design bases).
- Give the organization's leaders an audience analysis and a localization plan that they can debate and, ultimately, accept.
- Give the organization's leaders an executive summary that they can easily read and thereby get a feel for what you are doing.
- Finally, revise your project plan and risk assessment at the end of logical design so that it reflects your most current thinking.

In the next chapter, I deal with one of the most contentious and stressful parts of a CMS project: hardware and software selection.

✦ ✦ ✦

CHAPTER 16

Selecting Hardware and Software

◆ ◆ ◆ ◆

In This Chapter

The dilemma that CMS product companies face in trying to meet customer demands

The build or buy decision and the selection process

The deliverables that you may create as part of system selection

The staff that you may want to involve in the selection process

The criteria that you may use to select a CMS product

◆ ◆ ◆ ◆

In years past, there were no CMS products on the market. In the future, you'll no more think of creating your own CMS than you'd think of creating your own ERP system. Today, it's likely that you can get most of what you need from one of the commercially available CMS products. Still, it's likely that you need to do a fair amount of custom programming to get the results that you desire.

In this chapter, I provide an overview of the build vs. buy decision, giving you some basis on which to decide whether you're better off building or buying your own system. Assuming that most organizations prefer to start from a commercial product, I spend the bulk of the chapter discussing the process for selecting the product most suitable to your needs.

The Product Paradox

Between the time that it became possible to deliver information electronically and the time that it became necessary to do so, an impressive array of content mills were invented to fill the gap for organizations that had no choice. In the late 1980s and early 1990s, I invented five or six authoring and publication systems that were tailored to a particular group of people with a particular content set. About the time that I started taking a more general approach to the problem of content management, so did others, and the content management industry was born.

In the mid 1990s, I did most of my work for Microsoft. At that point, the company's Web site (at `www.microsoft.com`) was already getting out of control. One of the main contributors, the Microsoft Office group, had thousands of pages that needed to be tracked and posted each month. I was given the assignment to create some sort of management system for this process. "What's to be on the pages?" I asked Howard, the project manager. "We don't know," came his response. "Okay," I said. "What kinds of pages are you creating?" "We're changing that all around, so we can't really tell you," replied Howard. "Hmm," said I. "What sort of metadata do you need to track for these pages?" Howard smiled. "Any kind," he told me. "We're going to want it to be open, so make it that way." This wasn't going to be an easy project. "Wow, okay, umm, I'll see what I can do," I said. "When can I get the final page designs?" Howard smiled even wider. "You can't," he chuckled. "We'll be redesigning right up until launch. Make it so that whatever design we come up with, the tool can handle it." I picked my jaw up off the floor and we started designing. Before too many hours, Howard and I had a plan that *ought* to work.

Working feverishly with Daniel, a very young but quite brilliant recent immigrant from the former Soviet Union, I conceived and birthed WebPub. WebPub was open. It made no assumptions about the kind of pages that the company would produce. We could set it up for any navigation structure, page types, and page design. We invented the following:

- **An administrative interface** for setting up the components and elements of the system (although we didn't call them that).
- **A contributor interface** that presented forms, where you could enter the content of a component (although we didn't use Web-based forms).
- **A templating system** that enabled you to type special syntax into an HTML file to add content and navigation to whatever design the page had (although the syntax was proprietary and limited).

WebPub was a success. It became one of the first Web-site content management systems used at Microsoft. The system helped the Office group cut the redesign cycle on their large site from three months to three weeks and was used for more than two years (a real accomplishment at Microsoft).

I spin this tale not to get you to buy WebPub, but to discuss the lessons that it taught me.

Because WebPub was such a success, other groups inside and beyond Microsoft also wanted to use it. We hastily produced version 2.0 with documentation and tutorials and called it a product. *Ha!* After a few months, we realized that WebPub was the idea for a product, not the product itself. To turn this idea into a reality, we'd need to remake our entire company. In addition to a great idea, a product company needs the following:

- **Continual development:** Before version 2.0 was even released, we saw how much more flexibility we really needed. Potential clients constantly asked for more than our immature application could deliver. Daniel and Bob needed to be replaced by a complete development group.

- **Support:** Few people trust a software product (and for good reason), and they all have a lot of questions. Daniel and Bob needed to be replaced by an entire support staff.
- **Sales and marketing:** As a service organization, we were accustomed to personal relationships with a small number of clients. Selling product is a numbers game. You need to hit as many people as possible and move a lot of units.
- **Finance:** A product organization spends a lot of money up front and reaps profit through high sales down the road. As a service organization, we were set up to be paid for every hour that we worked. There was no pot of money waiting to be spent up front.

Note

Other CMS product companies are on the road not taken by WebPub. They had a good idea that worked in a particular context and the vision to sell it to the world. They've tried, more or less successfully, to remake their companies into product development, marketing, and sales organizations. The good ones are those that have a strong and continually developing technology base as well as an organizational structure that supports the continual revolution that's needed to keep the product moving forward.

Doing and then not doing WebPub taught me a lot about the contradictions and paradoxes that motivate the content management industry. Being involved in CMS vendor selection processes since then has helped me see from the customer's side the following contradictions that CMS companies must live with:

- **Innovation vs. legacy:** In any fast-moving industry, product companies have a constant struggle between continuing to support their old versions and releasing new versions that transcend the old rules. This conflict, I believe, is what kept the document management industry from dominating the content management industry. The drag of its existing client base kept it from seizing the new opportunities of the Web. (See Chapter 9, "The Roots of Content Management," in the section "Huge information bases," for more information on this topic.)
- **Sales vs. technical staff:** To build a strong product, you need a great technical staff. To sell that product, you need people who know how to make and close deals. The people who want to buy a CMS usually prefer to talk to the technical staff. The people who sell a CMS want you to talk to the sales staff. The sales staff has more time to talk to you, is usually much better at talking, and knows how to get you to buy. The technical staff is usually too busy, not good at talking, and may even tell you not to buy the product. The position of sales engineer was created to deal with this dilemma. If they're good, sales engineers can talk as well to programmers as they can to business people and bridge that gap. If they're bad, they're too poor at interpersonal communications to make it in sales and not technical enough to answer even the least probing questions.

- **The "next version" issue:** The functionality that you really want is always going to be in the next version. The reason is simple: The released product necessarily lags behind the market need, which, in content management, grows and evolves really quickly. In your search for the right product, you naturally take for granted the basic functionality that most products have and focus most on the stuff that's particular to your initiative and infrastructure. So that's what you ask for. To survive, the product companies must be responsive to customer needs and so, if enough people ask for it, they must plan to include functionality to meet the new need in the next version. Of course, by the time that the next version ships, the new functionality is taken for granted and there's a new set of needs that everyone wants. You should expect the next version issue. In fact, if you don't hear "next version" in response to many of your questions, you should be suspicious that the company isn't keeping up with the market. On the other hand, you should require solid evidence that there's more than vapor behind the next version.
- **Vision vs. reality:** Every good company has a vision of the future (such as the paperless office, a computer on every desktop, and so on). On the other hand, just like the next version, the future is always brighter than the present. The present is marred by the sad facts of reality. (The supporting technologies aren't yet ready, bandwidth isn't high enough, and so on.) The company needs to sell you fully on the future, so you put up with the present. You need to listen to this vision of the future and make sure that you like it and then get back to the present and make sure that you can really live with all those sad facts.
- **Biggest vs. newest:** As all big software companies have learned, you can be market leader or you can be market innovator, but you can't be both. The bigger a share of the market that you own, the more legacy customers you must support and the more new customers you need to attract to hold your position. Market leaders need to play it safe and play to the masses to be market leaders. Innovators, on the other hand, are up-and-coming players: They seek to undercut the leaders with their speed of development, revolutions in technology, and their ability to remake themselves as the market changes. Neither position is bad or should necessarily reflect poorly on a particular company. But you should question any company that claims to be both, and you should recognize that, in an industry as immature as content management, long-term industry leadership is far from decided.
- **Me, too, and not me:** Especially in these early days, the content management industry is extremely competitive. Companies are constantly monitoring each other's messages and coming up with responses. If one has a good idea, for example, the others quickly latch onto it and say, "Me, too!" Conversely, if one product is seen to have a flaw, the others are quick to claim, "Not me! I do it right!" From the company's perspective, this attitude is necessary for survival. It's a hostile world out there. Customers judge you based on a single random fact that they pick up in an article or by hearsay. You must get past the first uninformed hurdles to get to the real value that your product can offer. If you don't have quick and decisive answers to the "me, too" and "not me" comments, you're sunk. From the customer's perspective, pat answers that always manage to be just what you may want to hear can be a real turnoff.

- **Alliances:** No CMS product can do the job by itself. There's too much involved for one product to cover. On the other hand, customers naturally expect that the CMS can handle any issue remotely related to their information. Some companies respond by continually widening their offerings to try to cover the entire process. Even these companies, however, can't cover everything. And even if they did cover everything, they'd be accused of being proprietary and not allowing you to integrate "best of breed" third-party products. It's a bind — companies lose if they're seen to cover too little or too much. The answer most product companies have found is alliances. They tell you, "Rather than do all this ourselves, we partner with X, who's a known leader in the field of Y." But what does it mean to partner? In the computer industry, it can mean anything from agreeing to put your partner's logo on your Web site to having an ownership stake in your partner's company. The alliance that the customer wants to see is one where the two companies have tested and documented integration methods and have agreed in writing to ensure that their future releases stay compatible. Such solid agreements, however, aren't easy to find. They bind the two companies to a lot of work and to each other for the highly competitive and unknown future.

- **Defining the terms of the discussion:** A successful product company is proactive. They craft a message that they think will resonate with an audience, and then they design a campaign to deliver that message in as many ways as possible to that audience. (Some even use their own systems to do it.) From the company's perspective, they want to lay out the issues and frame a solution by using their product. Then they can train sales staff to deliver the set message, and they can ensure that they can progresses from problem to solution, to purchase in an effective way. Finally, by using a set message, they can quickly qualify sales opportunities. If you're too far from the message, you aren't a viable prospect. A set message can be propagated widely and used to quickly find the portion of the target audience that's ready for the message. Customers, on the other hand, want to get comfortable with the messaging and then move into how this product meets their specific needs, not a general need. In a field as immature as content management, your needs are likely to vary from the general (or at least you're likely to think so). A set message works well for the product company but may be seen as shallow or too marketing-oriented by customers.

- **Simple vs. complete:** A product can either be simple to use or complete in its coverage. This is a basic fact of application development. The more stuff that the application can do, the more difficult it is to make it all simple to understand and use. Customers, however, want both. The answer that people want to hear is, "We can do that, and it's really easy." In reality, the best applications make the easy things easy and the difficult things difficult. Beginners mostly want to do the easy things and need the simple user interface. Advanced users eschew simple interfaces and want all the power of the application in front of them in one hugely complex screen, where they can see everything and do everything. The best products layer functionality so that the beginner's tasks are in front of the more advanced functionality. So, rather than expecting that all features be easy, you should ensure that the difficulty of a feature is proportional to the ability of the intended user.

- **Why you shouldn't just wait:** From the product company's perspective, of course, you should never wait. You should buy now before your problems get the better of you. Also, while you wait for the right price or functionality, new needs may arise and you may end up waiting forever, your needs always ahead of the released product. From the customer's perspective, waiting may or may not be good. If there's money that needs to be spent, waiting is bad. If there's indecision or lack of budget in the organization, waiting is good. Decide for yourself how long you ought to wait for the right product to come along.
- **Stock vs. custom functionality:** From a product company's perspective, the best approach is to build an application infrastructure that provides a robust set of basic services that can be mixed and matched to meet any of a range of needs (as I discuss in Chapter 4, "Functionality is Content, Too!", in the section "Monolithic vs. Mix-and-Match Functionality"). They can then build a user interface on top of the basic services that can be easily modified to meet a range of needs (with some extra programming, of course). From the customer's perspective, customization is a nasty word. Customers want to flip the switch and have everything that they want already there and working. If that can't happen, they at least want to know that there are people available (internally or on contract) who can do the customization and that it's not going to cost too much.
- **Single support vs. realistic support:** Customers want a single solution and a single place to go for support (or, as one client put it, "a single throat to strangle"). No matter what or where the problem is customers want a single number to call and an immediate response. This very reasonable requirement makes most product companies cringe. Just think what it takes to provide such support. First, the CMS is so intertwined with organizational infrastructure that to even figure out where a problem's originated, let alone fix it, is tantamount to supporting the organization's entire infrastructure. Second, it takes a lot of skill to diagnose and fix problems in a CMS. This talent isn't cheap to have sitting around waiting for problems to arise. Finally, with companies that stretch across the entire world, the support must be in multiple languages and across all time zones. Can you blame companies for cringing?
- **Planning vs. action:** The more planning that you do, the longer it takes you to buy and the harder are the questions that you must ask of a CMS vendor. In addition, the more planning that you do, the less likely you are to want to let the product company set the terms of the discussion. Given this, you'd think that product companies would work against planning. This is rarely the case. With the exception of some companies that try to minimize the entire content management problem, most are in complete agreement that you need to plan before doing a content management project. The reason is simple: If you don't plan, the project is likely to fail, and you're likely to blame the product company. (Some have even been sued by unhappy customers.) To resolve the dilemma between planning and purchase, most product companies try to get you to buy first and plan later. They have professional services to help you do much of what I outline in this book. Many do a good job. The only problem is

that the entire planning process starts with the assumption that the product in question is the right one for the task. There's no opportunity (as you'd expect) to define the problem and then choose the product.

- **What customers want vs. what they need:** Product sales staff see a lot of projects go by. If staff members are good, they can pick up a lot of good and practical understanding about what you, the customer, need to do. On the other hand, it's not in the interest of a sales engineer to tell you things that lead you away from her product. It's also not in her interest to slow down the process by showing you how your simple understanding needs to be revisited and deepened. The customer, of course, wants the sales staff to be consultants, directing her toward the right solution and offering practical advice where needed. It serves the sales staff to become trusted consultants and offer advice, as long as it doesn't hamper the sale. Be aware of this dilemma and thank a sales person who slows the sales process for you or offers advice that's adverse to her real goal of getting you to buy something.

It's difficult being a vendor. There are contradictions at every turn that keep honest and straightforward sales staffs up at night. There can be either an overly dependent or an overly adversarial relationship between vendors and potential clients, neither of which is good.

The long and the short of it is that customers expect too much, and companies give too little. I don't even bother to discuss companies or customers with ill intent. The best of intentions still isn't enough to resolve the conflicts inherent in the software development process, the sales process, and the content management industry. I've seen many shades of gray, but never a vendor or a customer I'd call dishonest. Mostly, I've seen vendors who want a product they believe in to succeed and customers who don't want to be taken advantage of and are afraid that they don't know all that they need to know to make the right decision.

I've seen a lot of blame thrown around — customers accusing vendors of lying, cheating, and stealing, and vendors accusing customers of trying to soak them for knowledge with no intention to buy. Both sides would do better simply to understand the constraints of the other and the contradictions of the system in which both sides are embedded.

As a customer, if you don't expect vendors to be objective or to act in your interest if interests diverge, you can still trust and believe what you hear. As a vendor, if you don't expect that customers are ready or able to make a decision or that they'll easily give trust, you can still gain that trust and make a sale. After all, most customers are willing to buy something and most products have something worth buying.. The process turns out best for both sides if the customer knows what to ask and the vendor knows how to quickly answer the questions that determine whether the product fits the need.

Deciding Whether to Build or Buy

It wasn't so long ago that there were no commercially available content management systems. In those days, the decision was easy: Do all management by hand, build your own system, or stay out of the game. With the exception of the old SGML systems doing documentation (as discussed in Chapter 9, "The Roots of Content Management," in the section "The Principles of Document Management") and some text-manipulation tools for programmers, you could buy nothing that advertised any sort of management functions at all. This was less a problem than you may imagine, looking at today's world. Few organizations had made any kind of move to electronic publishing, the Internet was a government experiment, and any companies putting information on CD-ROM were expected to be software development houses.

Today, everyone in this business is at or near a crossroads. I can't tell you categorically to build your own CMS, and I can't tell you that you should clearly buy a system. In some circumstances, you may not have an option, because nothing like what you need exists. More and more commonly, however, something does exist. Then the question becomes whether it's cheaper and faster to start from the existing system and augment it or start from scratch. It's still rare, but getting less so, to find a system that does all that you want with little or no additional customization.

The buy-or-build decision, then, is a matter of degree. If a commercial CMS is cheap enough and does enough of what you want it to, it's no longer worthwhile to build your own. What I *can* say categorically is that you shouldn't take on a CMS building project unless you have the following:

- **A strong leader:** You need someone with the vision to conceive and then direct the development of the effort. This person needs to have a firm foundation in content management as well as in the tools and platforms that you intend to use.
- **A strong development staff:** You're going to do some hard-core database and Web development. You need a seasoned and sizable team to handle this task. You want developers who not only know their languages and databases, but also have a feeling for and comfort in editorial and publishing systems.
- **A desire for an ongoing effort:** It's not enough to create a CMS; just as must a product company, you must continue to develop it to keep up with the pace of change in your organization and in the content management industry. You can perhaps compete with product companies today in delivering the features your organization requests, but can you still compete in the future?
- **A strong support organization:** You must be willing to support the CMS as well as build it. All the support headaches that I mentioned in the preceding section, "The Product Paradox," are yours, not some other company's.

Indeed, all the contradictions that I mentioned for a product company in the content management industry apply to you if you take on the task of building a CMS. The organization looks to you as the "vendor" and traps you in all the same snares in which product companies fall. In short, you should be prepared to create a little product company inside your organization to build and maintain a CMS.

Maybe at this point you're saying, "That's pretty conclusive. I wouldn't bite off that task!" If you are, wait while I balance the scales a little in the following points:

- **You still may have a lot of development to do**, even if you buy a CMS. You can outsource this development, but you still must manage it and ensure that it fits into the architecture of the CMS that you buy. In fact, the jobs that I discuss in Chapter 11, "Staffing a CMS," don't change whether you're building or buying a CMS. Of course, the number of positions and the job duration of the staff do change.
- **You still have a lot of support to do**, even if you buy a CMS. Many CMS companies simply can't provide the kind of support that a large enterprise needs. If you use one of these products, you're still faced with building a robust support infrastructure.
- **You may end up replacing most of the CMS** anyway. I've seen more than one project where the custom code ended up dwarfing the contribution of the CMS product. At first, it seemed that the CMS would take care of large chunks of functionality. As these projects wore on, the CMS just seemed to get in the way of what could just as easily have been programmed. (At least, it was presented that way by the programmers.)
- **You must pay a pretty penny**—a CMS is expensive. For the cost of many of the major packages, you can buy an awful lot of programmer and analyst time.

If you're thinking in general about building or buying a CMS, I hope that I've put you right on the fence (because that's where I am). On the other hand, I hope you see that, if you weigh the issues of your specific situation, the balance ought to tip in one direction or the other.

Finally, take heart in the fact that whatever decision that you make today undoubtedly needs to be reviewed in the next two years and very likely gets overturned by another decision (or at least undergoes a thorough reworking) within five years. This industry is moving too quickly for any decision to last for long.

If you're at all in doubt, it's worth your time to go through a product-selection process. You don't need to buy at the end of it if you decide to build, and you just may find that there's a product out there that gets you enough of the way toward your goal (even if only in the next version) to be worth the price.

In the remainder of this chapter, I assume that you're selecting from among the commercially available CMS products. If you're set on building your own CMS, you can use the information in Part V, "Building a CMS," to start thinking about how you may do it.

Techniques to Get the Job Done

The CMS vendor selection process isn't so different from any other vendor selection process that it needs its own separate methods. Thus most of what I suggest is common sense to people who've done a lot of product selection. In the sections that follow, I infuse this common sense approach with a content management focus to give you some advice and guidelines to follow.

How to select decision makers

I've seen more politics wrapped around the product decision than in any other part of a CMS project. More than once, I've seen a CMS project stall and die because of the following:

- The lack of consensus that you previously could choose to ignore comes out and fouls up the entire process.
- Some groups have their favorites and inside players, others have the tools that they're already using, and yet others have one tool that they've heard of and just want.
- Some people see this as the most significant decision of the project and one where they can have a say.
- Some people see this as a way to influence how large budgets are spent.
- Some people think that, if you make a bad call on the product, the project's doomed, so they put too much emphasis on the process and drag it down.
- Some people just think that they're good at system selection and plug in because they have experience.

All these reasons have some validity and some invalidity. My overall feeling is that selection gets all the attention because it's so tangible and easy to understand. Money is spent and a product is chosen. The other parts of the project are more difficult to get your hands around and not so straightforward.

For whatever reason, system selection can be a real hassle if you don't put serious effort into reining in all the people who want to decide, as follows:

- **Depend on the consensus that you built** during the mandate process. If the process didn't yield a list of who gets to decide, you at least should have gotten a group of people with enough authority to help you choose a small but evenly distributed group with no members who've decided before the process starts.
- **Ratify a group process before you begin.** Make sure that everyone agrees to it and agrees that the winner wins. Too often, I've seen a clear winner emerge in this process, only to be shot down by people who don't like it "for some reason." If they can articulate a reason, it can be added to the list and scored with the rest of the criteria that you use. If they can't articulate why they like

or don't like a product, their opinion shouldn't count for much. Don't create process on the spot as you need it. This leads to a lack of respect for the process and rigging by people who can figure out how to manipulate the criteria and scoring.

✦ **Decide on a fair and impartial scoring mechanism** for each scoring step that you include in your process. Agree to the range for each scale and agree to what constitutes each value in the range. Then each person can make a judgment in the strict context of the scale and scoring criteria that you established.

Tip

The more numbers that you have in a scoring scale, the more room you have to differentiate between products—and the more work that you must do to define what each number means. I prefer scales that have three values for subjective measures (bad, okay, and good) and no more than ten for more objective measures. (Zero is no functionality, and ten is far more than you'd ever need.) Make your scales as small as possible while still retaining enough room to differentiate products.

✦ **Agree to stick to what your process decides.** It helps no one to add up all the scores and then ignore them because they don't come out right.

Note

If you find yourself stymied by the continual subversion of your objective process, you may try giving up. Maybe an executive decision just needs to be made about the product to use. Maybe your offer to give up gives new incentive to those who need to respect the process (or, at least, gives incentive to their supervisors). In the end, it's more important that some product be used than that it be the absolute best choice.

As far as who should be in the reviewers group is concerned, I can think of two types of people: those who represent a significant organizational interest and those who have the expertise or perspective to judge vendors. Obviously, you should shoot for people who have both qualities. I'd resist people who represent a constituency but have no basis to judge between vendors. As much as someone may want them to serve as a reviewer, they can slow down the process and cause a general lack of respect for the rest of the reviewers from the vendor and the organization.

You find more suggestions for how to staff a product selection process later in this chapter, in the section "Sorting Through the CMS Selection Criteria."

How to select a product

I've seen all manner of processes used to decide on the best system to buy. In many cases, you can't call them processes at all: You call around, see what other people are using, and go with that. In most cases, however, there's a process, and it centers around a list of requirements, vendor presentations, and some sort of selection committee. The process that I present is comprehensive and exhaustive. If you're planning to spend a lot of money on your CMS, it's worth the extra effort. On the other hand, you should balance the full treatment that I give you with the timing and depth that you think you need. In the full process that I describe, there can be a lot of questions to ask and answers to process. The process runs as follows:

- Create a high-level overview of the CMS project that you can pass around.
- Canvass the market for the products that seem to fit.
- Make the first cut from the list of candidates, selecting those that seem worth really pursuing.
- Send a Request for Proposal (RFP) to those who make the first cut.
- Select a small number of finalists by scoring the RFP responses and any follow-up questions that you ask.
- Have technical drill-down meetings and check references from each of the finalists.
- Have a presentation from the remaining candidates.
- Make a final decision by combining the scores for the references and presentations with those for the RFP.

The following sections give more detail to the process.

A high-level overview

To start the selection process, it's useful to create a short project overview that you can include in early correspondence with the vendors. The overview also should orient your selection committee to the major points toward which you're aiming.

To create the overview, take the high points from your project mandate, requirements, and logical analysis. Here's a sample overview:

> Our company is seeking to implement an enterprise-level content management system that aids in the creation, management, and publishing of personalized content to the Web and beyond. At the highest level, the system is successful if it does the following:
>
> - Provides a solid framework behind the staff and processes involved in creating and tagging content.
> - Supports a wide and diverse contributor base with tools and processes appropriate to skill level and commitment.
> - Provides a full-featured repository where content components and file-based resources can be stored, tracked, updated, targeted, combined, and archived.

- Provides a flexible, template-driven publishing capability that can format and output any combination of content and file resources to standard databases and file formats (including XML, HTML, flat text for e-mail messages, and QuarkXPress for print output).
- Provides strong, easy-to-use workflow tools that can guide the entire creation, storage, and publishing process.
- Supports a robust publishing, testing, and deployment environment.
- Provides a full-featured but extendable personalization capability for targeting and custom publishing of content.
- Integrates with our existing internal systems and current Web infrastructure.

The product or products selected are successful if they do the following:

- Have "out-of-the-box" functionality capable of covering a wide range of the preceding requirements with minimal customization.
- Have standards-based customization and extension capabilities that enable full integration with existing systems and rapid development of additional functionality."

Canvassing the Market

You can begin your search for the appropriate software by performing a broad overview of the CMS market, looking for products that address your general needs. Colleagues and the Web are good places to start.

Here are a few tips for conducting your search:

✦ **Get recommendations.** Try sending your overview via e-mail to anyone you can think of who may have an opinion. The recommendations give you a place to start and a set of products to which you can compare the ones that you find yourself. Interestingly, this technique may also turn up biases and inside favorites from the people in your organization whose opinion you ask.

✦ **Keep your analysis close at hand.** Nearby, you should have your requirements, logical design, and the selection criteria that you think are most important; look at them often as you do your search. This helps you avoid the disorientation that you're bound to feel as you follow endless links and get lost in the rhetoric and marketing spin of the Web sites that you visit.

✦ **Develop a short set of questions that you ask at each Web site that you visit.** This helps you stay on task and give an even, standard treatment to each product. As you may suspect, I like these three: What collection processes do they cover, what management functions do they cover, and what publishing functions do they include? Use these or create your own, but don't let the site control how you find information or what information you look for.

- **Consider core products vs. peripherals.** I'd roughly estimate that, in mid-2001, 40 or 50 products do enough really to be considered content management systems. (Early last year, there were half that many.) On the other hand, there are hundreds that advertise some connection to content management. Develop a quick eye for the products that don't have enough core content management functionality to qualify for your search.
- **Keep a side list of your special needs.** This list should go beyond standard content management concerns (special marketing needs, unusual publications, particular integrations, and so on). You may come across products that don't address all your needs but do hit one or two. You may find these products helpful later on, either as add-ons to the system that you purchase or as good examples of how a particular need can be met.
- **Document your search.** If you stay on a site for more than a minute or two, there's something of interest there. Paste its URL into a file so that you can track your way back to it later as you say, "I know I saw that somewhere . . ."
- **Don't spend too much time on this process.** You can easily get lost for days and days. Continue keeping up with the subject on the Web, but don't hold up the rest of the process for more than a day while you choose your initial list of candidates. If in doubt, include a company on your list. You can easily remove them later.
- **Start a file.** In fact, start both a physical and a computer file on each interesting company. You eventually accumulate a lot of material on the ones that make it to the end of the process.

The preceding points address Web sites of CMS product companies. My informal estimate is that far more than 90 percent of your Web search results come from such companies. The rest are from service organizations that do content management projects and from articles, publications, and conference presentations on the subject. Although I've not been overly impressed with the depth or breadth of coverage in the pages I've seen in this category, they're certainly worth perusing for general information and, especially, for comrades who may have faced problems similar to your own. I've always had good luck contacting and getting help (and even friendship) from people I've heard of on the Web.

No need to be stingy in this process: Include any product that looks seriously interesting. If you end up with a list of more than 10 or as many as 20 products, you're on the right track.

Making the first cut

With a little effort, you can winnow down the large number of product companies that have something to do with content management to the handful that seem to address your particular needs.

If you have enough lead time, you can send an e-mail message to or call each company on the list and request a complete marketing package. In addition, go back to each site and collect as much relevant information as you can about each candidate.

Tip If you request marketing packets, make sure that you ask them to include all white papers, case studies, demo programs, sample documentation, industry analyses, and pricing sheets that they have. These tell you much more than the glossy sheets that are standard fare for marketing folders.

Don't be afraid to summarily dismiss some candidates as you collect information and find out that they really don't fit. If you have time, you can also search the Web for each product name. This finds you reviews and other commentary on the product (as well as any flames or accolades that the product's received publicly).

Tip Note the names of people who've commented on a product. You may need to talk to them later about their experiences.

The core of the first cut is a preliminary set of evaluation criteria (usually a spreadsheet). Start by synthesizing your requirements and logical analysis with the questions listed in this chapter, as well as any others you came up with, to create a high-level cut on your selection criteria. Go back over your notes and links from your market survey and narrow the list to no more than 10 to 20 criteria to apply to each of your initial candidates. Use simple criteria and simple scoring rules (short scoring ranges). Have all your reviewers apply these criteria and, as quickly as possible, score each candidate product, using all the resources that you provide as well as any others that they may uncover and share with the group.

In this process, I prefer a small set of quick passes through the list to find products that easily drop out. People shouldn't expect that they need to do complete and exhaustive analyses. They don't have time for a complete analysis and may become very frustrated. Rather, ease them into the process by doing a number of quick passes. With luck, the first pass or two drops out half or more of your initial list of products. This may be all that you need to make the first cut.

The first cut should yield between five and 10 candidates. With five, you risk having too few to make it to the end of the process; with 10, you risk having to spend too much time collecting information. You decide what's appropriate for your organization and schedule.

The sales presentation

After the first cut, it's time to get serious. Your group should have enough practice reviewing and scoring to be comfortable with going deeper, and you should have a group of good candidates.

At this point, some groups choose to have a preliminary demo from each of the candidates. This is a nice idea for becoming familiar with the products and the organizations behind them.

If you choose to have demos, I recommend the following:

- **Don't require preparation of your team or the vendor.** Ask for the standard presentation. This serves a number of purposes. It gets the process going quickly, the company can send its usual talent, and your people can either attend or not, depending on their schedules. It gets the standard presentation over and done with so that, at the next meeting, you can get down to details quickly. It gives the vendor's representatives the opportunity to present their best face and walk you through the stuff that they're most proud of. In the next meeting, you talk much more about the parts that the vendor's not so proud of.
- **Aim to understand the product, not question it.** Save your probing questions for later and become as immersed in the product as you can. Be positive, to draw the presenter out and to have her give you the fullest demo possible. Each product has lots of interesting and good features that you should try to understand and appreciate.
- **Freely discuss your requirements and needs.** The vendor's presenter is trying to qualify you as a prospect as much as you're trying to qualify her company. Try to answer her questions as openly as possible so that she can get a solid feeling for what you've accomplished (which, by now, ought to be quite a lot) and what you want to accomplish in the end.
- **Find out the names of the people inside the product organization** that you can contact directly for specific information. If you're lucky or persistent, you can build a good list of expertise for future reference whenever you're having trouble getting an answer.

The RFP

Your main job here is to create the complete selection criteria list and form it into an official Request for Proposal (RFP). You may call it an RFP, or you may prefer to be less formal, but the idea is the same. You create a complete scoring system (most likely a spreadsheet that extends the simple one that you use to make your first cut) that lists all the questions that you need answered to come to a list of finalists, along with a place to score each finalist.

Note Vendor sales representatives may like to do more of a consultative sell, sitting down and talking at length with you about your issues before trying to propose how the solution would work. You can use the first demonstration meeting to accomplish this or schedule separate meetings. But make sure that it doesn't end up slanting your process toward the company that sells the hardest at this early stage.

Usually, my preference is to ask a lot and expect a lot of my questions to go unanswered. Others choose to ask only the most essential questions and insist on complete responses. I like the former approach because it provides a lot of discussion and input to the next stage of the project. Besides, even if you ask only essential questions, you're not guaranteed to get complete responses without a lot of probing and pushing.

If you choose to have demos, I recommend the following:

- **Don't require preparation of your team or the vendor.** Ask for the standard presentation. This serves a number of purposes. It gets the process going quickly, the company can send its usual talent, and your people can either attend or not, depending on their schedules. It gets the standard presentation over and done with so that, at the next meeting, you can get down to details quickly. It gives the vendor's representatives the opportunity to present their best face and walk you through the stuff that they're most proud of. In the next meeting, you talk much more about the parts that the vendor's not so proud of.
- **Aim to understand the product, not question it.** Save your probing questions for later and become as immersed in the product as you can. Be positive, to draw the presenter out and to have her give you the fullest demo possible. Each product has lots of interesting and good features that you should try to understand and appreciate.
- **Freely discuss your requirements and needs.** The vendor's presenter is trying to qualify you as a prospect as much as you're trying to qualify her company. Try to answer her questions as openly as possible so that she can get a solid feeling for what you've accomplished (which, by now, ought to be quite a lot) and what you want to accomplish in the end.
- **Find out the names of the people inside the product organization** that you can contact directly for specific information. If you're lucky or persistent, you can build a good list of expertise for future reference whenever you're having trouble getting an answer.

The RFP

Your main job here is to create the complete selection criteria list and form it into an official Request for Proposal (RFP). You may call it an RFP, or you may prefer to be less formal, but the idea is the same. You create a complete scoring system (most likely a spreadsheet that extends the simple one that you use to make your first cut) that lists all the questions that you need answered to come to a list of finalists, along with a place to score each finalist.

Note

Vendor sales representatives may like to do more of a consultative sell, sitting down and talking at length with you about your issues before trying to propose how the solution would work. You can use the first demonstration meeting to accomplish this or schedule separate meetings. But make sure that it doesn't end up slanting your process toward the company that sells the hardest at this early stage.

Usually, my preference is to ask a lot and expect a lot of my questions to go unanswered. Others choose to ask only the most essential questions and insist on complete responses. I like the former approach because it provides a lot of discussion and input to the next stage of the project. Besides, even if you ask only essential questions, you're not guaranteed to get complete responses without a lot of probing and pushing.

Tip

Just because you get a bad answer doesn't mean that the product really can't do it. Assess the credibility of the answerer and, if you want, give the company a break by digging deeper until you find an answer that you can trust. Similarly, answers of "yes" or "we do that," count for little unless you dig deeper.

If you choose to send a complete set of questions, warn the vendors that you're giving them a big job and make sure that you give them ample time to respond. Of course, no matter how much time you give them, most wait until the last minute and then panic if they can't complete the document.

Cross-Reference

The section "Sorting Through the CMS Selection Criteria," later in this chapter, has a large number of questions that you can include in your RFP.

Selecting finalists

I've learned not to expect too much from RFP responses. They often fall far short of what I'd prefer to see. I believe that the problem usually is that I expect too much detail and the respondents have too little time and training to respond adequately. You may think that, after a while, companies would build up a pool of stock responses to respond in full to almost any question that you may have. So far, this has not been my experience. Still, this is a very young field that's growing so quickly that it's understandable that vendors can't spare the time or talent to respond at the first request.

That's why I believe that it's a good idea to plan to have follow-up meetings with the candidates that make the RFP cut. Before you schedule these meetings, see whether any candidates drop out due to lack of response or unacceptably poor performance on the RFP. For those that make a reasonable effort and have answers that are on the target, if not on the bull's-eye, more discussion is warranted. By this time, you've undoubtedly been noticed by the companies, and if you're a big enough fish, you have their eager attention as well.

Again, in these meetings, I favor multiple quick passes through the questions over a single, in-depth pass. On the first pass, focus on the issues that the candidates did particularly well or poorly on in their RFP response. If they did poorly, was it because of a lack of responsiveness to the question or because they can't do what you want? Make a list of all the weak points. Do these points amount to a disqualification from the process, or with some additional time, may they be resolved favorably? If more time is warranted, divide your list into topics. (The criteria categories that I propose in the section "Sorting Through the CMS Selection Criteria," later in this chapter, may help here.) Rather than asking the same person to answer the same questions again, see whether you can find the right person inside the product company for each type of question. Providing your questions in writing and letting your sales contact know that continued participation hangs on deeper responses from a qualified expert may help bring in the right people.

You can use e-mail, phone calls, one-on-one meetings, or group discussions to move each question to a final score. You may find it helpful to divide your team by expertise or interest to ensure the most efficient coverage of the questions.

Tip Don't divide your team by product if you know that there are favorites. Instead, divide them by question type so that each person sees all products.

At some point after your first pass through the RFPs and before your follow-up is complete, you should officially score the responses. A spreadsheet with the tallying built-in is helpful. Make sure that each reviewer is fully aware of and in agreement with the process. An answer of "I don't know" or "I need more discussion" ought to be allowed. The reviewer herself can decide whether her reservations are enough to postpone the final scores. I've generally had several people score each product based on those team members' ability to judge the products and vendors. Then I've used the spreadsheet to average the scores to come to the final score for each criterion.

You want to prioritize each question that you score by giving it a weight. A question with a weight of 10, for example, is counted ten times as heavily as a question with a weight of 1. Make sure that you establish these weights *before* you begin scoring.

Note How much of your process to reveal to the vendors is always an issue. I'm of two minds. On the one hand, I think that the more they know about the criteria, how they're scored, and what you're questioning, the better they can respond. And most use the information that you provide to try to serve you better. On the other hand, the more that they know, the more some of them may try to interfere and bias the process toward themselves. You decide.

Your schedule and allotted effort should decide how many finalists you can choose. Two is low, and five is high. For the finalists, the process requires much more effort on both sides.

I emphasize again the points that I made earlier about who gets to decide. If you want a clean, decisive process, decide beforehand what your process is and stick to it. If you continually change your criteria or decision process, you frustrate your team and the vendors and you end up with no result or a result that's questioned and subverted.

Technical drilldowns

With a small pool of finalists, the time is right to get down to the small details. Do a more thorough job of analyzing the RFP and follow-up questions from the finalists, and schedule one or more meetings where your technical experts and theirs gather to envision the relationship. Experts can be of a variety of stripes, as follows:

You can use e-mail, phone calls, one-on-one meetings, or group discussions to move each question to a final score. You may find it helpful to divide your team by expertise or interest to ensure the most efficient coverage of the questions.

Tip Don't divide your team by product if you know that there are favorites. Instead, divide them by question type so that each person sees all products.

At some point after your first pass through the RFPs and before your follow-up is complete, you should officially score the responses. A spreadsheet with the tallying built-in is helpful. Make sure that each reviewer is fully aware of and in agreement with the process. An answer of "I don't know" or "I need more discussion" ought to be allowed. The reviewer herself can decide whether her reservations are enough to postpone the final scores. I've generally had several people score each product based on those team members' ability to judge the products and vendors. Then I've used the spreadsheet to average the scores to come to the final score for each criterion.

You want to prioritize each question that you score by giving it a weight. A question with a weight of 10, for example, is counted ten times as heavily as a question with a weight of 1. Make sure that you establish these weights *before* you begin scoring.

Note How much of your process to reveal to the vendors is always an issue. I'm of two minds. On the one hand, I think that the more they know about the criteria, how they're scored, and what you're questioning, the better they can respond. And most use the information that you provide to try to serve you better. On the other hand, the more that they know, the more some of them may try to interfere and bias the process toward themselves. You decide.

Your schedule and allotted effort should decide how many finalists you can choose. Two is low, and five is high. For the finalists, the process requires much more effort on both sides.

I emphasize again the points that I made earlier about who gets to decide. If you want a clean, decisive process, decide beforehand what your process is and stick to it. If you continually change your criteria or decision process, you frustrate your team and the vendors and you end up with no result or a result that's questioned and subverted.

Technical drilldowns

With a small pool of finalists, the time is right to get down to the small details. Do a more thorough job of analyzing the RFP and follow-up questions from the finalists, and schedule one or more meetings where your technical experts and theirs gather to envision the relationship. Experts can be of a variety of stripes, as follows:

- **Business experts** can gather to discuss terms of the contract that you may sign.
- **Editorial experts and information architects** can come together to discuss the collection system and metadata modeling facilities.
- **Programmers** can meet to understand how the system may be developed.
- **IT folks** can envision the architecture and deployment of the system.
- **Publications staff** can deliberate on the system's capability to produce the appropriate output.

As part of this set of meetings, you may want to schedule additional demos of the product or other background sessions to get your team fully up to speed on the capabilities of the products.

These discussions should result in a clear idea in the minds of your team of how you'd complete your tasks with the different systems. At this point, you should feel comfortable sharing your full requirements and logical design with the vendors to give them as much as possible to go on and to test the strength of their understanding and desire to work with your team.

The vendors should understand that they need to move beyond the sales staff to bring in people from their development groups or from their professional services groups who can really contribute. It help to bring these people in if you're clear, during these meetings, that you're testing the relationship that you hope to have with the company that you finally choose to work with.

I favor having each meeting result in a diagram that each side believes represents what the system may look like or do, given your needs and their product. Although the content of the diagrams change based on whom you're meeting with, the concept should hold across all the meetings that you hold. If you do this, you thank yourself later. You can hang all the pictures next to each other to help you choose from among a possibly confusing array of competing approaches.

Envisioning the project

In the technical drill-down meetings, you've worked through most of the issues that were troubling. If you have, you're now ready to figure out just how the project would progress with each potential product.

You can start by adding detail to the RFP responses that you receive and rescoring them based on the information that you receive through follow-up meetings. If you can get past the bad feeling that they could have told you in the first place all the stuff that you found out later, you can capture and organize a lot of the detail that you discover and still have a single place to grade the company (in your scoring matrix). Make sure that you mark the revisions that you make to the RFPs so that you can tell what you added and what they originally said.

Compare the first scores that you came up with to the revised ones, as follows:

- The second set of scores are more reflective of what the product and the vendor are really capable of doing.
- If the second score is higher than the first, you had a bad sales rep.
- If the second score is lower than the first, the difference is the amount of spin added to the original RFP response.

Your biggest task is to create modified project plans for each prospective product. Work through your existing project plan with representatives from the product company. Work through the complete plan, but pay particular attention to the parts of the project that the product can't do out of the box, as follows:

- **For the major tasks that require custom development**, develop an extension to the project plan that has enough information in it to be used as a basis for cost and schedule estimates. Try to get as good an approximation as you can of the amount of customization that you need to do to augment the stock functionality of each product. You can perhaps get the vendor to separately estimate doing the custom development.
- **For the major integrations**, try to get as specific an explanation as possible of how the integration would occur. Get whatever time and cost estimates that you can and try to plan the integration in the same way that you planned the major customizations. You can also possibly get an estimate from the vendor for integration services.

You can update your risk assessment, once per potential product, noting any new, heightened, or reduced risks that using that particular product entails.

Note If you think that this is a lot of work, you're right. The only consolation is that one of the project plans that you create now is the one that you can actually go forward with later.

The vendors can be very helpful to you in this endeavor. They have a lot of great experience doing CMS projects and can bring a great level of detail to your plan. If they're not helpful or know less than you do about your plan, that's good information to know as well.

References

Rightly or wrongly (rightly, I believe), most companies don't give you references until they're sure that you're a highly qualified prospect. The reason that they wait is that they have precious few clients willing to put out the effort that a reference requires, and they want to reserve them for the highest value. There are some companies that don't have many references because they're not well liked, but that's not something that they're likely to tell you. With any luck, you weed out these vendors in your first cuts.

Compare the first scores that you came up with to the revised ones, as follows:

- The second set of scores are more reflective of what the product and the vendor are really capable of doing.
- If the second score is higher than the first, you had a bad sales rep.
- If the second score is lower than the first, the difference is the amount of spin added to the original RFP response.

Your biggest task is to create modified project plans for each prospective product. Work through your existing project plan with representatives from the product company. Work through the complete plan, but pay particular attention to the parts of the project that the product can't do out of the box, as follows:

- **For the major tasks that require custom development**, develop an extension to the project plan that has enough information in it to be used as a basis for cost and schedule estimates. Try to get as good an approximation as you can of the amount of customization that you need to do to augment the stock functionality of each product. You can perhaps get the vendor to separately estimate doing the custom development.
- **For the major integrations**, try to get as specific an explanation as possible of how the integration would occur. Get whatever time and cost estimates that you can and try to plan the integration in the same way that you planned the major customizations. You can also possibly get an estimate from the vendor for integration services.

You can update your risk assessment, once per potential product, noting any new, heightened, or reduced risks that using that particular product entails.

Note If you think that this is a lot of work, you're right. The only consolation is that one of the project plans that you create now is the one that you can actually go forward with later.

The vendors can be very helpful to you in this endeavor. They have a lot of great experience doing CMS projects and can bring a great level of detail to your plan. If they're not helpful or know less than you do about your plan, that's good information to know as well.

References

Rightly or wrongly (rightly, I believe), most companies don't give you references until they're sure that you're a highly qualified prospect. The reason that they wait is that they have precious few clients willing to put out the effort that a reference requires, and they want to reserve them for the highest value. There are some companies that don't have many references because they're not well liked, but that's not something that they're likely to tell you. With any luck, you weed out these vendors in your first cuts.

If you've followed anything like the process that I've laid out, you have more than enough proof at this point that you're serious. References are just about the last step that you take on the path to your decision.

What you want from references is people like you who are using the product to do the things that you want to do. You need to be very lucky to get such a perfect match. More likely, you find people who speak kindly of the product (same as you'd do if you agreed to be a reference) and are in a different industry or have only some of the same issues that you face.

One mistake that people make with references is to leave the choice entirely up to the vendor. What can you do to come up with two or three people who are similar to you and are using the products that you're reviewing? Following are some things that you can do to reach that goal:

- **Check the Web.** In your initial search, did you find any people who had opinions on the product? If so, ask them to whom you can speak. If you didn't fully search the Web for the products that are finalists, do it now.
- **Check the logos on the vendor's site.** Product companies proudly display the names of their major clients. Who in these organizations can you talk to?
- **Check conferences and journals.** Although content management is still a bit obscure, most technology magazines have done at least one story on it. In addition, there are content management conferences where people do case studies on the use of one product or another. Maybe you can find just the person that you need to talk to in one of these venues.

The point is that you need comparable installations to help you decide which product is best for you. Whether you get those from the company or by yourself is irrelevant. Besides, if you tell the companies that you're doing your own search, they may be a bit more helpful in the references that they provide.

After you have the references, you need to schedule a conversation. If you can go to them, you get a lot more time and effort from them. Plus you can sometimes see the system running and talk to all sorts of people. If you can't manage to make it to the reference's site, a phone call may still be enough.

The biggest issue is how much time and effort you can get from the references. You should need no more than an hour or two if the person is sufficiently briefed on her own system. Prioritize your questions to make sure that, if you're suddenly cut off by a higher priority issue, you've at least gotten the main points out.

To me, the best approach is to recognize that these references are your peers. Even if you choose a different product, here's a person who's in the same boat as you. What do you have to offer her? At the very least, you should have a little sympathy for her problems. More likely, you have some ideas or information that she can use as well. If you can establish a relationship rather than being a single inconvenience that the reference must endure, you get more and better information. More important, you may get a friend that you can collaborate with over the course of your career in content management.

What should you ask the reference? The same questions that you've been asking yourself and the vendor. Use the reference in conjunction with the technical drill-downs to solidify your feeling about how it would be to use this product. Of course, general questions such as the following are always a good start:

- Did the vendor deliver what she said that she would?
- Did the product do what you expected it to do?
- Have you gotten the support that you needed?
- Was the training any good?
- How close are your requirements to ours?
- What problems did you have with the installation?
- With what are you integrating?

Some of the most telling answers come from asking questions such as the following:

- What was the most surprising or least anticipated part of the implementation?
- If you had to suggest one improvement to the product, what would it be?

These may be good icebreakers, but make sure that you get quickly to the key issues to make the best use of the limited time that you probably have.

Vendor presentations

By now, you're very close to a decision; in fact, you may have a clear leader who all but has the contract. But there's no contract yet; only a lot of discussion. You also likely have a list of issues that never have been successfully resolved for each vendor. Finally, you may have people with the authority to sign the purchase order who'd like a chance to meet these people before giving them a lot of money.

You can combine all these needs into a final vendor presentation that has the following purposes:

- **Final resolution of any outstanding issues.** What you should aim for is that there are no outstanding issues that would be embarrassing to talk about at such a weighty meeting.
- **A full discussion of the terms of the agreement** that you may sign with the company. Prior to the meeting, you should get sample contracts and pricing and service schedules that you can review and comment on.

- **An executive review of the company.** This gives your sponsors (or whoever else wants to attend) the chance to meet with whatever executives the product company can send. Having the big guns there on both sides can have an amazing effect on the issues that seemed unsolvable before. In addition, you have the right people present to discuss any strategic aspects of the relationship that you may form. Can the two companies help each other in more than just content management?
- **A cost estimate.** You should work to have a final price tag for the product available at the meeting. Alternatively, you can set this meeting as the point at which the price is set. You may have a lot of hoops to jump through to arrive at a price. In my experience, CMS companies aren't forthcoming with cost estimates. That's another good reason to set this meeting as a deadline. You also may want to have all the customization and integration estimates that you prepared ready as well at this meeting to verify them with the company's official representatives.

This is decidedly not a meeting to demo the product or have the sales people present their slide shows. It's a down-to-earth, do-we-want-to-work-together meeting. The time has passed for features. It's time to make a final determination of the company itself.

Tip

Make sure that you brief your sponsors on the need to avoid wasting meeting time on demos. Find out whether they'd like a demo, schedule a separate time before this meeting, or invite your sponsors to one of the earlier meetings where they can ask functionality questions.

Centering the discussion on the contract has the effect of putting a seriousness and reality on the discussion that no other context can yield. There's nothing like saying, "So can we insert a sentence here that binds you to the conditions that you just agreed to?" to make the point hit home and finally be decided.

The result of your final vendor meetings should be that your sponsor group (and the people who sign checks, if they're not the same) should form an opinion about the companies that you're reviewing. If their opinions match yours, you're finished. If the two don't match, you have a bit more work to do.

Making the final call

If you've followed the process to this point, you have enough information to make a selection, as follows:

- You have large files on each product.
- You have RFP responses that are augmented and scored.
- You have diagrams showing each possible approach.
- You have prospective project plans for each product.

- You have risk assessments for each product.
- You have references from peers using each product.
- You have cost estimates.
- You have agreement in principle on a contract for each product.

You have the information that you need, but you also need the organizational will and process to decide and move on. Your selection process should have produced a numerical winner in the RFP scores, as well as project costs. In addition, you have one or more subjective assessments from your team and sponsors on each of the other deliverables from the process.

If you're really lucky, all these align and point toward a single winner. Otherwise, you have the following options (in order of my preference):

- **Drive for consensus** by hashing through the disagreements in open debate until all opinions converge enough for a decision. You can expect a lot of debate but a solidly supported decision.
- **Create an objective scoring** method for the subjective factors. You can create a scale for each factor and a weight for each subjective factor relative to the weight of the RFP score and the product cost. Then create a calculation that sums the scores for each reviewer and product and calls a winner.
- **Escalate the decision** to someone who can make and enforce it. Provide the decision maker with all the information that you've collected and let her decide.

Clearly, the process that you choose has more to do with the nature of your team than the information at hand. A well-coordinated team with mutual respect and a strong unified vision of the project can easily reach consensus. A team that feels bound by the process more than by respect for each other is likely to accept an objective scoring. Even for a well-functioning team, the objective scoring method may be fastest and easiest. If the team isn't at all well functioning, the only good course may be to punt the decision to someone higher in the organization. That at least ensures that the decision is made and everyone abides by it.

Of course, the worst time to decide how to decide is at the end, when emotions may be high. At the beginning of the selection process, you should assess your team's inclinations and decide how to decide.

Taking Stock of the Deliverables

I've spent ample time on the selection deliverables in the course of describing the process, so I just provide a few additional comments in the following sections.

Product files

It's a good idea to have the files that you created for products in a central place where the whole team can access them. I also favor having them jointly owned by all team members, who can add to them as they find good material. Don't forget to attach a URL or other identifier to all materials so that you can track back to their sources as needed. You undoubtedly need a directory as well as a physical file for each product. I've had luck with a big binder, where all materials were three-hole punched and the directory location for computer files was written on the cover.

You can include the notes that you compiled from content management Web sites in a separate file. If you're way more organized than any group I've ever worked with, you can continue to add to the links file and periodically publish it as a resource for people in your organization to learn about content management.

The selection criteria and RFP

I'd suggest starting with a spreadsheet that catalogs all your criteria and their weightings. You can continue to develop the spreadsheet until it's time to send out the RFPs. At that point, you can send a spreadsheet to each vendor, but it's more friendly to convert the spreadsheet into an electronic document (Microsoft Word is an accepted standard) and send it that way. The conversion takes a little extra effort, but it makes the RFP easier to work with (at least for the CMS product vendors). In addition, you can add a cover page and company background that's difficult to do well in a spreadsheet.

I've found it particularly useful not only to provide a long list of questions in an RFP, but also to include a short description of what you're trying to accomplish at the start of each section. You can draw from your logical design to create these introductions. By doing this, you not only provide needed context to the person answering the questions, but you also give yourself a way to make sure that you've covered all the most important questions that your logical design may have raised. As you put in the introductions, ask yourself, “Do the questions that I ask here cover the implementation of what I just said that we want to do?”

You may also consider including parts of your project mandate and the high-level overview that you wrote at the beginning of the selection process (which I present in the section “A high-level overview,” earlier in this chapter). Finally, consider augmenting your lists of fill-in questions with scenarios. Suppose that you have some authors who seem particularly problematic. (Maybe they use strange tools or are remote). In addition to (or, in some cases, instead of) asking your authoring questions, you can explain the situation with your authors and ask the vendor to propose a solution. You may get mixed results, as scenarios require more thought and effort by the respondent than short answers, but it may be worth a try to see who has the most creative solutions.

The score card

You can build your RFP scorecard directly into the spreadsheet that you devise for selection criteria. If each reviewer gets a copy of the spreadsheet file, each has her scores immediately tallied for her. To combine the scores of all reviewers, you can create a master spreadsheet that looks inside each reviewer's copy and tallies the full results, or you can just copy and paste the individual scores into a new sheet.

Tip

Make sure that you have a way to link questions in your RFP to your scoring matrix. I prefer to number the sections in the RFP so that I can refer to them exactly in the scoring matrix.

If you decide to score subjective factors, you can simply add them to the RFP spreadsheet file and sum them with the rest of the RFP questions.

The design diagrams

In design, a picture is worth more than 1K words. By making a diagram deliverable from each technical drill-down meeting, you accomplish the following:

- Ensure that the meetings focus on design rather than become unfocused discussions of what's possible.
- Ensure that some solution is reached. Meeting notes can meander and never conclude. A picture can be more or less precise, but it always tells a complete story.
- Can compare one product easily to the others. Each diagram is a possible solution to the same problem.
- Have good input to the implementation process that comes next.

Convinced? If so, here are some guidelines to help you make the most out of these diagrams:

- **Title them.** Preferably, the titles should be the names of the systems that they describe (such as "PageMaker Publication System" or "Repository Structure"). You also may include a short description of the problem that each diagram addresses.
- **Label all major shapes and lines.** It's amazing how quickly people forget what all those parts are and do.
- **Include inventories in the margins.** Lists such as staff, CPUs needed, and connections all help you later develop plans from the diagrams.
- **Use a standard iconography.** If everyone agrees on the picture for a database, LAN connection, Web server, and the rest of your symbols, your diagrams are easily read by anyone on the team and more easily compared to each other.

The decision report

At the end of the selection process, you've reached a major milestone in the project. You're finished with the last big decision. From now on, the decisions are all about *how*, not *what*. To mark the milestone and also to inform the organization, you should prepare a short decision report.

Include the following in the report:

- The high-level overview that you wrote at the beginning of the process.
- An overview of the process that you followed.
- Major factors that led to the decision.
- An overview of the product selected.
- Information about the company selected.

This report should be distributed at least to all the people who participated in the project so far, and, ideally, to anyone who may participate in the CMS in the future.

If there's enough interest, you may want to schedule an open session with representatives of the product selected so that anyone can come and see a demo and ask questions.

Tip

After all the hard work that you do on product selection, don't forget to reward yourself and your team. This decision is a clear cause for celebration. Don't forget the people at the product company. If you used this process, they worked as hard as you did and deserve a party, too!

Risk assessment update

At the beginning of system selection, your risk analysis has mostly high-level organizational issues and stuff about how you possibly can't do what everyone wants to do. As a result of the selection process, you can add a set of more concrete risks and mitigations to your assessment. Table 16-1 provides an example of a simple product risk assessment to get you started. (Effect and probability are on a scale of 1 to 5, where 1 = low and 5 = high.)

Table 16-1
Product Risk Assessment Example

Area/Risk	*Effect*	*Probability*	*Score*
Design/Planning			
Technical training and implementation, support not timely	3	1	3
Development			
Major development effort required	3	3	9
Inadequate developer documentation	2	3	6
Integration			
Major technical issues related to integration	3	2	6
Integration results in poor site performance	3	2	6
Personalization inappropriate or inadequate	1	3	3
Rollout			
Usability/training issues	2	2	4
Ongoing Support			
High level of end-user support required	2	2	4
Inadequate end-user documentation	1	3	3
Incapable or unresponsive help-desk staff	2	2	4
Scalability issues, significant hardware requirements	2	4	8
Futures			
Vendor's technology becomes obsolete	2	2	4
Vendor becomes financially unstable	2	3	6
Vendor experiences high level of change/turnover	2	4	8
Vendor's technology direction becomes misaligned with ours	2	2	4
Vendor's market direction becomes misaligned with ours	2	0	0
		Overall Score	78

Project plan update

If you did your homework in the selection process, you need little extra work at the end to update your project plan. Make sure that you go through it again, however, and fill in all the spaces that you left blank because you ran out of time.

If you've added people to your team from the product company, make sure that you add these new staff members to your plan. They probably have their own project plan, so you must work together to integrate the two. In any case, you should maintain a master plan where the all staff and schedules come together to complete the rest of the project.

Taking Stock of Your Staffing Needs

The CMS product selection process begins with the build-or-buy decision. This is a big decision with lots of political and technical overtones. I believe that the decision should be presided over by as high a person as you have on the core team. In my taxonomy, that's the content manager. The content manager most likely at least wants to supervise this activity and pass the recommendation by the sponsors before making a final decision. The content manager or her equivalent probably also should be the one to write and circulate the high-level overview.

I favor having a single person head up the vendor-selection process and others participate as reviewers or outside observers. My preference is also to have the person running the process refrain from being a reviewer as well. There's just too much conflict of interest between coming up with the evaluations and doing them. Also, the leader may have her hands full managing the opinions of the reviewers and may not need the extra hassle of offering opinions as well. Finally, as main contact to the vendors, it's useful for the leader to be able to say, "It's not my decision!" Whether she participates in the selection or not, the leader should take responsibility for the following:

- ✦ **Doing or supervising the market canvass** that begins vendor selection. It may be easier for one person to do this, but it may be faster and more politic for a group to do it.
- ✦ **Overseeing the first cut.** Although the reviewer group should make the cut, it may need a lot of help keeping to the process and closing on a decision.
- ✦ **Creating the final set of selection criteria.** With ample input from the reviewers, the leader can still be responsible for forming the final selection matrix.
- ✦ **Creating the RFP.** The mechanics and styling of the RFP document should come from a single source.
- ✦ **Overseeing the process of creating scoring metrics.** Again, help and prodding may be needed to get a final agreement.

- **Overseeing the selection of finalists.** Here, the impartiality of the leader really comes in handy.
- **Deciding which technical drill-downs need to happen** and receiving the design diagrams that result. Diagrams result only from constant but amicable pressure. Of course, input to the decision should come from all quarters.
- **Overseeing the vendor presentations.** You need an officiator of these events who understands both the vendors and the organization well enough to negotiate between them.
- **Overseeing the final call.** You hope here is that facilitating this process doesn't go beyond process and prodding.
- **Producing the decision report.** The selection leader is in the best position to create this report, but the content manager, or her equivalent, ought to be the one who circulates it and gets feedback from it.

Here are some of the other roles that you may need to fill in the selection process:

- **Reviewer group.** Earlier in this chapter, I discussed staffing the reviewers group, so I'm not repeating that here. I add only that you should aim for between four and 10 reviewers for optimum team performance.
- **Scheduling** is a big part of selection. I suggest enlisting administrative help to make sure that all the vendors and reviewers are where they're supposed to be and know what's required of them at all points in the process.
- **Technical drilldowns** require an appropriate analyst to lead them and participation from other analysts and subject-matter experts (implementers) who can drill as deeply as needed on a topic and help construct the diagrams.

Cross-Reference

See Chapter 11, "Staffing a CMS," for the full descriptions of the CMS job titles.

Sorting Through the CMS Selection Criteria

There's a vast number of questions that you may want to ask a vendor. In the sections that follow, I outline a number of them. As you read through these questions, I'm sure that you can think of some of your own and also, for better understanding, ways to reword the ones that I supply. Recognize that not all these questions are relevant to your needs, so choose them accordingly. Make sure, however, that you take into consideration future needs and ask the questions that may not apply today but that could conceivably apply a year from now.

These questions are the same ones that you must answer later to implement your system. In this chapter, I call these questions *selection criteria*. After you've selected a CMS product, however, they serve as physical design constraints. You can use them to specify how your system is to be created from hardware, CMS functionality, third-party software, and customizations that you develop.

The selection criteria in the sections that follow ask about how certain functions are accomplished in a particular CMS. Use the logical design that you created previously to augment these questions. In addition to the following questions, use your logical design to explain what you're trying to accomplish and to ask vendors questions about what you want to do and why.

For each of your criteria, it's also important to decide what the ideal responses to each of your questions would be. Doing this makes it much easier for your reviewers to score by comparing the vendor's response to the ideal response. I include some suggested "right" answers to the criteria that I propose.

Business criteria

In addition to meeting the technical and process requirements you may have for your CMS, you want to make sure that the CMS product company that you decide to work with meets your business criteria for a successful relationship. After you have its product installed and part of your e-business initiatives, you're quite tied to the fortunes of this company. You don't want to find out then that it wasn't the company that you understood it to be.

Here are some of the business criteria by which you may want to judge CMS product companies:

- **History:** You can ask vendors for a short description of the history of the company. Understand where they came from. Did they morph from a different industry (document management, for example)? Did they "ride a WebPub" from concept to product, bootstrapping from a service to a product company? Did they raise a lot of capital and burst onto the scene as a startup? It's less important which path they took than that you understand their history and how it may affect their future.
- **Business profile:** You can ask for general business data, including the number of years they've been in business and have been producing the proposed product. What's the ownership of their companies? What are their capitalization strategies? What have their earnings been in the past and what kind of growth have they seen? You may be surprised at how small even some of the major players are. If you're dealing with public companies, you can certainly check their growth and analysts' guesses of their viability in the future.
- **Customer profiles:** One of the best questions you can ask of vendors is how well they know their audience. If they can send you the profiles of the accounts that they see as key to their success, you can decide whether you're one of them. If, on the other hand, you're nothing like their target customer, are you looking at the right product? Ask for the number of completed and ongoing installations they've done that are similar to yours. Which use the current version and which use earlier releases?

- **Next versions:** Given that next versions are going to be needed (a near certainty), you should ask vendors about their upgrade policy, including how new releases, upcoming features, hot fixes, betas, and patches are communicated to customers and released. Ask how often to expect updates or upgrades and how much they cost.
- **Binding agreements:** For the set of issues that you may come up with that aren't resolved in the current version but are promised in future versions, what sorts of binding agreements or penalties is the vendor willing to take on? These binding agreements are one way to ensure that the vendor is serious about what's in the next version.
- **References:** Ask for references at the sites of the projects that are like yours. If possible, the references should include a business as well as a technical contact.

It's a lot to ask of vendors that they supply this much information (especially on top of the technical requirements, which are even more voluminous). As you negotiate the delivery (or lack of delivery) of these items, notice as much what the process is like as what's delivered. Is the vendor accommodating? Is the information available, or does she need to make it up? Do any vendors stall on items that they don't give you, hoping that you stop asking, or are they up front about why they can't deliver? Finally, do they just say what you want to hear, or do you detect a measure of sincerity and genuine desire to work with you? The impression that you get right up front is likely to carry through the rest of the relationship.

Overall criteria

The selection criteria in this section apply to the CMS as a whole. You find selection criteria sorted by collection, management, and publishing later in this chapter.

Project process and staff

The goal of asking the questions listed in Table 16-2 is to assess not only the process that the company recommends, but also how well the company knows its own product implementation cycle. If it can provide you with this information in an organized and well-considered form, you can assume that it has the project process down.

Cross-Reference

You can compare the process that the product company uses to the one that I overview in Chapter 13, "Getting Ready for a CMS," in the section "Understanding the CMS Project Process," to see whether it's (or I've) missed anything.

Table 16-2
Project- and Staff-related Questions

Question	*Notes*
Describe the general steps involved in an implementation of your product.	Does it seem overly simplified and does it diverge widely from the steps that I outline in Chapter 13, "Getting Ready for a CMS"?
What factors are necessary to ensure a successful deployment?	Do the vendor offer any particular wisdom that shows that they've seen and surmounted a lot of problems?
What's the typical time frame for a deployment and what can be done to minimize the time and effort required?	Does the vendor's timeframe include the full customization that probably needs to be progrmmed in or just general installation and some basic template setup?
Provide an example of project plans and deliverables from projects that are close in scope to ours.	The response tells you whether the vendor's completed projects like yours, how organized the the project process is, and how strong the deliverables are.
Describe the various project roles (both your own staff and your customers') and the numbers of people involved in each role. Describe what roles people had before the implementation and how their roles changed after implementation.	How complete and realistic are these lists? Do miss any categories?
What professional services does your company provide?	Many CMS product companies can provide full service, from design to maintenance. Each, however, has its own particular way of doing so.
For a project such as ours, what sort of professional services do you recommend?	You likely get a description of the standard services packages that the company offers. Try to get past this to the particular project composition that it sees for your project.
Can you provide a list of your service staff positions and their billing rates?	You can compare this list to the staffing list that I provide (in Chapter 11, "Staffing a CMS") and your own staffing plans to decide what you may need.

Development environment

The questions in Table 16-3 should give you a fair appraisal of the kinds of development tools that are allowed and preferred in working with their product. Keep in mind that some of these criteria are fairly technical in nature, so if you're not too familiar with software development, make sure that you have a technical resource review this section.

Table 16-3
Development-related Questions

Question	*Notes*
What development languages and tools were used to create your product?	It's good if the company used the more common Web-ready tools, such as Java and C++.
Please provide an overview of the ways programmers can customize or extend your product, including the languages and tools needed.	Here you're looking for a clear description as well as the tools that you want to use. Nonproprietary development languages and tools are preferred if you don't have ones that you particularly want to use.
What APIs are provided as standard with your product?	An extensive, well-documented API is preferred. Best is that the company builds its own user interface from the same API so that you can make sure that it's well tested and provides complete coverage of the product's functions.
What skills and tools are needed to create and modify basic templates for authoring, submitting, and publishing content?	Basic templates shouldn't require programming skills. Ideally, the tools are those that a person who can use a graphical HTML editor can easily learn and use.
What skills and tools are needed to extend and integrate templates beyond basic content inclusion and layout?	Nonproprietary development languages and tools are preferred if you don't have ones that you particularly want to use. You may prefer layout tools that are targeted at the kind of skills that a graphic designer may have. If a programmer is required to create page layouts, you may have trouble.
What languages and tools does your system support for automating administrative functions?	You want to hear that all the administrative functions of the product, including workflow, user administration, database management, and reporting, can be called independently by you, using Web technologies that you want to use (ASP, JSP, CORBA, COM, J2EE, and the like).
What languages and tools does your system support for automating content authoring and acquisition?	At the very least, you should be able to modify collection Web-based forms and embed them in your own Web applications. It's better if the product offers out-of-the-box integrations with the authoring tools that you like. (Microsoft Word is usually the most important, but integration with HTML editors and graphic design applications is becoming increasingly common.) You should be able to extend and modify these tools, using the programming languages that you like.

Localization

Localization is a complex issue, often barely recognized by CMS products. If you have localization needs, first understand exactly what they are. Don't expect a product company to tell you how to localize. Given your plan, however, it should be able to tell you how its tools can help. Table 16-4 lists a series of localization-related questions.

Cross-Reference

There's more on localization in Chapter 21, "Cataloging Audiences," in the section "Audiences and Localization."

Table 16-4
Localization-related Questions

Question	*Notes*
Please provide an overview of how your product supports publication localization. What specific tools and processes do you provide?	The overview ought to tell you how well the company understands localization and how deep its support for it is.
How is translated content produced and synchronized with the source content?	Ideally, the system maintains linkages between translated components to manage updates. It may be capable of handling content in different languages, including support for double-byte character sets or Unicode, international date formats, and global currencies. The screens that the content maintainers use must be easily translatable into different languages.
Are there any facilities in your product for multiple selection images or other regions.	You may need various versions of graphics and other media that are localized for your various media?
Please give a full example, with references and URLs, of a customer that has successfully used your system to create a global localization and translation solution.	A successful implementation by a customer who agrees to be a reference is a strong point in the company's favor.
Do you have any translation aids in your product, or do you integrate with any?	Translation aids help you put the master and translated versions of content side by side for comparison and difference checking.

Continued

Table 16-4 *(continued)*

Question	***Notes***
Can you version and track changes at the element level?	It's really helpful to be able to send some component elements for localization separately from others that don't need to be localized.
Is there template functionality for sniffing the origin nation and redirecting?	The CMS may be capable of reading HTTP headers and helping you decide the region of origin of Web visitors.
Do you provide multilingual UI for content contributors and producers?	Your international contributors may need, and are certain to appreciate, anything that you can do to localize their CMS interface.

Training

The questions in Table 16-5 assess how complete, convenient, and affordable training on this product is.

Table 16-5
Training-Related Questions

Question	***Notes***
Please provide an overview of the training that you provide.	You're looking to see a well thought-out response that shows that the company has put effort and resources into training and that it's professionally delivered.
What's the availability and cost of training, and how frequently is it offered?	You want to make sure that the training schedule doesn't affect your implementation timing.
Is training held at your site or ours?	Some may offer on-site training for large enough groups. Otherwise, travel costs need to be included in budget considerations.
What CMS staff positions do you train for?	There's almost always end-user and administrator training, but is there training for developers, managers, information architects, IT staff, or designers?

Table 16-4 *(continued)*

Question	***Notes***
Can you version and track changes at the element level?	It's really helpful to be able to send some component elements for localization separately from others that don't need to be localized.
Is there template functionality for sniffing the origin nation and redirecting?	The CMS may be capable of reading HTTP headers and helping you decide the region of origin of Web visitors.
Do you provide multilingual UI for content contributors and producers?	Your international contributors may need, and are certain to appreciate, anything that you can do to localize their CMS interface.

Training

The questions in Table 16-5 assess how complete, convenient, and affordable training on this product is.

Table 16-5
Training-Related Questions

Question	***Notes***
Please provide an overview of the training that you provide.	You're looking to see a well thought-out response that shows that the company has put effort and resources into training and that it's professionally delivered.
What's the availability and cost of training, and how frequently is it offered?	You want to make sure that the training schedule doesn't affect your implementation timing.
Is training held at your site or ours?	Some may offer on-site training for large enough groups. Otherwise, travel costs need to be included in budget considerations.
What CMS staff positions do you train for?	There's almost always end-user and administrator training, but is there training for developers, managers, information architects, IT staff, or designers?

Technical support

Bad technical support can make you very sorry that you chose a product. The questions listed in Table 16-6 should help you assess what sort of support to expect. Remember to be reasonable in your expectations of the support that you receive. It's not reasonable to expect the company to provide one-hour turnaround on all requests, anywhere, anytime. Recognize also that the better the support that you require, the more expensive it is. The correct response from the vendor to unreasonable support expectations is, "We can provide whatever level of support that you're willing to pay for."

Table 16-6
Technical Support-Related Questions

Question	*Notes*
Please provide an overview of the technical support that you provide.	As with training, you want to see a well thought-out response that shows that the company's put effort and resources into support and that it's professionally delivered.
Do you provide support, or do you work with any partners?	If the company's small, it's better if it's partnered with a larger support organization that can bring its support up to enterprise levels. On the other hand, you want to see that the company can provide the specific expertise to the partner to augment the partner's ability to address problems.
What's the geographic coverage of support, and at what hours is it available?	Obviously, the company should cover the same territory as your organization does.
In what languages is support available?	This may be an important consideration for you if you're embarking on a global deployment.
What's the guaranteed response time for a support request?	How willing is the company to guarantee anything having to do with support?
Is there a single point of contact for support, or do people need to know whom to call for what?	A single point of contact with fast triage and routing to the appropriate contact is preferred.
What kinds of self-help facilities do you provide?	FAQs, known bugs and workarounds, and tutorials for beginners can save a lot of money for the support organization.

Continued

Table 16-6 *(continued)*

Question	***Notes***
What guaranteed expertise is available on the first call, and what are the escalation procedures? How high in the technical organization can an issue be escalated?	The optimum cost-effective solution is to have enough skill on the first call to correctly route the problem. You want to see that, from the first call, there's problem tracking and a logical and reasonable set of criteria for escalating to each new level of expertise.
Describe your support licensing plans and associated costs. Do you offer a pay-as-you-go plan as well as contracts?	You may prefer a cost-per-use model that you can charge back to contributing groups rather than a fixed fee.
What comes as part of the purchase price and what are the costs associated with ongoing support after the initial purchase and installation?	The extent of services for setup and the cost of ongoing support can vary. You need to take into consideration the "total cost of ownership," not just the initial purchase price.
Are there any provisions for downtime or loss of revenue from systems impaired by a CMS bug?	Depending on how mission critical your system is, this sort of seriousness may be necessary. A willingness to enter into this sort of agreement shows that the company has extreme confidence in its system (or isn't smart enough to know that these sorts of agreements put it at grave financial risk).
How wide is the support? What software and hardware is covered?	How are problems that may be outside the CMS software supported? Are extensions using the standard API supported? Are template bugs supported? Are platform, operating system, Web application server, and database and network connectivity issues supported? For issues that aren't supported, how is it determined where the problem lies and how does the staff then tie into the appropriate other support organization?

Deployment

The questions in Table 16-7 help you understand how the product can be rolled out in your environment.

Table 16-7
Deployment-Related Questions

Question	*Notes*
What operating systems does your product run on?	The right answer is the operating system that you use!
What operating systems are required on the client machines for the database client and authoring tools? Describe any client-side software needed to use the system.	All basic functionality should be available through a Web interface so that the client needs only a Web browser. Client software should be available, but not required, as a supplement to a browser interface to enable a richer and more responsive UI.
Which Web browsers are supported for authoring, management, and administrative functions?	You hope that it's the ones that you use.
Describe the skills and size of the team needed to successfully deploy and maintain your product.	An open-ended question that you can use to assess the company's process and savvy on this issue.
What models do you employ for distributing the management and publishing environments across business units that aren't in the same facility?	If you need for people to contribute from around the world and for them to publish locally, you want to assess how the CMS can help you keep this distributed environment organized and minimize redundancy.
Is your system best deployed inside or outside a firewall? Describe the difference in requirements or performance if your system is deployed inside or outside a firewall.	This question helps get at the distinctions of using this system for an intranet vs. an Internet site. It also uncovers any problems that you may experience with remote contributors accessing the CMS.
Describe the process that you'd use to deploy your system across an organization with a structure such as ours.	Another open-ended question to see how well the company can understand and provide for your specific environment. You want to add a list of the significant factors that are present in your organization.

Collection criteria

The criteria in this section can help you assess the collection technologies and processes that come with the product or can be integrated with the product.

Authoring criteria

CMS products vary widely in their support for authoring. The questions in Table 16-8 should give you a way to compare each of the products that you review.

Table 16-8
Authoring-Related Questions

Question	***Notes***
Do you ship with any specific authoring tools? If so, which one(s)?	Only a few CMS products ship with authoring tools (word processors, image-creation tools, HTML editors, and so on). Although it's helpful to know whether the product includes these tools, it's probably more important to find out whether the product supports the tools in use by your organization.
With what third-party authoring tools do you directly integrate?	Direct integration means that some aspect of the CMS appears within the authoring tool. Some CMS products create a menu within a particular authoring tool, for example, so that you can save content to and retrieve content from the CMS repository.
What file formats can your CMS repository store?	Many CMS products can manage files that are authored in any tool. Of course, they generally don't manage the content *within* these files. In other words, the CMS serves as a document management system for files that it can't interpret.
What file formats can your product recognize and convert to XML or database records?	In this case, you're interested in the product's capability to open foreign file formats and parse them. The result is generally either XML that can be stored in an XML repository or relational database records with the full content of the source file in them. Some products, for example, take the text that you write in Microsoft Word and convert it automatically into XML for storage in the product's repository.
Please describe how a novice content contributor could author the textual content for a Web page and include graphics or other types of media to go along with the text.	You want the system to be simple enough for your beginners and casual authors to create simple pages with little or no help.

Question	***Notes***
Please give an overview of your Web-based form system for content input.	Almost all products now have a Web interface for content entry. Some may have a non-Web system as well that has fuller functionality than the Web-based forms.
Do you enable rich HTML or XML editing within the fields on your Web-based forms?	Some products include a little editing environment right in a Web-based form so that authors can format information as they type it.
Can Web-based forms show different sets of elements based on the user's permissions?	In other words, can you personalize input forms so that different people see different parts of the same content component, based on what you want them to enter?
What support do your forms offer for linking and uploading images and other media?	If this is a common need within your organization, you want to ensure that it's easy to add media and associate it with textual content.
Can your forms be automatically created from a DTD or database schema?	Some products autogenerate Web-based forms from the structures that you've previously defined, saving you a lot of work.
Can your Web-based forms be extended or modified by additional HTML or other programming?	This is an important factor, because it's likely that you want to customize forms and need additional features. Ideally, the forms can be customized by using a nonproprietary programming language.
Can your forms represent hierarchical relationships?	All products enable you to put a set of fields on a Web-based form. A few also enable you to nest some fields within others. This is really handy for entering the titles and text in sections and subsections.
Can authors create content while they're not connected and then submit it later?	In other words, can you use the CMS authoring tools "offline," a handy feature for authors who don't work on-site or for contributors who travel frequently?
Can authors preview content as it may appear in publications, before they finally submit it?	A preview function is extremely helpful and streamlines content creation and quality assurance workflows considerably.
Can authors view a revision and workflow history for their work?	An audit trail of workflow for each component is helpful for accountability as well as content troubleshooting.
Can authors perform a spelling check on their work?	A spelling check feature isn't yet mainstream among CMS products, so keep asking for it and eventually it may become mainstream.

Continued

Table 16-8 *(continued)*

Question	*Notes*
Describe the training expectations required for new content contributors.	The system should require minimal training for novice or casual contributors. You can expect some training requirements for more advanced authors.

Acquisition criteria

The questions in Table 16-9 can help you decide how well a product meets your needs to gather and process existing information.

Table 16-9
Acquisition-Related Questions

Question	*Notes*
With what acquisition tools does your product ship?	Some products include advanced tools for scanning content sources and automatically capturing content. Some have syndication tools that enable you automatically to accept incoming syndication feeds.
With what third-party acquisition tools do you integrate?	Most of the larger companies have strategic relationships with other companies (often, content syndicators) to augment their abilities to acquire content automatically.
How would we migrate our existing Web sites and other publications into your system?	Is there an automated process to import an existing Web site, or does it require an in-depth content analysis and restructuring into predefined content structures? Are there any tools at all for migrating publication types other than Web-based ones?
Can your product help manage the scheduling, usage rights, attribution, and payment for acquired content?	There's probably not much beyond the standard workflow system that the company can offer, but the question may get its people thinking.
Can your product automatically retrieve records from a database available locally and turn them into content components in your repository? Can these downloads be scheduled to happen periodically, with no human intervention?	The company may depend on standard programming to accomplish this, but it should demonstrate that its people understand the issue and have worked through it. You may have a product catalog database, for example, and want to create new components after new products are added to the database.

Conversion criteria

It may be that a product comes with few advanced conversion tools. The CMS product company, however, should be able to answer all the questions in Table 16-10 if the company is aware of and well versed on conversion issues.

Table 16-10
Conversion-Related Questions

Question	*Notes*
Does your product ship with any conversion tools? If so, please describe their capabilities.	Products really vary on this count. Some have no converters at all. Others have a wide range of built-in tools for text and media conversion.
With what third-party conversion tools do you integrate?	The answer to this question ought to give you a feeling for how well the company understands conversion. Does it have any partners there? Does it have an idea of how conversion fits into the process?
How does your product support on-the-fly conversion of text and media on a Web site?	As pages are requested, can the product retrieve content and convert it to the appropriate format? If so, you can store your media in one format and have it converted only if it's requested in another format. This isn't typical functionality, but some products can do it.

Aggregation criteria

When I first proposed a list of metadata management tools like the following one, very few product companies even recognized that it was an issue. Today, they're beginning to see the point of managing the metadata system. Over time, if you don't manage it, the system becomes unwieldy, with conflicting lists and values, and stops working as a way to unite all your content. Although the questions in Table 16-11 are still difficult for many product companies to answer even today, I suspect that, over time, most of or all these metadata capture and management features will become standard equipment in enterprise CMS products.

Table 16-11
Aggregation-Related Questions

Question	*Notes*
Does your product include or integrate with any metadata management tools?	Some products include auto-discovery and taxonomy creation tools that help you design metadata lists and automatically find metadata in the files that you process.
Can you automatically apply metadata values to content based on its type or location?	Some products enable you to choose a directory or other branch and batch-apply a metadata value to it. In addition, some can supply default values for metadata fields, based on the type of content being tagged.
Does your product have any tools for helping the administrator combine or split metadata fields that currently exist in the system?	As you move forward, you inevitably need to combine two similar metadata fields into one (for example, you may decide to merge an author field and a contributor field into one field) or split one field into two. (A single status field may need to be split into a publishing status field and an archive status field, to give you more control.)
Can you define mandatory and optional metatags?	This may be done in the Web-based form interface, but it may also be possible to do it globally for all uses of a particular metadata field.
Can you pull metadata lists from other sources, such as a database or a flat file?	You want to see, for example, if the product enables you to draw your list of products automatically out of a database and present them in a Web-based form.
Can you write metadata lists to other external files and databases?	You want to see whether the product enables you, for example, to add to the list of products and then periodically export the updated list to the database in which it originated.
Can you automatically bring up metadata lists or values for periodic expiration or review?	As part of managing your metadata system, you need to periodically review the validity of lists or values and either update or delete them. If you do, it's nice to have a tool that helps you update all the content that's affected by the change that you're making.

Question	***Notes***
Does your product tie into any accepted standards for metadata?	The Dublin Core Metadata Initiative and a host of other industry-specific metadata systems are being created and revised. At some point, you want to use one or more of them. Can the CMS help you, or does it stand in your way?
Does your product support all types of metadata types in the repository and metadata fields in your collection forms?	See the section "Metadata fields," in Chapter 20, "Working with Metadata," for more information on types of metadata.

Management criteria

The questions in this section can help you understand how the product stores and administers content and how it creates the infrastructure of a CMS.

Storage

The questions in Table 16-12 address the basic repository structure of the CMS.

Table 16-12

Storage-Related Questions

Question	***Notes***
How do you store content? In a relational database? An object database? A file system? A hybrid of the preceding types?	You want to know how the company structures its content storage to know whether it's compatible with your infrastructure requirements and can handle the sorts of content that you intend to manage.
Describe how your system stores the following content relationships: multiple hierarchies, multiple indexes, cross references between components and between publication pages, and sequences.	These are all the access structures that you need to store and access. All decent CMS products enable you to build a hierarchy. See how well the products that you survey do at building more than one hierarchy for the same set of components. You may need this facility if you intend to build multiple publications, each with its own TOC.
Does your content repository support extended character sets (double-byte) as well as international date and currency formats?	This is a must if you intend to localize in Asian or other double-byte languages.

Continued

Table 16-12 *(continued)*

Question	Notes
If your repository system supports XML DTDs or schemas, can you create one DTD for the entire repository? Can you create multiple DTDs that are enforced for part of the overall structure of the repository?	There's more information on the issue of multiple schemas in Chapter 32, "Building Management Systems," in the section "Storing the content model."
Does your product have the capability to check the validity of links and media references that appear within text fields in the repository?	In many systems, the hyperlinks and media references (such as an `<IMG>` tag in HTML) that are embedded in the middle of a block of text go unnoticed and can be broken without any clue. Some systems enable you to track these links and references even inside paragraphs of text.
What indexing and searching does your repository support? Does searching respect security?	A good answer includes rich indexing and querying, metadata (keyword) searching, full-text searching, proximity searching, and relevance ranking of search results, as well as the capability to exclude results that a user doesn't have rights to see.

Administration

Any CMS that you buy should do most of what I point out in Table 16-13. The differences are in how well it performs these tasks and how convenient it is.

Table 16-13
Administration-Related Questions

Question	Notes
In what ways can your system enable nontechnical business users to manage their own content?	This is an open-ended question designed to give the company a chance to address ease-of-use issues and how it caters to novice or casual users.
Please describe the utilities provided for system administration and operations personnel to effectively manage and support the system. Please identify any administrative functions, such as configuring workflows or publishing schedules and so on, that can't be accomplished through Web-based tools.	Ideally, all administration should use Web-based tools. If there are some administrative functions that require other software, you should know about them and make sure that they work in your context.

Question	*Notes*
Can you do a global search and replace throughout the repository?	The CMS should support search and replace through free text and metatags, assuming that appropriate permissions and security rights are observed.
Can you automatically verify and manage broken internal and external links?	This is a nice feature to have.
What user roles and rights model do you use, and what tools do you provide for managing and manipulating them?	There are many variations on how user roles and rights are configured, including those that tie into your own network directory services and ones that support "inheriting" permissions based on a hierarchy of user roles.
What's your approach to data synchronization and replication across multiple databases? At what level is synchronization managed and communicated?	It's unlikely that it does a lot of this automatically. Instead, you're most likely to get replication tools that you must control to ensure synchronization.

Workflow

CMS products vary from no workflow support to integration with third-party tools to strong native workflow modules. The questions in Table 16-14 can help you correctly place the products that you survey on the spectrum.

Table 16-14
Workflow-Related Questions

Question	*Notes*
What workflow tools does your product include?	You're looking for the vendor's description of its workflow tools. You may end up with a paragraph or two of marketing information or a good overview here.
With what workflow tools does your product integrate? What support do you provide for the integration?	Even if a product has internal built-in tools, and especially if it doesn't, it's important to know whether it enables you to use strong third-party workflow systems. If you're thinking of integrating in a workflow system, make sure that you get the company to answer all the questions in this table from a support perspective, as opposed to an implementation perspective. You want to make sure that events and content status transitions in the CMS can correctly trigger events in the external workflow system.

Continued

Table 16-14 *(continued)*

Question	***Notes***
Please describe the internal or external inputs that can be used to trigger workflow steps.	Look for the capability to trigger workflow steps by an external process, such as an e-mail message arriving or a file being added to a directory.
Please describe how your system handles the configuration of user roles for workflows.	You want to see that the product has strong user interface that gives administrators complete and easy access to the functions that the product offers.
Do you support parallel as well as sequential workflows?	In a parallel workflow, two people work on an item at the same time. In a sequential workflow, one person finishes before the other begins.
Please describe the range of system activities (such as content submission or deletion) that can be supported by workflows.	You want to find out whether the system can handle requests for content, content submission, content addition, deletion, edits for existing content and content bundles, content reviews, content approvals, content migration, migration approval, and so on.
Please describe the content or system attributes that can be used to drive the rules for automated workflow routing.	In other words, what can the system look at in deciding what to do next with a piece of content? (Submitter, content type, and status are some of the usual attributes.) The best answer is that the workflow module can use any piece of metadata that you create.
Please explain or demonstrate the process for creating or modifying a workflow process.	You want to make sure that creation of workflows doesn't require programming skills. The slickest systems have a visual workflow designer with drag-and-drop manipulation of processes. Make sure that either the interface is very complete or that it's extendable.
What types of content collection, content management, content publishing, and administrative activities can be handled by an automated workflow process?	You want to know whether the workflow module can handle all authoring and publishing requests, including content submission, content addition, deletions and edits, content review/edits, content approvals, content migration, publishing, and testing.
Do you have the capability to include an external service provider (HTML provider, graphics, or translation services) in the workflow?	If it's an issue for you, make sure that the workflow tools can manage people with only a slow Internet connection to your CMS, no access to your LAN through your firewall, or only an e-mail connection to you.
Can you view all content that's in a particular state for a particular user?	The best systems have rich queues that show each user what task she must accomplish.

Question	*Notes*
Can you add comments at the end of each workflow stage?	This is a handy feature to have. (If not built-in, you probably want to customize the system to include a comments field.)
Do you maintain an audit trail for all workflow processes?	It would be super if you could roll back a piece of content to any step by selecting that step in the audit trail.
Using your product, can you integrate e-mail into the workflow processes?	Some systems can run the entire workflow process through e-mail. Users receive notifications in their inboxes and then click links in their messages to complete tasks in the CMS.
Is there an administrative override to automated workflow processes?	This feature is a must if you need to stop or redirect workflows in process.
At what content level can you create workflows?	Most systems can attach a workflow to a file. The better ones can attach a workflow to a component (or whatever name are given components in the product). The best ones can attach a workflow to a particular element of a component. The body element, for example, can have a localization workflow attached to it to make sure that it gets translated. The rest of the component may not need to participate in that workflow.

Management integrations

In all questions about integration, it's important that you don't settle for the typical answers, such as "We have a partnership with X," or "I think someone has done that." Rather, ask for the documentation describing how the integration was done. If that fails, ask for the name of a person who's done the integration and talk to her about how it went. At the very least, get specific information about the system on which the integration was done to make sure that it's similar to your own.

Here are some of the external systems with which you may want to integrate:

- Document management systems.
- Media and asset management systems.
- Operating system user registries (such as Microsoft Windows 2000 Active Directory).
- Enterprise resource planning (ERP) systems.
- Custom databases that your organization uses for catalogs and other content storage.

For each type of system with which you expect to integrate, tell the vendor the name of the product that you're using and what platform it's running on, and ask the vendor to tell you specifically how the product interoperates with yours.

Table 16-15 lists some other integration questions that you can ask.

Table 16-15
Integration-Related Questions

Question	*Notes*
What capabilities does your system have out of the box and with additional programming to connect to and share data with other databases?	The best case is a nontechnical interface for connecting to simple databases that can be used by administrators, plus the capability for the connection to be extended by programmers using a standard language.
Can you launch other applications from within your system? If so, how? What outside applications are supported?	Many products integrate with Microsoft Office products and Adobe Acrobat. The CMS product also may be capable of launching any viewing and editing tool that the system recognizes based on a file extension.
Do you follow any open standards for data access and data sharing?	Some products follow the XML schema or other open standards such as ODBC for data exchange.

Version control

Not all CMS products have a version control module. For those that do, the questions in Table 16-16 can help you decide whether it can work for you.

Table 16-16
Version Control-Related Questions

Question	*Notes*
Can users check content into and out of your repository?	This is the most basic form of version control.
Does your product save earlier versions of content that can be restored later if necessary?	Make sure that it's reasonably easy for a user with permissions to restore the previous version without the need of a database administrator.
What level of granularity is supported for checking content in and out?	File level is most common; component level is the most useful; and element level check-in and check-out is necessary for more complex tasks, such as localization and collaborative authoring.

Question	***Notes***
How is locking handled if content is checked out?	In the best case, you have the option to lock the component being checked out and, if you want, also its siblings or parents in the repository hierarchy.
Describe the different levels of granularity at which content can be rolled back to an earlier version. How do you support rollback of a complete site, an individual page, or a single content component?	You may need any of or all these kinds of rollback functions.
Can you perform differencing between two versions of the same content?	The best system enables you to put two versions side by side and compare each change.

Reporting

There are any number of reports that you may want your CMS to provide. Table 16-17 has some of the questions that you can ask to find out what sort of reporting a product supports.

Table 16-17
Reporting-Related Questions

Question	***Notes***
Describe the logging capabilities of your system. What types of activities can be monitored, for both content producers and consumers? How would you track usage by user, group, or geographic area?	In the best case, you can track and log users, their time of access, and actions performed (such as downloading a particular version of a file or checking out/editing a file). You can also track system usage by user, group, geographic area, and so on. Logging and analysis usually can be performed on a Web server by third-party tracking products. You can use these programs for tracking the use of your Web publications. In addition, you should know what kind of logging the CMS provides for staff activities.
Describe the workflow tracking reports that your system can generate.	Workflow reports that are most commonly used show each component or document and its workflow status, along with other supporting information. Or you may want reports that show all components that are at a particular workflow stage.

Continued

Table 16-17 *(continued)*

Question	***Notes***
Can your product track usage and fees associated with copyrighted media elements?	If you buy media, you probably need to track and report on where it was used and how many times it was accessed or downloaded. This is rarely part of the product, but it ought to be possible to integrate into the product.
Can your product generate a complete site map of the Web sites that it produces, including page names and statistics (size, components, and so on)?	These sorts of reports are very useful to publication teams.

Disaster recovery

If you ever have a catastrophic failure of your system, you're going to be glad that you asked the questions in Table 16-18.

Table 16-18
Disaster Recovery-Related Questions

Question	***Notes***
What features does your system support for content archiving and disaster recovery?	Features to look for are versioning, rollback, and automated backups. These features may be part of the database software that the CMS uses but could be controlled from the CMS administrative interface.
Can you provide any statistics on the failure rate of your product, such as average time between failures or the average percentage of system availability?	Some can provide this information and others can't. It's a good question to ask as you get to checking references.
Describe any functional and architectural features of your product that are optimized to maximize the reliability of your system.	This is an open-ended question to give the company a chance to brief you on its reliability. Some companies can provide technical white papers that describe their system architecture and reliability features.

Security

The questions in Table 16-19 can help you assess the user and access features of the CMS.

Table 16-19
Security-Related Questions

Question	*Notes*
Describe how your system can leverage existing operating system or directory services security profiles to avoid needing to re-create definitions and duplicate security maintenance. With what servers and services can you integrate?	In the best case, whatever permissions or user groupings are already available in your operating system also are available and extendable in the CMS.
Do you support a hierarchical security model?	Especially if the system doesn't tie into existing security systems, you need the capability to group users and have them inherit permissions from parent groups.
Can your product help us enforce intellectual property rights and security permissions by controlling the capability to publish media?	If you work with content that has restrictions because it's copyrighted or because it's secret, the CMS may enable you to create and enforce a policy.

Performance

The questions in Table 16-20 help you assess how well the product can scale to the size that you need it to be.

Table 16-20
Performance-Related Questions

Question	*Notes*
Describe the minimum and recommended hardware, software, and network requirements of your product at the level of usage that we expect.	The trick here is to specify your level of usage. Usage has two parts: the usage of your Web publications and the usage of the CMS by your staff. Check both.
Please provide examples and references that demonstrate your system's capability to scale to an enterprise-wide, global implementation. These examples should illustrate your capability to scale to multiple instances on multiple servers, to scale beyond a single Web farm, and to fully support load balancing.	If this is an important criterion for you, asking this question provides an opportunity for the company to present you with its scalability story. Check the story that it provides for architectural validity and applicability to your situation.
Describe any data size or performance limitations of the content repository. Include any known limitations, such as maximum number of rules, number of unique components that can be stored, number of concurrent content contributors that can be supported, or number of concurrent content consumers that can be supported.	Make sure that you check these capacity limits for staff as well as for concurrent audience users of your major Web publications.
Describe your mechanism for content replication to regional sites.	Replication usually consists of a master version and copies that are distributed regionally to improve performance.

Publishing criteria

The questions in this section help you assess how many of your publishing needs the product can cover out of the box or with customization.

Templating criteria

The questions in Table 16-21 canvass the product's templating capabilities.

Table 16-21
Template-Related Questions

Question	***Notes***
Describe your overall approach to publication templating?	This is your chance to get an overview.
What programming languages can be used to develop Web templates?	The right answer is the languages that you use. Barring that, open languages that have a wide skill base in the programmer world are the best choices.
Do you support developing templates for non-HTML publications?	Sadly, most don't. One day, CMS products will support templates in word processing and graphical layout program formats as well.
Can your system process templates in a batch mode, producing complete static sites?	In other words, can the CMS render a static, Web-based publication as well as serve dynamic sites?
Can your system process templates on demand, producing dynamic sites?	Not all products can do this.
Can your system mix batch and on-demand processing to produce sites that are partly static and partly dynamic?	This is the ideal. You want to produce as much of a publication as possible statically for performance and robustness, but still produce other parts dynamically for the flexibility that it offers.
Describe the methods that you support for dynamically creating site navigation components. How are searches, links, a table of contents, or other navigational aids built by logic in the templating system?	You want to see that there are some tools for this but, moreover, that the system is open enough to enable you to extend it to whatever navigation that you may want to produce.
Does your system support the concept of component and navigation templates? If so, how?	Refer to the section "Publications and Templates," in Chapter 22, "Designing Publications," for more information.
Do your templates support integration with outside systems? If so, how?	Some tools may be provided, but you really want to make sure that the CMS doesn't get in your way as you do this yourself.
How is personalization accomplished in your templates?	This is a very open-ended question that ought to yield a general discussion of the product's personalization tools and integrations.

Content deployment

CMS products have a variety of capabilities for distributing content from the central repository behind the organization's firewall to servers inside and beyond the firewall. The content deployment questions in Table 16-22 are weighted heavily toward the Web because that's where most deployment happens today.

Table 16-22
Content Deployment-Related Questions

Question	*Notes*
How can your system be used to deploy content files to remote Web servers?	This is the basic deployment functionality. Surprisingly, many products expect you to do this with third-party products.
How can your product be used to deploy database records to remote databases?	In addition to files, you may need the CMS to send data to databases outside the CMS repository.
Describe or demonstrate a scenario illustrating how content is deployed from a staging server to a production server.	A staging server enables you to test a Web site before it goes live. Find out whether the CMS can support staging. (Also notice that many products' licensing models require that you purchase separate licenses for each server. The staging server counts as one of those.)
How can your system deploy appropriate content to various destinations, based on properties of the content?	You may, for example, want to key on various values within content components (language, for example) to determine which server to deploy the content to. Can the CMS help you here, or do you need to write your own routines?
Does your product enable you to assign default values for file directories based on content type?	It's useful to have all pages of a particular type deploy automatically to a particular output directory.
Can your product automatically deploy files referenced in a field?	If a file (an image, say) is referenced inside text that's being published, it's a good idea to have the CMS deploy the file to avoid a broken reference.
Can you set the publish and expiration dates of content to be published?	It's useful if you can have the CMS automatically release and remove content based on the rules that you establish.
Can you publish on a set schedule without manual intervention?	This is what you want for small, frequently published material (weekly e-mail broadcasts and syndications to other systems, for example).

Publication platforms

With the questions in Table 16-23, it helps to ask for specific examples of users who are doing the kinds of publications that you want. Don't stop with the answer, "Sure, we can do that."

Table 16-23
Platform-Related Questions

Question	*Notes*
Can your system produce broadcast e-mail?	The system should be capable of integrating to your e-mail system and pushing content to it. Alternatively, it may integrate to a third-party bulk e-mail system.
Can your system create individualized e-mail messages?	As opposed to bulk e-mail, in which the same message is sent to a lot of people, individualized e-mail is where a lot of different messages are sent to one or a few people each. This capability ties the personalization functionality to the e-mail creation functionality.
How can your system be used to produce print publications?	You want to focus this question more on the specific kind of print publication that you need to produce.
Can your system create publications based on WAP and other XML-based formats?	Again, have a good idea of the exact publications that you want the system to create.
Can your system syndicate content?	Syndication capabilities include creating format-neutral (generally XML) content chunks and deploying them to FTP sites, remote servers, or to mass storage devices (hard drives, CD-ROMs, DVDs, and the like).
Can your system create content for proprietary electronic formats, such as Microsoft Help, Macromedia Director, or other CD-ROM platforms?	You should query specifically for any non-HTML electronic platforms that you may need.
Can your system create downloadable documents in Portable Document Format (PDF) or other proprietary formats?	PDF is good for downloads that can't be modified by the user. Query for any others that you may need.

Personalization

The questions in Table 16-24 help you figure out the types of personalization that you can expect to get from a product.

Table 16-24
Personalization-Related Questions

Question	***Notes***
Does your product ship with any personalization tools?	Some don't, although most don't easily admit it. Others have extensive offerings.
Does your product integrate with any third-party personalization tools?	Most at least have marketing alliances with personalization companies. Look for documented integrations.
What kind of user interface do you provide for creating and modifying personalization rules?	Some products have very well-developed point-and-click interfaces for rule building. Others require a programmer to code the personalization rules – not a good long-term solution.
Describe or demonstrate the process for defining personalized experiences. What skills and training are needed to work with your personalization engine?	See whether you can get a concise description of the overall approach to personalization that includes a discussion of how it's actually accomplished.
How does your system facilitate the collection and storage of user profile data?	Some products have functions for monitoring the user's actions and their responses to questions and then storing the results in an accessible profile – but don't count on it.
Can your product receive syndicated data from third-party suppliers of customer profiles?	You may want to supplement the information that you collect about your users with data from commercial suppliers. Can the CMS help this happen?
What forms of data are accessible to your personalization engine?	Good answers include user session data, HTTP header information, page history, all content metadata, user profiles inside or outside the CMS repository, user groupings, and personalization rules.
What facilities do you have for producing individualized print publications or other non-Web personalizations?	Most of the action today is in dynamic HTML page building. The same tools could be used to make any other kind of personalized publication, provided that there's template support for the publication format that you need.
Do searches, links, a table of contents, or other navigational aids reflect the personalized content that's available to a specific user? If so, how does it work?	For relatively sophisticated personalizations, the system should be able to modify these access structures automatically, based on the personalization rules.

Question	***Notes***
What sort of personalization rule-building process and user interface do you employ?	Some products have well-developed user interfaces for point-and-click personalization rule building.
Can your system perform collaborative filtering?	Collaborative filtering is the process where the system recommends content based on requests of similar users (an approach popularized by Amazon.com).
Please outline the kinds of personalizations that are possible based on user profiles.	This is an open question that gets the company to talk about how personalization interacts with profiles. The product may enable you to define profiles, for example, and then target content to them.
Please outline the kinds of personalizations that are possible based on a user's actions on a Web site.	This is also called behavior-based personalization. Some systems track a user's behavior and modify the subsequent pages accordingly.
Can your system facilitate the creation or running of push campaigns?	In a push campaign, the system actively presents certain content on a Web page or in another form to a group of selected users for a specified period of time.

Publication integrations

There are a lot of systems to which you may like to tie your Web publications. As with the management integrations, make sure that you get as much detail about the ones that you care about as the company can provide. The following list features some of the systems to which you may want to connect. It may be that the CMS does some of these functions as well, so phrase your question so that the CMS product companies tell you how much of the function their products do, as well as what third-party products they integrate with if you want to integrate with one or more of the following systems:

- Ad management and server products.
- Brand management tools.
- Campaign management and promotions systems.
- Merchandising systems, including up-selling and cross-selling tools.
- Customer relationship management systems.
- Lead generation products.
- Surveys and questionnaires.
- Web access and security tools.
- Data logging and mining tools.

- Product configurator systems.
- Decision tree tools.
- Download packaging tools (which automatically bundle together and compress electronic content for efficient download).
- Chat products.
- Bulletin board and threaded discussion systems.
- E-mail acceptor and automated response systems.
- Event planning and registration tools.
- Webcasts and streaming technologies.
- E-commerce and financial transactions systems.
- Credit card transaction systems.
- Any of the enterprise systems that I mention in the management integration section, earlier in this chapter, that also need to be accessed dynamically from the Web.
- Conversion systems for on-the-fly conversion of content (for example, from XML to HTML).
- Custom database applications and custom programming objects that you may have created or may use on your Web site.
- Web servers and Web application servers.

Web UI

The Web is guaranteed to be one of your publication platforms, and Table 16-25 lists questions that can help you assess how much help you can expect from the CMS product in building Web sites.

Table 16-25
Web UI-Related Questions

Question	*Notes*
What types of searching and indexing capabilities can be made available to an end user viewing a Web site produced by your system?	Searching isn't necessarily performed by the CMS. For static sites, a standard third-party search engine may be fine. For dynamic sites that are produced by the CMS on the fly, you need integration between third-party search engines and the CMS search functions.

Question	Notes
What particular support do you have for producing browser-independent Web pages?	Some products include browser sniffers and supporting code to make serving browser-specific pages easier.
Does your product support user-session management, including cookies or database caching of user profiles?	This is important stuff for personalization and customization.

Web architecture

If you have the CMS active on a Web site, ask the questions in Table 16-26 to see how the product integrates into your Web infrastructure.

Table 16-26
Web Architecture-Related Questions

Question	Notes
Which Web servers are supported by your product?	Yours, you hope.
Does your product include a proprietary Web application server?	Some products still do have their own application servers. You want to make sure that the product doesn't lock you into its own application server but enables you to integrate to one of the better commercial application servers.
With what Web application servers does your product integrate?	For any product that you expect to use for more than the next year, the company needs at least to have plans for how it intends to integrate it with industry-standard application servers.
What databases can your system use?	The right answer is the ones with which your technical team is familiar.
How does your system facilitate or perform content caching?	Caching can happen at many levels. Make sure that you understand the difference between the caching models used by your various contenders. Make sure, too, that the company can demonstrate Web-server performance commensurate with the amount of traffic that you expect.

Continued

Table 16-26 *(continued)*

Question	***Notes***
How does your system facilitate or perform database connection pooling?	This may be handled completely outside the CMS. Know which parts are performed within the CMS and which parts depend on a third-party application server.
How does your system facilitate or perform server load balancing?	You'd like at least some support from the CMS for this.
How does your system facilitate or perform user session management?	Does the system rely on cookies, or does it provide other methods of tracking user sessions?
Can your system leverage user-profile information stored in directory services? Is any custom development needed to implement this functionality?	Probably, there's some integration effort needed.
List the LDAP servers with which your system can integrate.	This is a concern if you're using LDAP services in your environment.

Summary

If it's done well, system selection can galvanize the entire organization behind a platform and approach to your CMS. If done poorly, it can create wide rifts between groups and result in a system that was nobody's favorite. To keep the selection process on the good side, do the following:

- Understand the position of product companies. Trust the people who are trustworthy but understand the constraints and goals that guide their behavior.
- Develop a process before you need it. If you're in the heat of decision, there may be too little energy and good will to yield a fair and binding selection process.
- Develop a set of selection criteria that ask the key questions that you need answered to compare and choose products.
- Be diligent. Rather than jumping ahead, start as wide as possible and continually narrow the field until only one choice remains.

- Be official. Include the people from outside the process who need to understand and support the process as well as those inside the process who decide. Herald your final decision and your great process.
- Update your ongoing project deliverables, such as your project plan and your risk assessment.

In the next chapter, I stop talking about planning and start talking about doing a CMS.

✦ ✦ ✦

[illegible]

Summary

[illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]

CHAPTER 17

Implementing the System

After you've designed your publications, content, and system and you've chosen an appropriate CMS platform, you're ready to implement your project. To begin, you must turn your requirements and designs into a set of specifications.

Following specification, work can begin on putting the system together. You may divide the work by the three phases of content management (collect, manage, publish) or by functional tasks (programming, authoring, publication creation, editing, and so on.). Either way, you're best served by managing closely to your specs and project plan.

In this chapter, I present an overview of the processes that you may use and the deliverables that you may create to implement a CMS. Please note the following:

- I can present only a general model for implementation. Without knowing the particulars of the system you're creating, it's difficult to provide specific advice.
- I assume that you're working with a commercial CMS product and that your job is to implement it rather than create a CMS from scratch. If you're creating your own system, you can still use this information, but you may find the information in Part V, "Building a CMS," more helpful.
- If you work with a product's professional services group to do implementation, they no doubt have their own methods and deliverables. If so, you can use this information for comparison and general guidance.

✦ ✦ ✦ ✦

In This Chapter

Taking stock of the process and project so far

Looking at the specifications and plans of a CMS implementation project

Providing an approach to staffing and implementing the project

✦ ✦ ✦ ✦

Looking at the Process So Far

Maybe you think that I want you to put a whole lot of process in place before you get to the real stuff. You may feel that you go through a lot of talking and planning but no doing until late in my process. You're right, sort of. Contrary to some, my major belief is that the technology to do content management is much less a problem than the attitudes and processes of the people who do content management. The technology is immature, indeed. It requires a lot of added customization and can't do all that you want it to do. Still, it can do most of what you want, if you know what you want.

In many other computer applications, it's much easier to know what you want. In a financial application, for example, you want to put a user interface and storage mechanism around all the finance operations that accountants, bookkeepers, and financial officers have been doing for years. In scientific applications, you want to create an algorithm that collects data and then processes it according to the equations you've devised. Even in a standard Web application, you know what you want to do. You want to store information of a known structure in a database and then access it for use on an HTML page.

The simple explanation for what you want to accomplish with a CMS is as follows:

> You want to create a system that harvests valuable information and functionality from the organization and its allies and delivers it to a known set of audiences wherever, whenever, and however they want it.

This simple definition immediately forces you to confront a number of questions that have nothing to do with technology. And until you've gotten past these questions, no amount of technology in the world can help you.

Now, to my grand language here, you may reply, "The simple explanation for what I want to do is to make my Web site easier to create by putting it in a database!" I wouldn't question this attitude. Maybe that's all you need to do. I'd just ask you to consider the following two points:

- You've described a Web application, not a CMS. Maybe you're biting off too big a solution to a smaller problem by considering a CMS.
- If you think that you need a CMS for what it offers beyond laying out stored content on a Web page, you very well may be backing yourself into the same nontechnical issues that I suggest you consider. If you need a CMS to organize authoring as well as to produce Web pages, for example, you must take at least some of the prior mandate and logical design steps that I propose, simply to figure out who your authors are and how you can work with them.

It's still early in the electronic Information Age. The idea of orienting your organization around the collection and distribution of content (both information and functionality) is new enough that it requires a lot of education and time to get it started and still more energy to keep it going. Over time, perhaps, as the ideas sink in and become part of the fabric of organizations, much of the groundwork that you need to lay today becomes a given in most organizations. The life of a pioneer is spent breaking ground that later settlers don't even notice.

Looking at the Project So Far

I assume that, happily or grudgingly, you've done most of what I've proposed prior to beginning implementation. If you have, you've accomplished a lot. You've done all that you can do to ensure that the organization is ready for a CMS and can make use of it. You've worked completely through what you want it to do and now simply must make it do that. In particular, you've done the following:

- **Built a strong consensus in the organization** for what the system should accomplish. More to the point for implementation, you have a fixed set of goals that are measurable. You can gauge and direct your implementation based on these goals to make sure that you hit and then surpass them.
- **Organized a collection effort** that ensures that the CMS has a steady supply of high-quality input. More to the point for implementation, you know exactly what must be done to the existing content to get it ready for the system. You also know what sorts of tools and processes you need to assist your authors and acquirers move content efficiently into your system.
- **Produced a content schema** that unites all your content in a single, overarching structure. More to the point for implementation, you've produced a relational database or XML schema that you can quickly turn into the structure of your repository.
- **Designed a set of publications** that deliver the right content through the right channels. More to the point, you've defined the output of the system in such a way that it can be programmed into a set of templates that can produce those publications automatically.
- **Chosen a CMS product** that meets as many of your needs as possible. More to the point, you've created an alliance with an organization that can help you make this system work. You've also produced the project plans and architectural diagrams that feed directly into your implementation specs.

If you've followed the process to this point, you've done a lot of work, and all of it can help you better and more quickly implement your CMS.

Techniques for Getting the Job Done

Implementation is largely a process of making your plans come true. It's equally a process, however, of making your plans consistent with the reality of what happens as you try to make them come true. It's just as important to know when to change plans as it is to know how to stick to them.

Cutting back

At the point that you begin implementation, reality sets in. You become all too aware of the magnitude of the collection effort, the system setup, and system integration, plus the complexity of the publications that you want to produce. As this happens, be prepared.

You inevitably plan a bigger project than you can really do. If you know that this stage is coming, you needn't panic and run from the entire endeavor, and you needn't stiffen your upper lip and agree to do it all.

The attitude that I believe serves you best is that the requirements and design stages defined the foreseeable goals for the organization and that the implementation and roll-out stages define what's immediately doable for the organization. Interestingly, this attitude both closes down and opens up the project, as follows:

- Just because something isn't immediately doable doesn't mean that it can't be done. The only question is when. The parts of the design that can't be done in the first version of your CMS are the starting points for the second version.
- Just because something wasn't foreseen in the design doesn't mean that the organization doesn't want to do it. The design process itself needs to continue. As you move into implementation, and especially as you begin to use the system, new content, system features, and publications become apparent.

Cutting back, then, is not only necessary at implementation time, but it's also just a part of the ongoing design process.

Cutbacks may be natural, but they're not easy. Someone's desire is thwarted for this version, and she isn't pleased. In addition, you must go back and redo some amount of the planning process to account for the change in project scope.

You have a few tools for coping with unhappy campers. First, you have the prioritized requirements that you originally collected and had approved. Your cuts should start at the low-priority content and publications, to be fair to all and to avoid conflict. Second, you have your sponsor group, which is in place to help with just these sorts of events. Make your case for cuts to your sponsors, not to the people whose requirements you must deny. Finally, you can console hurt feelings with the assurance that a high-priority implementation in version 2 can produce a much better result in the long run than a low-priority implementation in version 1.

Tip

Resist the temptation to keep all content classes and publications and just do less volume. In a CMS, most of the work consists of setting up the component classes and publication infrastructure. So cutting volume rather than types of publications and content gives you less bang for your cut.

You also must decide how far back in your planning to make the cut, as follows:

- **Do you simply need to scale back the logical design?** If so, you can get by with some markup on your design documents, showing which parts must be postponed. In addition, you must make whatever changes are needed to your project plan.
- **Do you need to drop requirements?** If so, you need to get the requirements reworked and approved again. Then you can mark up the design documents and project plan.
- **Do you need to change the goals?** If so, you need to reconvene the mandate group to make the change. Then you can make the corresponding changes to the requirements and get them approved again. Finally, you can change the design documents and project plan.

Obviously, the further back you go in the process, the more work it is. On the other hand, the further back you go, the bigger the change you can make. Personally, I think you derive collateral benefits from reengaging your mandate or requirements group. Doing so gives members a chance to stay active as a part of the process and to reassert their support. You want this to remain everyone's project.

Considering one project vs. many projects

Up until now, I've referred to "the project" as the entire CMS initiative. And it behaved as that. You had one team, whose members were all working together to create consensus and design the CMS. In the implementation phase of a CMS project, however, it may suit you better to think of a number of simultaneously running projects, each with its own project plan and team. This approach has a number of advantages that I believe outweigh the extra effort it entails — namely, the following:

- **It helps you manage the complexity** of the project at this stage, where a lot of different teams work more individually than with other teams. Rather than managing tasks, you can manage entire efforts and enable the team project managers manage to the tasks.
- **It enables you more easily to scale** to a larger project team from the smaller team that you've needed so far.
- **It helps you incorporate external resources**, such as the product company's professional services group and other third-party development and processing companies.

- **It isolates problems and setbacks.** If one team is having problems, it needn't bog down the rest of the effort.
- **It makes for more effective meetings.** Most meetings need only be within a particular team.
- **It prepares you for later**, when you may want to move to a functional project organization. (See Chapter 12, "Working within the Organization," in the section "Creating functional teams," for more information.)

Note Even if you share staff extensively across projects, it's still worth the effort to differentiate the projects for the organizational advantage that it gives you.

If you use this method of project management, you must be very careful to understand and closely manage the dependencies between groups. As one example among many, you must make sure that, at the time an integration team is ready to test a connection, the CMS team has the system running.

The key to this approach is to keep a master plan and schedule that has each team as a black box except for where its effort connects to that of other groups.

Following are some of the projects that you may want to create:

- **The CMS infrastructure project**, which is responsible for installing and configuring the CMS product. This team has mostly IT people on it and may be led by the future CMS administrator. (See Chapter 11, "Staffing a CMS," in the section "CMS administrator," for more information on the role of an administrator.)
- **The template project**, which is responsible for template design and programming.
- **Custom development projects** that create auxiliary applications for features that the CMS itself can't provide.
- **Integration projects**, which are responsible for connecting and establishing communication between the CMS and other existing systems.
- **Content-processing projects**, which are responsible for content preparation.
- **The test project**, which ensures that all the pieces work together and are as bug-free as possible.
- **The documentation and training project.** Although I class these activities as rollout and not implementation, it's not too early to start the project. Implementation is where most of the information this project needs is created.

Detailing the implementation process

In the sections that follow, I lay out the major parts of a CMS implementation project. If you've done either a custom software development process or an enterprise integration process in the past, part of this process probably seems quite familiar. I've drawn from both software and integration to create these materials. In addition, I spin the entire process in the direction of collection, management, and publishing to make it solidly a CMS implementation process.

Preparing specifications

As in any development and integration project, the core of the CMS implementation is a set of specifications that tells the doers (programmers, IT specialists, data processors, and so on) what to do and how to do it. In a CMS system implementation, the specs merge the logical design that you created previously with the functions and procedures of the product that you chose.

The physical design of CMS includes the software, configuration, and hardware that you need to turn the logical design into reality. If you're lucky, the majority of the software doesn't need to be created from scratch; the majority of the configuration can be done in a good user interface; and the majority of the hardware is stuff that you already own — but don't count on it.

To prepare specs, you can start from your logical design. For each part, you can ask, "What must we do to make this happen given the system that we chose?" Write the answer down in the most explicit terms that you can use. The logical design is what you want to do, and the system (plus any additional programming or processing resources you have) is the tool set that you have to accomplish this goal.

Note Many people who don't have the benefit of having done a logical design turn this process around. They ask, "For each thing that the system I bought can do, what do I want it to do?" This process gets you to a system, but it's unlikely to be a system that addresses any more than your most crucial needs. But, hey — it's a start!

Following the logical design, your specs can detail the following parts of your implementation:

- **Your collection system**, including all the tools that you use to author or acquire content, all the CMS interfaces (such as Web-based forms) that you use to load content, all the conversion processes that you use to transform content to your master format, and all the metatorial processes that you use to completely tag each kind of content.
- **Your management system**, including the way that you install the CMS product; the way that you set up your relational, object, or XML repository; the way that you configure all the options of your CMS (to create component classes, users, workflows, version control, and so on); and the way that you connect and integrate the management system with other data sources.

- **Your publication system**, including the templates that you program to create the publications; the Web infrastructure that you create to deliver Web publications; the other software and process that you need to create other publication types, the personalization system setup, and rules; and the way that you call or otherwise integrate with the range of non-CMS data sources and functionality components that your publications require.

You've probably seen enough of these terms by now to know what they all mean (another reason to do a lot of homework before launching into implementation).

I've seen any number of styles and layouts for specifications, most of which seem fine (although people always seem to find a favorite to argue for). The only things that I feel any motivation to agitate for are the following:

- **A numbered section heading style** (where, for example, 1.2.3 is the third sub-subsection under the second subsection under section 1), which makes it so much easier to refer to sections and makes the hierarchy of the specification show through quite easily.
- **Enough concept** that the reason why you're creating a particular function doesn't get lost. You should always begin a section by stating why the function described is important and what goal it relates to. Then, as readers work through the specification, they can make their inevitable tradeoffs and enhancements based on a solid knowledge of the real goal of the function.
- **Enough detail** that the person doing the implementation knows exactly which part of the CMS or third-party application is being used for a function and what part she's expected to custom code. I've found that it's rare to put too much detail in a specification. You almost always run out of time and energy long before anyone would say that it's too specific. Nonetheless, make sure that leave room for the creativity and initiative of the implementer to have an effect.

Tip

The vast majority of specifications I've seen are created as print documents. If you're smart, you create yours in HTML (or better, XML, if you can manage it). This gives you a lot better chance of repurposing your specs later into documentation and training. A little push now to make your authors learn and use an electronic authoring environment (or even a CMS) to create their specs saves you a lot of time later when you really need it.

Installing the system

Even if I knew which system you chose, I wouldn't know much about how to install it. IT, from the standpoint of system setup, isn't my greatest strength. What I can help with is the attitude that you should take toward installation. Keep the following points in mind:

- **Be prepared for it to be difficult.** A CMS has its tentacles into a lot of systems, including databases, Web servers, operating system services, user and other system registries, LAN connections, and the host of other systems with which you expect the CMS to integrate. Your installation may go without a hitch, but just as likely, even with the expert help of the product company, it takes a long time and doesn't work correctly the first time.
- **Document the installation.** Especially if the documentation from the product company is lacking, keep your own log of the options that you need to choose in the operating system. You're sure to thank yourself later (and so do others) if you need to reinstall the whole system.
- **Don't be afraid to play.** If you treat the system like a fragile china dish that you're afraid to break, you quickly develop a can't-do attitude that prevents you from trying half the things that you need to do over the life of the system. Instead, break it on purpose a few times to see how it responds. Become accustomed to seeing the system crash or stop connecting after certain parts are changed. The more that you play before the system is live, the better able you are to quickly fix it after it's live.
- **Start early.** Don't begin to set up the system a week before you need to start filling it. Install it a bunch of times, months before it goes live. Using your diligent documentation and your reckless disregard for crashing and trashing the system, you should spend as much time as you have mucking around to get comfortable with the system.
- **Make a test bed.** Create a hardware and software set that you can use to test installation and configuration options in safety.
- **Get comfortable with flakiness.** CMS systems aren't mature software applications. In addition, especially on Web servers, you have so much interconnection and dependency that something is always broken or limping. Rather than complaining about it, get used to it and enjoy the challenge of keeping all the parts in place.

You can't change how difficult it is to install a CMS, and you can't make it much less flaky. But you can come to terms with the difficulties and not let them hold you back.

Configuring the system

After you have a blank CMS system installed, you need to configure it. Configuration is the process of setting the options and parameters that structure the CMS and preparing it for your purposes.

If you're lucky, your configuration can be done through a convenient user interface. To create a workflow, for example, maybe you can just click the workflow button and begin to drag tasks onto the screen. If you're less lucky, you must do your configuration by using one or more of the following options:

- **Text files**, where you type parameters and options by using anything from a simple to a very arcane syntax.
- **System registries**, where you enter configuration information into the interface (provided by the operating system) for specifying options.
- **Third-party tools**, where you use the interface (or lack of interface) provided by add-on products to type your parameters. Some systems, for example, require you to do a lot of database configuration in the database system itself.
- **XML files**, where you enter parameters into a set XML structure.
- **The software that you create.** If you write a batch-upload program, for example, you must figure out how the user configures it to post files to the appropriate directory. You can put in a nice user interface or use one of the less easy techniques. After creating a few configuration interfaces yourself, you may understand better why so much configuration doesn't have a better interface to go with it.

Of course, I can offer no specific advice on configuring a particular system, but here are some tips to help you take a good perspective on configuration:

- **Configuration is behind many problems.** A configuration is a specific piece of information that the system needs to accomplish a task. If the system isn't doing what you want it to do, ask yourself, "What information is needed to complete this task and where is it likely to be hiding?" Directory names, user names, on/off choices, and display parameters are typical examples of the many bits of information in a system configuration.
- **You can take it back.** In the vast majority of cases, if you mess up a configuration parameter, all you must do is change it back for the system to work again. Some systems have buttons called *defaults*, which automatically restore the configuration to an "out-of-the-box" state. Don't be afraid to play around with configuration parameters (after you've written down what they were before you started). They're a low-risk way to get to know how your CMS works.
- **You must look for them.** Many CMS systems are notoriously bad at telling you where the configuration you're looking for is hidden. If you know something should be configurable, make sure that you check all the sources that I mentioned earlier (text files, the system registry, and so on) before giving up. Do yourself and everyone else a favor by documenting any parameters that you find in out-of-the-way places. If you needed it, someone else does, too.

Doing content processing

Content processing can take a lot of time and effort to do well. It can also spiral out of control quickly if it's not well organized.

Cross-Reference All of Chapter 30, "Processing Content," is dedicated to content preparation, so I don't dwell on the details of the process here.

Aside from the process itself, the main thing to understand about content processing is that it's a project unto itself and should be treated with the same respect and planning as any of your custom development or integration projects. In fact, depending on how much content that you need to prepare to launch and run your system, it may be your largest project. It may require more people and programming than any other single deliverable. Why?

If you have 1,000 items to prepare (not a particularly high number) and each takes one hour to process, that's a 1,000-hour project. If you can spend 100 hours of programmer time and 50 hours of manager time to cut the time down to 15 minutes per item, you can save yourself a lot of money. As the number of items to process gets higher, the economics become even more convincing and you can't debate the wisdom of processing content like a professional.

From a project perspective, here are some of the key tips to keep the content processing project running smoothly:

- **Plan first.** You learn just how expensive content processing can be if you start changing the processing plan after production has begun.
- **Start early.** After you know the exact format and tagging you're aiming for with a particular content source (somewhere between the end of logical design and the end of system selection), you can initiate the project to convert that content.
- **Don't depend on casual resources.** Anyone who volunteers to process a mound of content for you has probably never done it before. Can she produce decent-quality work? Can she guarantee that she can finish all that she promised? Can she follow your rules? If so, go for it. If not, you may be better off paying for a resource.
- **Keep the flow.** Ensure that, after your processing team is in place, you can keep a steady flow of information going to the team. It's cheaper and better for morale and quality to wait longer but work at a higher flow rate than to start and stop. So make sure that you can ensure a steady flow of raw information before you get the team started.
- **Check the output.** Don't assume that the output from your process as you planned it is exactly right. Instead, do the process in advance on a small sample and test it to make sure that you can move it without problem into the next stage of its life (metatorial processing or right into the CMS repository). Not only does the sample content help you gain confidence in the conversion process, but it also provides good sample content for the people setting up the CMS.

Cross-Reference

For more details, please refer to Chapter 30, "Processing Content."

The Slime Line

I'll never forget working one summer at the salmon canneries on the Aleutian Islands of Alaska with my friend Tim. I learned more there about content processing than from any other job I've had. My job was "gutter" on the Slime Line. Salmon with a long incision on their abdomens would come my way on a conveyor belt. My job was to make sure that no guts were left by the time they passed me.

Like all factories, the cannery ran on a very tight schedule and margin. The biggest lesson the cannery taught me was this: Do your organizing and planning before the fish arrive. After the fish start flowing, the whole process had better go like clockwork or you end up with a lot of dead fish on the floor, a lot of not-busy workers, and one very broke fish company.

Template coding

Templates are a cross between programs and design prototypes. In fact, the way most templates are created is to start with a page prototype and progressively replace the dummy text and other placeholders standing in for dynamically generated content and navigation with their programming equivalents.

Cross-Reference

This section covers the project process behind template coding. You find more information from the analysis side in Chapter 22, "Designing Publications," in the section "Analyzing Templates," and more from the technology side in Chapter 33, "Building Publishing Systems," in the section "Detailing the Components of a Publication System."

The following are two different types of templates that you need to implement:

- **Input templates** are what I've called *Web-based forms* elsewhere. They generally use the same code base and technologies as publication templates but move information in the opposite direction (into the repository, as opposed to out of the repository).
- **Publication templates** produce output pages. They're the kind that I discuss most under the heading of *templates*.

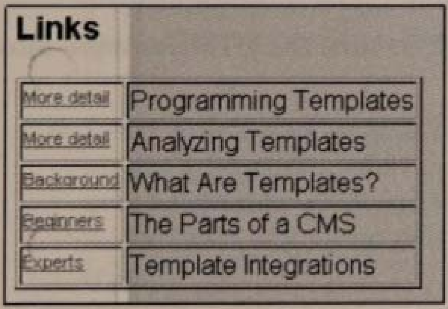
The two kinds of templates are quite similar and depend on the same sorts of processes and skills. I suggest that in your project, however, you separate input and publication template coding. Even if the same people are doing the work, the input templates are part of the collection system and should be developed with the needs of content contributors in mind. The publication templates are part of the publication system and should be developed with the needs of the audiences in mind.

Although each CMS has its own templating techniques and languages, the general process of template development remains the same.

Based on whatever you may have heard about how easy it is to create templates in your system, you may be tempted to give the task to a CMS administrator or even to your design staff. Be careful. If you scratch the surface of even the most user-friendly templating system, you find real programming needs underneath. And although template programming doesn't tend to be the hardest programming task, its scripting and programming-like tasks can get beyond a nonprogrammer quickly. Some systems, for example, use XML syntax to specify content inclusion in templates. These commands don't strictly require a programmer, but they end up providing all the major programming constructs (looping, variable creation, conditional statements), and if you want to do anything nontrivial in your templates, you need all the commands that are provided (and probably more).

From a project perspective, the main issue with templates is mediating between all the people who have some part in creating the final page. By definition, a template mixes design and programming code. You may also have editorial content in a template and user interface design. Sometimes the work of more than one person becomes mixed to such an extent that it's indistinguishable.

Suppose that the design calls for cross-references to be placed in a table, with a small identifier on the left telling what kind of link it is. The page designer creates a page mockup that shows the spacing, fonts, and general properties of the cross-reference table (and undoubtedly does a better job at it than I do), as shown in Figure 17-1.



Links

More detail	Programming Templates
More detail	Analyzing Templates
Background	What Are Templates?
Beginners	The Parts of a CMS
Experts	Template Integrations

Figure 17-1: The design element on the left identifies the ink type, and the text on the right names the link.

Next, a page developer comes along and turns the design into an HTML page prototype that shows the exact code used to produce the design, as follows:

```
<H3>Links</H3>
<TABLE border="1">
  <TR>
    <TD>
         <DIV class="LinkType"><A href="http://link1">More
detail</A></DIV></TD>
    <TD>Programming Templates</TD>
  </TR>
  <TR>
    <TD>
```

```
        <DIV class="LinkType"><A href="http://link2">More
detail</A></DIV> </TD>
    <TD>Analyzing Templates</TD>
  </TR>
  <TR>
    <TD>
        <DIV class="LinkType"><A
href="http://link3">Background</A></DIV>
    </TD>
    <TD>What Are Templates?</TD>
  </TR>
  <TR>
    <TD>
        <DIV class="LinkType"><A
href="http://link4">Beginners</A></DIV>
    </TD>
    <TD>The Parts of a CMS</TD>
  </TR>
  <TR>
    <TD>
      <DIV class="LinkType"><A
href="http://link5">Experts</A></DIV>
    </TD>
    <TD>Template Integrations</TD>
  </TR>
</TABLE>
```

Finally, the template programmer comes through to turn the HTML sample into a piece of working code, as shown here:

```
<H3>Links</H3>
<TABLE border="1">
[FOR EACH LINK]
  <TR>
    <TD>
      <DIV class="LinkType">
        <A href="[INSERT URL]">[INSERT LINK TYPE]</A>
      </DIV>
    </TD>
    <TD>
      [INSERT LINK TITLE]
    </TD>
  </TR>
[NEXT LINK]
</TABLE>
```

For the purpose of illustration, I've used a simplified pseudo-programming language in the preceding example. The lines that are all capital letters and enclosed in square brackets ([ ]) stand for programming commands. The point here isn't to dwell on the mechanics of template coding, but to illustrate how tied together design and programming can become.

What happens, for example, if the page designer changes her mind about the look of the links table? Or what happens if the page developer decides that she needs to change the system names used? Their work product has been totally subsumed by the work of the programmer. They may not even recognize where the table is anymore, let alone know how to modify it. But suppose that they do understand enough to change the programming code. What's to keep them from changing the code, only to have the programmer change it back later by pasting new code over the change?

The often-sad fact of template programming is that a number of people of very different stripes must play nicely together in the same files and even on the same lines of the same files. If you don't want continual blunders and territory wars over your template files, you'd better come up with some workable processes for ownership and changes to templates. Here are some rules to get you started:

- **Modularize code.** The more distinct and chunked that you can make your code, the more you can separate it out of the template file. You may, for example, create a function that builds the cross-reference table and moves it out of the template file entirely, leaving behind only a function call (such as `[BUILD LINKS]`). This not only minimizes the need to share the same files, but also helps produce reusable code segments.
- **One owner at a time.** This maxim seems simple, so why is it so often ignored? If you have a version control system in your CMS, use it to manage work on your templates. If you don't have version control in your CMS, get a source management product. It pays for itself in the rework that you don't need to do.
- **Strict touch and no-touch zones.** Mark each template section. (HTML comments are usually what people use.) Then be clear and consistent about who gets to change what sections and when. A simple spreadsheet with the section names along one axis and your job titles along the other axis can do the trick.
- **Close communication.** Of course, you can find no substitute for a close working relationship among all the people who must live together in the same file. If the page developer actually talks to the template programmer periodically, they avoid the majority of potential collisions.

Template programming is a real project and should be treated that way. You may have dozens or scores of templates to create and other parts of the implementation depending on the completion of certain templates.

Custom application coding

In a large CMS project, you're bound to have at least a few standalone applications to create to augment the stock functions (those built into your CMS). By *standalone,* I don't mean completely independent, but rather separate enough that you'd use a language outside the CMS tool set to create and test the application. Following are a few examples of custom applications:

- **A Java applet** that sits inside a CMS input template (a Web-based form). The applet connects to your organization's product catalog database and enables you to select one or more product IDs to attach as metadata to a content component. From the contributor's perspective, the applet looks like just another control on the CMS form. The programmer knows, however, that the applet is a standalone program that knows how to query the product catalog and pack the user's choices into the data that's passed back by the form to the CMS.
- **A C++ program** that scans the CMS database once a day and automatically archives components whose archive date has arrived. Maybe after trying to get the CMS to do this, the team decided it was much easier to write a small program that's triggered by the operating system scheduler.
- **A Perl script** that turns the HTML stored in the CMS repository into Maker Interchange Format (MIF) for use with the organization's Adobe FrameMaker print publication system.

As you can see from these examples, far from being independent of the CMS, the custom applications tie directly into it. For this reason, it's useful from a project standpoint to pay close attention to the dependencies between the custom applications and the stock CMS functions, as follows:

- **Make sure that the CMS can't do a function** and that you really need that function before custom coding it. I've seen programmers overly anxious to take on functions that the CMS already has. You don't need the added burden of scheduling and supporting programs that offer limited return over what the CMS can do natively.
- **Make sure that you use the front-end CMS interface if possible.** If the CMS supplies an API or other standard automation interface, prefer it to faster and more direct methods. In later releases of the CMS, the front-end interface is likely to remain compatible with your custom code, whereas the sneakier methods you may discover are likely to break. You may want to check with the CMS vendor on any back doors that you find that may disappear in the next version.
- **Treat custom development as a separate project.** Even while paying close attention to the part the CMS plays in a custom application, understand that you're creating a separate application, because the CMS isn't capable of doing everything. Keep the custom application teams small and separate. This enables you to keep them effective and also enables you to engage and disengage them as your budget and schedule allow.

Integration coding

Integration code connects your CMS to other systems. The connection may be input (getting HR data, for example) or output (passing transaction data from a Web site to an e-commerce system, for example). It may consist of a custom application, template programming, or both.

Suppose that you plan to integrate with an inventory system from a Web site that your CMS produces (see Figure 17-2).

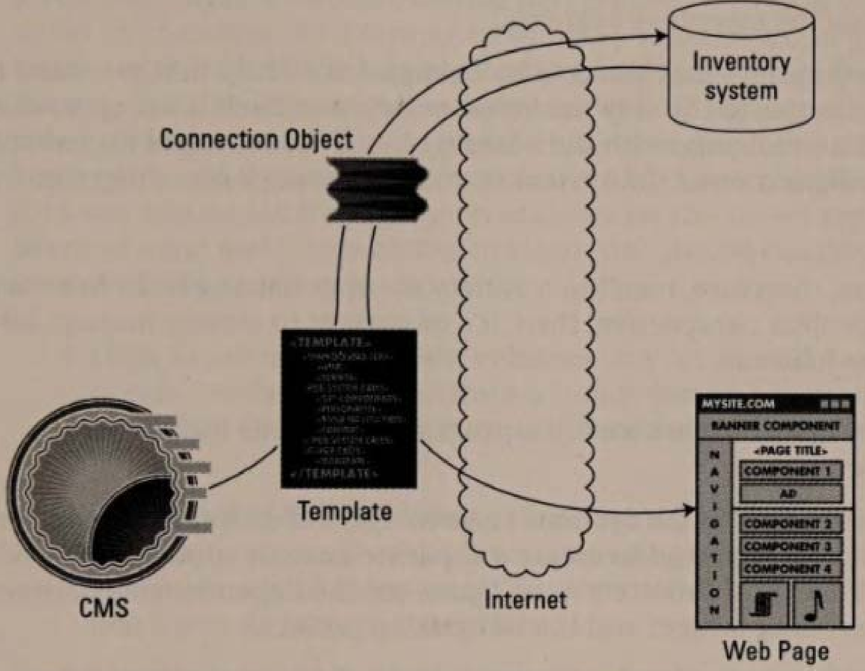

Inventory system
Connection Object
<TEMPLATE>
</TEMPLATE>
Template
CMS
Internet
MYSITE.COM
BANNER COMPONENT
<PAGE TITLE>
COMPONENT 1
AD
COMPONENT 2
COMPONENT 3
COMPONENT 4
NAVIGATION
Web Page

Figure 17-2: To integrate a Web site with a CMS and an inventory system may require programming in a number of different areas.

To start, the template draws from the CMS whatever content and navigation it may include. In addition, the CMS recognizes a call within the template to an external object (called the *connection object* in this illustration). The object connects to the inventory system (over the Internet, in this case) and retrieves data. The object also may put data into the inventory system if, for example, the user orders something and the inventory must be decreased. Finally, the object returns data to the template, which formats it for appear on a page that's sent to the end user's browser.

Note For simplicity, I've excluded other integrations you'd probably need in an e-commerce application to transaction and fulfillment systems.

In this situation, programming may need to happen in three places:

✦ **The custom object must be created.** On one hand, it needs to know how to establish a connection with the inventory system, how to access the inventory data, and how to correctly store data in the inventory system. On the other hand, it needs to know how to package the data that it retrieves from the inventory system for use by the template.

- **Template code must be created.** The template needs to be programmed to include the calls to the custom object and routines for formatting the results. In addition, a user interface may need to be added to the template to give the end user the capability to make choices (such as the choice to buy the items displayed from the inventory system).
- **The inventory system may need to be changed.** It's likely that you need to make some changes to the way the inventory system itself is set up to enable access and data exchange with the custom object. The changes may simply require reconfiguration of the system or some amount of new programming code.

Integration projects, therefore, result in a variety of programming tasks in a variety of places. From a project perspective, then, it's important to closely manage integration projects, as follows:

1. You must clearly determine what the programming areas for an integration are.
2. You must gain access to the systems to be integrated with (not always the easiest task). You may need to create a separate custom application development project, and you definitely must figure out the dependencies between the template coding project and the integration project.
3. You can task the template programmers with hooking up the integrations in their own templates, or you can add the integration programmers as another group of people who share the template files. Both approaches have the following benefits and drawbacks:
    - If you task template programmers with integration tasks, you have fewer problems sharing template files, but you also have two different people working on the same integration.
    - If you task integration programmers with template tasks, you may get a higher quality integration but more trouble working through the sharing issues.

Of course, this isn't as black and white as I may be presenting it. Again, the compromise position comes with good communication and trust among the team members who must share responsibility as well as files.

Tip

Have the implementation team members keep track of all the key problems that they encounter and the solutions they've found. You can use these lists later as the basis for troubleshooting tips for the people who administer the CMS.

Testing the system

I assume that you're likely to be familiar with Web-site testing, software-application testing, or, perhaps, no testing at all. If you have no familiarity with computer testing at all, your effort to read up on it or take a short course is definitely repaid. I don't spend any time at all going through standard testing methods.

Cross-Reference I talk more about quality control in Chapter 30, "Processing Content," in the section "Defining test methods."

If you come from a software testing perspective, your best information is found under the heading of enterprise integration. Assuming that you do all the standard testing on the CMS's custom applications and template code, your main concern is the innumerable dependencies between the CMS and other systems. The most critical of these are on the Web server. Web servers aren't the most robust systems to begin with, and if you add the complexity of dynamic, high-volume sites to them, they can and do fail frequently. Stress tests on the server are thus critical to your peace of mind and future ability to sleep. You should consider testing for at least these following sorts of loads:

- **High browser transaction volumes** on your most dynamic pages. In logical design, you've estimated transaction volumes. I suggest doubling these values in testing. Or, better yet, increase the volume of page requests until the system fails.
- **The effect of live publishing.** Try pushing a lot of new pages and database records to your site while it has an average to high transaction load on it. This simulates a publishing cycle. You want to make sure that no bad things happen if you do large updates later.
- **Simultaneous authoring.** If your live databases have data added to them at the same time they're accessed by end users, make sure that the process doesn't destabilize your system. If you collect and publish news stories on your live site, for example, ensure that the news-story page continues to publish correctly even as you plow in as many news stories as your staff can add simultaneously. Of course, it's rare that you wouldn't impose a workflow step or two between the news-story creation and its publication, but still, make sure that nothing strange happens.

In planning for system tests, the best approach that I've seen is to create a large dependency spreadsheet that lists all the parts of the CMS and supporting systems (Web servers, enterprise data systems, and so on) on both the row and column headers. In the cells of the spreadsheet, type what can go wrong for the row and column that intersect there. Then use this as your first test plan. As you (or your test crew) test each intersection, don't forget to enter the new scenarios that come up.

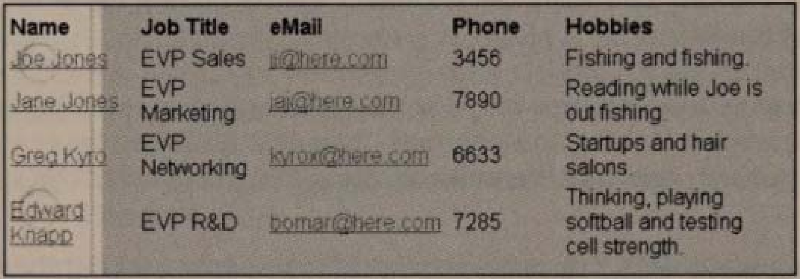
In addition to being a great way to test the system, this is a great way to learn the system and how all its parts interact. Make sure that you pass this spreadsheet on to the people who must document the system and do training.

Publication tests

If you've come from a Web testing background, the training that helps you the least is the standard click-through tests. Many Web sites are tested simply by having someone click every link to see what happens. The whole point of a CMS is to create publications that change constantly. To keep up with the productive capacity of a running CMS requires that you have an army of clickers constantly trying every new page.

You must take a much smarter approach to testing the publications that your CMS produces. You find an analogous issue in software testing. In applications that can produce an unlimited set of output screens, instead of trying to use the application's user interface to test to make sure that all screens come out all right, testers back up to the code itself. They test the screen-producing functions to make sure that, for all scenarios they can think of, the functions produce reasonable screens.

The same methodology can work for testing CMS publications. Suppose that you want to test the employee information page shown in Figure 17-3.



Name	Job Title	eMail	Phone	Hobbies
Joe Jones	EVP Sales	jj@here.com	3456	Fishing and fishing.
Jane Jones	EVP Marketing	jaj@here.com	7890	Reading while Joe is out fishing.
Greg Kyro	EVP Networking	kyrox@here.com	6633	Startups and hair salons.
Edward Knapp	EVP R&D	bomar@here.com	7285	Thinking, playing softball and testing cell strength.

Figure 17-3: To test this page, don't just look at the output.

By looking at the output, you can see only so much. You're better off looking behind the page to see how to test it and what's likely to go wrong, as follows:

- **Rather than clicking the links** associated with names and e-mail addresses, check how they're built. What happens, for example, if an e-mail address is missing? Does the template logic catch this and not add a broken link? Back up one more step. Does the input form or conversion process ensure that an e-mail address is present? If it does, you have no need to worry. If it doesn't, where's the bug? Is it in the input method or in the page code?
- **Rather than checking the actual text on the page**, look behind the page to how the table is built. Are the column widths set? If not, can you think of a way to make the table become too wide for the page? Are text lengths limited for any of the columns, either in the page logic or in the input methods? If not, how may this table build poorly?
- **Rather than assuming that the same components are always there** or are in the same order, imagine many more components or their order changing. Note, for example, that Jane related her hobbies to Joe's. That's fine if they're right next to each other as they are now, but what if many more employees are added and the two components become widely separated? This is a subtle bug (if it's a bug at all), but the concept is quite important. Not only must the tester test for the mechanics of the publication, but she also must test for sense. Does a page make sense in any of the varieties in which it may appear? The page designers must be the first to ask this question, but it's the tester who's in the best position to judge whether they got it right.

As the few examples I gave illustrate, to test a page, you must know and test the whole system. As clear as the problem may become on the completed page, it's not at all clear sometimes where the problem originated and where it should be corrected. The result of a testing bug report may be a change of algorithm in a conversion routine; a change of rules in an authoring process; or a change of template design, template implementation, or template programming.

Usability tests

Usability tests determine how understandable and useful your user interface is. Publication designers often perform professional usability tests on prototypes of publication pages to make sure that audience members can navigate through them and find what they're looking for. Professional usability studies use advanced observation techniques such as eye-movement detectors, click recorders, videotaped use sessions, and directed tasks. You should also try to do some form of usability tests on the custom application interfaces, Web forms, and other interfaces that you create for staff members. You needn't go overboard with these tests. They can be as simple as watching a typical staff member try to do a series of common tasks using your interface. If she gets stuck, you know that you need more work on the interface.

Beta tests

In a beta test, you test an entire system with a small number of real users shortly before releasing the system to the world at large. The usual assumption is that the system is finished except for the bugs that you find during the test. This assumption is almost always violated by staff that end up counting on the beta test time to fix all the known deficiencies that they couldn't get to during the normal course of implementation. For that reason alone, it's good to plan a beta test. It puts a nice cushion between the project completion and release. But the benefits go way beyond the extra time you get. A beta test cycle does the following for you:

- **Gives you an inside user group.** You have the chance to build an unusually loyal set of audience members who are on your side not only in finding problems, but also in suggesting new directions.
- **Gives you a dry run** before you release for real. Any big problems that are going to come up in your system ought to appear in your test. Of course, a lot of the biggest problems have to do with performance at large volumes of users, and these don't show up during a beta test unless you simulate them yourself by using stress-testing tools.
- **Gives you a deadline that's serious** but not deadly. People work hard toward a beta release with much less anxiety than they do toward a looming release date.

If you can afford the extra time for a beta test that includes both internal users and external audiences, it's worth your while.

Cross-Reference

You can find more information on the beta system in Chapter 18, "Rolling Out the System," in the section "The beta system."

Taking Stock of the Deliverables

For consistency and simplicity, I break the implementation deliverables into collection, management, and publishing-system specifications. You can choose these three major divisions or break your deliverables into a larger set of specifications. Rather than give you a blow-by-blow account of creating implementation deliverables, I offer the following two broad angles of attack on your implementation deliverables:

- **You can use your logical design** to decide what the system must accomplish and to provide all the parameters and lists of particulars that you need to specify the system. Logical design is described from an analysis perspective in Part IV, "The Logical Design of a CMS."
- **You can use the selection criteria categories** to define the main areas of implementation that must occur. The selection criteria categories are described in Chapter 16, "Selecting Hardware and Software," in the section "Sorting Through the CMS Selection Criteria."

> Consider, for example, this sample overview from the content-deployment section of a publishing specification: Our logical design calls for our publishing system to deploy content to our Web servers in North America, South America, and Europe. The same base content must be replicated at each site. In addition, locale-specific content must be deployed only to the sites for which it's marked. All components have a Locale element, with values of NA, SA, EU, or ALL. Components are also all tagged with Type and Audience elements. The table in section 2.4.3 shows the values for those elements and how they're used to calculate a destination directory for each component file that's deployed.

Table 17-1 shows the major functions of deployment, what software supplies those functions, and where in this document they're fully specified.

Table 17-1
Deployment Functions

Function	*Source*	*More Information*
Deploying content files	CMS Distribute function accessed through the admin interface	CMS Admin Guide p. 96 and in section 3.4.5 below.
Deploying database records	CMS Distribute function augmented with custom connection object	CMS Admin Guide p. 112 and in section 3.4.6 below.
Deploying appropriate content to each locale	OS Scheduler set to run deployment script periodically for each locale. Passes appropriate Filter and Destination parameters to the CMS Distribute function	OS Reference Guide Vol. 1 p 352, CMS Admin Guide p. 115, and in section 3.4.8 below
Deploying to a staging server for testing	Destination parameter in the CMS Distribute function	CMS Admin Guide p. 101 and in section 3.4.4 below
Directory targeting by content type	Destination parameter in the CMS Distribute function. Use CMS event script to access component elements Type and Audience	CMS Admin Guide p. 101 and in section 3.4.3 below
Deploying files referenced in a component element	Custom script that scans files as they're deployed.	Section 3.4.7 below
Expiration dates	Passed as part of the Filter parameter of the CMS Distribute function	CMS Admin Guide p. 115 and in section 3.4.8 below
Scheduled automatic deployment	OS Scheduler set to fire periodically for each locale	OS Reference Guide Vol. 1 p 352 and in section 3.4.8 below

Notice the following aspects of this sample:

- **It combines the logical design with the physical design.** The logical design is cited in the introduction and is elaborated in other parts of the document (section 2.4.3, in this example). The physical design criteria for deployment are listed in the table. Called *functions* here, they're simply rephrasings of the deployment system selection questions that I pose in Chapter 16, "Selecting Hardware and Software," in the section "Content deployment."
- **It points to CMS functionality.** Where the CMS is capable of performing a function entirely, it cites the function and states where that function is documented.

- **It points to custom functionality.** Where the CMS functionality must be augmented by custom programming, it points that out and states where the CMS functionality is documented, as well as where the custom functionality is specified.
- **It points to other software.** You can assume that the OS Scheduler, for example, is neither part of the CMS nor a function that you code yourself.

Whether you use a method similar to this one or make one up on your own is less important than that you make full use of your logical and physical analyses and decide what functions are stock, custom, and third-party.

Of course, it's not enough simply to state what you must do. Somewhere, you must write down exactly how you plan to do it. You must provide enough detail that someone who's not gone through the logical analysis or system selection can know just what to do to implement the system.

Collection specifications

You can create a collection specification that does the following:

- **Fully details your assumptions for component sources** (authoring or acquiring) and how each component loads into the CMS repository.
- **Details ownership and workflow steps** for each component to be created and how the jobs and workflows are created in the CMS.
- **Lists standard collection tools** and how they should be configured.
- **Fully specifies any custom tool** or process development that must be done.
- **Includes any content conversion specifications** that are required. (See Chapter 30, "Preparing Content," in the section "Managing the Processing Project," for more information.)
- **Details the parameters of any integration** that you may need to connect the collection system to the enterprise.

The collection specification can draw directly from your logical analyses. Your author and acquisition source analyses, of course, figure most prominently in a collection specification. Use the following list to begin to transform your logical design into collection specifications.

- **Author analysis:** Most of the information that you collect on authors goes into the specification of the authoring system and the tools that you provide each group. I favor creating sections for each author group that note the information that you find out about each group and how you can set up their tools and processes. Among the major points that you want to cover is the configuration of input templates. Depending on how much metadata you think each author group can enter, you can show more or fewer elements on the input form. You also want to describe the primary tools of each group and how they're configured to interface most cleanly with the CMS. Finally, make sure that you work through from the description of any custom authoring tools to their complete specification.

- **Acquisition analysis:** The information that you collect in an acquisition analysis contributes mightily to any content processing specs that you create. The source structure data that you collect is the starting point for a conversion, and the component and element analyses show the target of the process. The quantities show the processing marks to hit. The only thing left to specify is how you can use your limited resources to do all the work that you've so aptly specified! The source analysis can either be brought into the collection specification or just referenced. It's probably best to recapitulate the source structure analysis, for example, but only reference the source agreements.
- **Workflow analysis:** Your workflow modeling should be all the input that you need to decide what tasks, triggers, and sequences to set up in your chosen CMS. Of course, if your chosen CMS can't do the sorts of tasks, triggers, and sequences that you propose, you must specify how you intend to augment the CMS (or cut back on your workflow needs).
- **Component analysis:** Finally, your content structure analysis ought to give you an additional set of constraints to put on the collection system. How is each component tagged to fit into the structures that you define? You have plenty of room here for innovation and labor-saving aids. You can provide lists and other controls on any input form, for example, to put a point-and-click interface on the selection of keywords, cross references, and, with a little work, hierarchies. You also can write code that makes educated guesses about expected values for these controls.

Among the categories of selection criteria (described in Chapter 16, "Selecting Hardware and Software," in the section "Sorting Through the CMS Selection Criteria") and collection criteria, of course, are the ones applicable to the collection system. You can use these criteria as a starting point to decide how each major collection function can be accomplished. As you may suspect, I favor organizing the collection specification overall under the collection headings of authoring, acquiring, converting, and aggregating (as I lay them out in Chapter 7, "Introducing the Major Parts of a CMS," in the section "The Collection System." But you may find a different organization better, given the way that your chosen CMS is organized and presents its user interface.

In addition to the collection criteria, pay attention to the following criteria categories as well:

- **Management administration:** Make sure that you include a specification of how you can use the CMS and other tools to configure and manage the collection system over time.

- **Workflow:** The criteria here ought to help you gauge the workflow possibilities of a system and choose the ones that are best for the workflows that you laid out in logical design.
- **Version control:** The criteria here guides you through what sort of version control is possible and what level you need for your authoring processes.
- **Reporting:** You need to decide what sorts of reporting your collection system requires and how to accomplish it.
- **Security:** If access to controlled areas of content is an issue, you want to devise a security policy for authors and other staff who can access the content in the CMS repository.
- **Personalization:** Personalization happens in the publishing system, but it's made possible by the tagging that's accomplished in the collection system. For the personalization that you intend to do, you need to specify how each of the tags that support it is determined and entered in the collection system.

Management specifications

A *management specification* can detail the way that the chosen system must be set up and configured. It can also include the complete content model, fully describe each workflow that you create, and include any integration specifications that you need to connect the management system to other systems.

On the logical analysis side, the core of the management specification is the component analysis. Somewhere in the CMS, you need to enter your component classes, elements, and element values. In some products, you can do this completely within a user interface. In others, it's accomplished in part by using database management tools. In either case, you need to convert whatever general modeling that you did in logical analysis to a specific set of actions in the CMS and its repository software. In addition, because the management system is (quite literally) central to the CMS, every other part of the logical design plays some part in your management specification, as the following points indicate:

- **Your audience analysis** yields audience profiles that must be entered somewhere in the system. Either in the repository or in a separate database, you probably establish a user-profile table, where you can collect and track data on your audience members.
- **Your template analysis** yields a set of template files that must be managed. Either by using your CMS version control or other software, you must figure out who needs to share these templates and how you can facilitate it.
- **Authors and other staff members** must be created as user groups in the CMS system. You already know what each group needs to do. Knowing the capabilities of your chosen CMS and the extensions that you plan to create, you can specify exactly how groups are to be set up and what access each can have.

- **Acquisition sources** often have a batch interface to the CMS repository. Maybe you configure your CMS to scan for new material on a partner Web site once a day, for example, and load it as a batch into components of a particular variety. You're likely to have a lot of configuration and custom programming behind these types of sources.
- **Management workflows** must be configured and added to the CMS workflow system.
- **Content access structures** must be implemented in the CMS repository. The structures you want may be easy to set up through the CMS user interface, or you may have a lot of custom coding to do to get your structures (especially multiple hierarchies) to work and integrate with the rest of the CMS.

It should come as no surprise that the management selection criteria (as described in Chapter 16, "Selecting Hardware and Software," in the section "Sorting Through the CMS Selection Criteria") have the most influence on the management specifications. And again, because of the centrality of the management system, all other categories of criteria play some role in the management specs. A few, however, deserve special mention, as follows:

- **The collection criteria**, as a whole, impose requirements on the metadata creation and administration facilities of the management system. Make sure that you specify which functions of the CMS can and can't be used to create and update the metadata that your collection system says to store in the CMS repository.
- **The deployment criteria** are as well cast as management constraints as they are as publication constraints. After all, it's the management system that contains the functionality for deployment.
- **The Web architecture criteria** are management constraints to the extent that the CMS runs on a Web server. Many systems are mostly located on the same server that's used for serving the Web publications. In fact, you can rightly say that the CMS interface is just another Web publication produced by the CMS!

Publication specifications

Publication specifications state exactly how each page or section of each publication is constructed. They augment the concepts and drawings done during logical design with detailed specifications of the graphic, style, and layout elements to use. In addition, a publication specification specifies the logic and coding behind the templating model for each publication. The publication specification details the queries, content manipulation, and personalization processing that's needed to structure and format each publication. Finally, the specification details any integration that must be done to connect to other systems.

In general, I recommend a separate specification for each publication that gathers all the information for that publication in one place. Sections that are common to more than one publication can be broken into a separate global publishing resources specification.

Your publication logical design has three parts (publication, template, and personalization analysis) that contribute in related ways to the publication specifications, as follows:

- **Publication design:** Little of the publication analysis (audience, author, purpose, and the rest) shows up directly as system customization or configuration. It mostly results in publication guidelines and specifications. Indirectly, however, the analysis is reflected in the audience, author, and personalization specifications and the customizations and configurations to which they lead.
- **Templating design:** How much of the templating process happens as configuration rather than as programming varies from product to product. Usually, not much of the templating process is done through user interface. One exception consists of products that have a visual template designer that enables you to drag content components and navigation elements onto a screen. Regardless of how the templating can be accomplished in the chosen system, the templating analysis gives you everything that you need to say what templates are needed and how they should be constructed.

  The only thing missing from the templating analysis to make it a specification is the technical detail of how logic is implemented. Given that you now know the templating language that you're going to use, you can state the number and kinds of routines and other programming code that you need to implement the template logic that you describe during logical design.

Tip

Make sure that the specification contains a solid plan for code sharing between templates. The worst thing that you can do is use the same code in a variety of templates that all change independently of the rest. Instead, try to modularize template code, move it out of specific templates, and then call it back from some central location. You can even include template code as a kind of component that's automatically added to a template as it's published.

- **Personalization design:** Commercial CMS products vary from no personalization system to very extensive systems. Assuming that you augment any deficits in your product with third-party personalization systems, you have a lot of configuration to do. The audience profile information somehow must be provided to the personalization system. The system probably also automatically accesses the component configuration that you create. Additionally, you may need to configure it to look at other data sources. Finally, and most important, somewhere you must enter your personalization rules. If you're lucky, a slick point-and-click interface makes rule building easy. More likely, however, rules are entered in text or XML configuration files, and you must augment the system that you use with custom code to get the personalization results that you want.

Many of the other parts of logical design also inform the publication specs, as the following points indicate:

- **Audience analysis** is, of course, a part of the personalization process.
- **Component analysis** gives you what you need to plan how each component finds its way onto the correct publication page in the appropriate format. For each component/publication combination, you can state on which pages the component appears. For each page where the component appears, you can specify which elements are used for accessing the component (and what values they must have), the elements that are displayed, and the number of components of that type that also may appear on the page.
- **Author analysis** provides the right data for deciding how to attribute and reference authors in your publications.
- **Publication workflows** must be incorporated into the CMS workflow module or implemented by using third-party tools and customizations.
- **The content access structures** you've devised must be transformed. Your publication specs need to describe how the general access structures in the repository can be turned into the very specific content-access structures that you have in your publications.

As before, the selection criteria that you used to decide upon a CMS product can also be used to guide your publication specifications. Pay particular attention to the criteria in the following categories:

- **Management administration:** To whatever extent that the management administrative interface can help you set up and lay out publications (usually not a lot), you should include a discussion of it in your publication specs. If it can't help you very much, make sure that you describe what else you must do to configure your publications.
- **Management integrations:** Some of these integrations may need to happen in a Web publication as well as in the management system itself. You may, for example, need to connect to a live product catalog in your Web publication. Make sure that you state how these integrations occur.
- **Version control:** Do you need to version and keep track of finished publications? If so, the versioning system of the CMS or alternative tools need to enter into your publication specs.
- **Security:** The CMS security system may or may not extend out to the Web publications that you create. If it does, how does it do so? If it doesn't, and you need security on the Web publication, what else can you do?
- **Performance:** To the extent that the CMS serves dynamic pages for you on a Web site, you need to include information about its performance metrics in your publication specifications.

The project plan

After you've created the specifications for the CMS, you're finally in a position to create a definitive project plan. Your hope is that the changes that the specs introduce to the plan are in the detail, not in the major headings. You may, for example, have a line that reads, “`Template coding..... 3 weeks.`” After creating a publication specification, the single line may turn into 100 lines for the tasks associated with each template and the templating system as a whole. As you expand the project plans, make sure that you work with the entire team. This gives you the best set of tasks and task estimates and also helps you get the full team's commitment to work to the plan. I've seen mixed results if the team itself is responsible for creating the plan detail. Sometimes you get unreasonably long estimates from people who don't know and so guess very high, and sometimes you get unreasonably low estimates from people who have optimistic attitudes. The best method that I've seen is to have a scheduler work together with the staff. That way, staff estimates can be leveled out by the scheduler, and the scheduler can negotiate the support of the staff for the plan.

> **Note**
>
> I've often seen scaling factors applied to the time estimates made by staff. If a person characteristically underestimates by half the amount of time it takes to accomplish a task, for example, you can simply multiply all her estimates by 2. The multiplication is easy; knowing the factor by which to multiply is difficult. It requires that you really know and understand the personalities on your team. Of course, by the end of implementation, you should have your scaling factors very well estimated.

The remainder of implementation is a time for doing, not thinking up project plans. It's also the time when some projects abandon their project plans because they're too difficult to maintain in the face of a rapidly evolving project. Abandoned project plans are a sign of the following:

- **A changing mandate:** If outside forces continue to change the agenda for the project during implementation, no amount of project planning can save you. If you have the sort of mandate that I've suggested, this ought not to be a problem. If you don't have such a mandate, or if you still get pressure to change plans while in implementation, your project plan can be your best ally. The plan is where the effect of any change can be quantified and debated. If you have a strong project plan, use it to assess and communicate the results of the changes that are being suggested. Become accustomed to trading scope changes for schedule and budget changes.
- **Poor planning:** Abandoned project plans are often the ones that weren't very good to begin with. If you never quite figured out the project before beginning, the project begins to take its own direction as you get into implementation, which may be quite different from the one sketched out in the project plan. If, on the other hand, the project was completely thought out, it may change, but each change is an increment from the existing plan. And each increment can be tracked and merged into the plan to keep it relevant and definitive.

- **Poor training or no time:** Project plans are a lot of work to maintain. Every task and person is interrelated in the plan in a giant web. If you change one part, it's likely to have a wide effect throughout the rest of the plan. If the person who's in charge of the plan isn't really comfortable with the project management tool that you're using, or if she doesn't have enough time to keep up with the changes, the plan is bound to get out of synch in the heat of the project. If the same person's been in charge of the plan from the start, she should have had ample time to really learn it by now. If time is the issue, you need to shift responsibilities to make sure that you have room to continually mend the net that holds the project together.

Given that you can put the effort into a well-maintained project plan, here are some ideas for how to make it pay back this effort:

- **Inertia:** The project plan is a big, bulky, complicated mass that's difficult to change. It's just the sort of thing that you need to keep a big project on course. Given the amount of time and energy that you've put into the planning of the system, and the fact that you're no longer planning, it's appropriate for changes to the plan to be difficult to make. The project plan is like the massive keel of a sailboat. It's below the waterline, keeping the boat on a straight course and making sure that it doesn't tip over.
- **Effects:** Use the project plan to gauge the effect of a change on the implementation. If a change is suggested, work it out in the plan first and prepare an assessment of what it took to fit it in. If it takes you all day just to adjust a copy of the plan to accommodate the change, you know that this change may be major. The difference between the original plan and the changed plan tells you exactly what the change means.
- **Scale-backs:** Given that even the most liberal estimates of time often aren't enough, the project plan gives you a way to manage the continual growth of the project before it explodes. A 10 percent schedule overrun in the first week may compound to a 15 percent overrun in the second week, 20 percent in the third week, and so on. As you see overruns occur in your plan, run them forward, trying to judge how they may compound. Cut first the tasks that have the most ripples throughout the rest of the project.

A project plan is a model of how the project ought to behave. If you put enough time and intelligence into planning, it's a reasonable model. But it's not an accurate model. Just too many contingencies arise after a lot of people start working for you ever to account for them all ahead of time. Instead, you must be willing to continually test and revise your model using what really happens. If you accept this responsibility, your plan can be your biggest aid. Your plan is a place to record the effect of what you didn't account for in the past and to predict its effect on the future of the project. Only the parts of your plan that are in the past are 100 percent accurate. The parts that are to come, however, ought to be more and more accurate as you continually revise them with the lessons that you're learning.

In addition to being your project model, a plan is also a project contract. You can use it to form agreements with the staff about what happens and how long it takes. Your plan is a grand negotiation of all the factors that make and break the project. The biggest part of that negotiation is with the staff of the project. And it truly must be a negotiation for it to work. If you hand out schedules at the beginning of each week and say, "This is what you must do for me by Friday," you get back on Friday all the reasons why you were wrong in your estimates. If, instead, you say, "Here's the plan; tell me how to make it work," you get active participation and a deeper commitment to make it work. On Friday, you still may get excuses, but at least they're phrased in terms of tangible changes to the plan rather than in terms of all the reasons that you, not they, are responsible.

The key to staying in the general vicinity of your plan is the commitment and a shared sense of responsibility that everyone on the project feels for the plan.

The risk assessment plan

During implementation is the time when most of the risks that you've predicted either come true or go away. You can use your risk assessment to plan contingencies in advance of beginning the risky activity. If you predicted that the CMS may not be capable of communicating with your catalog system, for example, be prepared for it not to work. Assess your options and start the planning enough in advance to be able to change directions quickly. In advance, get pricing and training costs on different catalog systems, for example, and estimates from custom development groups. Then, if the communication risk comes true, you can change directions quickly.

Of course, just because you're in implementation doesn't mean that no new risks occur. Quite the contrary. Every day you hear of new risks from your staff. You hear that you may need twice the number of templates or that an integration may take twice the time that you'd anticipated.

Your risk assessment and project plan can help you defuse the tension and panic that often surrounds the gloomy feelings of a staff person who's had a bad day. To mitigate the risk, and the feelings that surround the risk, you can do the following:

- **State it:** Have the staff member clearly articulate the risk. What exactly are you afraid may happen?
- **Find the contingencies:** Have the staff member state clearly the conditions under which the risk may come true. Have her be as precise as possible and lead her through a thorough examination of the situation.
- **Find the probability:** Looking at all the contingencies, how likely is it that the risk may come true?
- **Find the effect:** If it does come true, how big a problem is it for the project?

- **Find the mitigations:** With the project plan in front of you, work through a variety of scenarios with the staff member. Go wide at first, to get at all the possible ways to handle the risk, and then work through the most promising ones in the plan.

My experience is that, with the vast majority of risks presented to you, you never need to go through this complete process. Each step filters out problems that are less project risk and more staff anxiety. The first step especially is the best way to sort anxiety from fact. Simply stating the problem as clearly as possible often is all that's necessary for the staff person herself to solve it immediately.

Taking Stock of Your Staffing Needs

The whole range of CMS job types could be involved in a CMS implementation. Rather than going through the process person-by-person (a subject well covered in Chapter 11, "Staffing a CMS"), I focus here on the leadership and communication structure only.

The main concerns that I have about CMS implementation are keeping the process focused, keeping the scope steady or shrinking (not growing), and ensuring that all the dependencies between processes and teams are resolved to the timely completion of the implementation project as a whole.

With this in mind, I'd design the staffing as follows:

- **A single overall leader** taken from the ranks of the leading analyst or content manager. This person is in place for final escalation of project issues and to keep the various teams cooperating nicely.
- **An overall project manager** to manage the scope, schedule, and budget of the overall project. Because so many contingencies and dependencies are possible between CMS teams and between the CMS effort and other groups in the organization, this is a very important job.
- **Individual project managers** for each part of the implementation that you designate as a separate project. These managers control the scope, schedule, and budget of a part of the project *in the context of the entire project.* Their goal is to make sure that their project is at the milestone where it needs to be to interact with other projects and that, at all other times, their project needs no attention other than that of their own project team.
- **Technical leads** to make the projects happen. They should be free to create an informal communication network for discussing project issues and for looking actively for appropriate responses. If this group functions as well as it can, it detects problems long before they cause slowdowns or bottlenecks. The trick is to give enough rope to the technical leads (who may be involved in any kind of implementation) that they can freely communicate and explore, but not enough that they can constitute a separate management or decision-making structure.

Summary

Most people think of the implementation of a software system as the part with the most action. As I've laid it out, a CMS implementation ought to be as uneventful as possible. All the major technical and organizational hot points should be worked out (or at least been anticipated and planned for) before the first line of code or setup program runs. During implementation, you do the following:

- Decide how to segment your project into multiple parallel projects that can interact within a larger project framework.
- Create a set of specifications that guide the actions of implementation staff.
- Install, configure, and code the CMS.
- Test the CMS, its publications, and its integrations.
- Decide how far to scale back, given the contingencies that arise.
- Deal with new risks and changes to the project plan as they arise.

In the next chapter, I discuss the final stage of a CMS project: rolling it out to the organization.

✦ ✦ ✦

CHAPTER 18

Rolling Out the System

◆ ◆ ◆ ◆

In This Chapter

A definition and overview of deployment

Techniques for deployment, including how to power up your system

Deployment deliverables, including documentation, training, and maintenance plans

◆ ◆ ◆ ◆

A CMS can become a very large development project. The work that you do to plan and organize the system is the only thing keeping the CMS from becoming an even larger administration and maintenance project. To keep the deployment process on track, you can create a set of guides and plans that state exactly what's to occur and when.

In this chapter, I provide an overview of those guides and plans to give you a feeling for the kinds of issues that you're likely to confront in rolling out your CMS and how you may handle them.

As in the case of implementation, I can't (and really shouldn't) give you much specific advice about deployment without knowing your particular situation, so I intend to stick to those points that apply regardless of the particular CMS that you're using.

What Is Deployment?

System deployment is making the CMS that you develop run well in the context of your organization, your existing infrastructure, and any new CMS infrastructure that you create. After you roll out the system, you finalize the CMS integration with enterprise systems and begin to integrate the system with the people in your organization.

At the risk of taking this analogy of programming too far, you can think of your staff as having an existing set of methods (skills) and functions (job duties). To make the CMS run, you must tie into these methods and functions as well as possible and extend them as necessary. People aren't enterprise systems, however, so if you want to make the integration a success, you must account not only for the intellectual effect the new system has, but for its emotional effect as well.

CHAPTER 18

Rolling Out the System

◆ ◆ ◆ ◆

In This Chapter

A definition and overview of deployment

Techniques for deployment, including how to power up your system

Deployment deliverables, including documentation, training, and maintenance plans

◆ ◆ ◆ ◆

A CMS can become a very large development project. The work that you do to plan and organize the system is the only thing keeping the CMS from becoming an even larger administration and maintenance project. To keep the deployment process on track, you can create a set of guides and plans that state exactly what's to occur and when.

In this chapter, I provide an overview of those guides and plans to give you a feeling for the kinds of issues that you're likely to confront in rolling out your CMS and how you may handle them.

As in the case of implementation, I can't (and really shouldn't) give you much specific advice about deployment without knowing your particular situation, so I intend to stick to those points that apply regardless of the particular CMS that you're using.

What Is Deployment?

System deployment is making the CMS that you develop run well in the context of your organization, your existing infrastructure, and any new CMS infrastructure that you create. After you roll out the system, you finalize the CMS integration with enterprise systems and begin to integrate the system with the people in your organization.

At the risk of taking this analogy of programming too far, you can think of your staff as having an existing set of methods (skills) and functions (job duties). To make the CMS run, you must tie into these methods and functions as well as possible and extend them as necessary. People aren't enterprise systems, however, so if you want to make the integration a success, you must account not only for the intellectual effect the new system has, but for its emotional effect as well.

Integrating the CMS with the organization involves the following activities:

- **Fully planning** the deployment before performing it.
- **Creating documentation** for the system.
- **Initial and continuing training** of CMS staff and contributors.
- **Powering up the system and testing it** in a realistic setting before turning off the existing collection and publication systems.
- **Ongoing content preparation** to move new content into the system.
- **Bug fixes** for the software that you create and updates to software that you purchase (or build).
- **Revising the CMS configuration** to meet new or changed needs.
- **Revising the publishing templates** to meet new or changed needs.
- **Adding new component classes and elements** to meet new or changed needs.
- **Adding new author types or new acquisition sources** and the work on the component classes and elements that this step requires.

Techniques to Get the Job Done

In addition to the myriad techniques that you need to devise in your particular CMS, I offer a few that can help you, regardless of the system that you use.

Creating documentation

Most people's least favorite task is also one of the most important things that you can do to ensure that you use your system correctly and that it's as easy to support as possible. If you follow at all the project process that I propose in this book, you have more than enough source material to create exactly the right type of documentation.

To my mind, the primary purpose of the documentation that you create is to specify precisely the contract between you and the people who interact with the CMS. The "how-to" aspect of documentation is important, but people get that information from a variety of sources and can figure it out if necessary. As a contract, your documentation needs to outline the assumptions on which you build the system and the specific rules that you need people to follow to keep content flowing from creation to publication to eventual retirement. In return, the documentation must clearly spell out the benefits that accrue to the individual and the organization if they do things in the way that you prescribe. Just because it's a contract doesn't mean that the documentation needs to seem dry or overly formal. The essence is that it codifies an agreement between the system and its users.

The key items that I see as part of the CMS documentation set are the following:

- A CMS overview for all staff to read.
- A full set of the product company's documentation for the CMS that you buy (or the equivalent, if you build one yourself).
- A set of authoring guides.
- A set of acquisition guides.
- A metatorial guide.
- An editorial guide.
- An administrator's guide.
- A template guide.

You may also want other written information available, including all the documents that you produce during the project; organizational and departmental policy regarding content creation and publication; content management background information (including this book); Web links; and any sort of glossary that you may produce.

Tip

Producing a miniature CMS implementation to keep track of and publish all your documentation is a great training exercise for some of your staff.

You need to move all the information that you want to continue on after the release of the CMS from the project documents that you create to the documentation set. The CMS project team should probably own the task of putting the documentation together, but it needs to quickly transition the review and maintenance of the documents to those who are to continue to maintain the CMS.

The CMS documentation set needs updating over the life of the first version of the system. For this reason, make sure that people can suggest revisions as easily as possible. If you can manage to include messaging functionality (for example, e-mail links) right in the electronic form of your CMS documentation, doing so really helps you capture and process comments.

The CMS documentation serves as a starting point for work on the next version of the system. In this process, new information moves back from the documentation to the CMS specifications, where you augment it with new functionality. Again, you can aid this process immensely by adding messaging functionality to the electronic version of the documentation. If, in addition to revision suggestions, people can submit extension or new functionality comments right at the place in the documentation where they're reading, you get a great leg-up on the next version. In any case, using a numbered section style so that people can accurately refer to the section that they believe needs changing may prove well worth your while.

The CMS overview

Producing a short overview of the system that each staff member can read to understand what you create and what that person's part is in it is a very good idea. The point of the document is to provide a single official perspective on what the CMS is and does.

Tip

In addition, you can use the document to help "brainwash" new recruits into the CMS way of thinking. I'm only half kidding: The CMS mindset is different enough for some to require a bit of education and, possibly, convincing to instill in people who are new to it the idea that it's a good thing to do.

Consider including the following information in your overview:

- **The final project statement:** That statement comes from the mandate process. (See Chapter 14, "Securing a Project Mandate," for more information.) This statement establishes the same context for everyone who joins the project as it has for those you've carried forward from the start.
- **The final requirements:** These are the requirements that are ratified prior to logical design, along with any changes that you've made over the course of the project. (See Chapter 15, "Doing Requirements and Logical Design," for more information.) You set up the CMS staff to succeed, but they alone ultimately fulfill or miss these requirements.
- **A high-level diagram of the CMS:** Produce a diagram with enough detail that any staff member can see his or her place in the team. Stop before the diagram can no longer fit on a page and no one can understand it after 5 or 10 minutes of trying. This diagram is what you want each staff member to visualize whenever she thinks of the CMS.
- **A high-level chart of team structure and responsibilities:** This chart answers the question, "Who is everybody and what does everyone do?" To make the chart a bit more durable, you may avoid people's names and instead list job titles. On the other hand, including and maintaining names is a good reason for people to return to this document over and over again.
- **An overview of components:** Add just enough detail that the staff can answer the question, "What kind of content are we managing?"
- **An overview of the publications:** Make sure that you make the case of why, without the CMS, this set of publications isn't possible to produce as well or as efficiently. (If this statement isn't true, you've done a lot of work for nothing.)

This document can serve not only to orient staff members, but also as the general overview that you send out whenever anyone asks what you're doing.

Product documentation

You'd think that a CMS company would offer automatically produced, totally personalizable, and perfectly componentized documentation that you can publish in any form that you want. In fact, you may think that it would just load all the documentation into the repository that it ships so that you can publish it as you see fit. Unfortunately, I don't usually find this approach the one that such companies take. Most of the product documentation that I see is sparse, incomplete, unchunked, and in one format only. This situation reflects the fact that CMS product companies, by and large, follow a software-development model of documentation rather than a content management model of value creation and distribution.

Gathering product documentation and disseminating it as well as you can, therefore, is essential. At the very least, you need it all in one place and you need to task someone with keeping track of it and making sure that the latest versions are available.

Product documentation is necessarily generic. You're likely to find the parts that apply to your system packed away deep inside a lot of irrelevant information. If possible, you can produce an index to the product documentation, listing the sections that apply to you and describing how they apply. This approach may not prove as much work as you think if you've already produced a specification such as the one that I suggest in Chapter 17, "Implementing the System." In any case, cross-referencing product documentation in the documents that you produce is worthwhile.

Authoring guides

Each type of author that you define needs to receive a short, task-oriented guide to the sort of authoring that he's to do. The point of the guide is to educate authors of a particular type on the assumptions that you make about them, the rules they're to follow, and what results you expect.

Tip

> You may find that, as you write the guides, your assumptions about your authors sharpen and the groups that you create for them change. Don't fight it. Reorganize your approach now or fight against reality for the life of the system.

The guide should draw heavily from your logical and physical design and include the following items:

- **The author profile:** This profile comes from your logical design (as I discuss in Chapter 24, "Accounting for Authors"), which defines the assumptions about system savvy, closeness to the CMS group, and any other information from that analysis that can help authors of this type understand how they fit within the context of CMS. Include information as specific as possible about the production rates and target audiences for these authors.
- **Attribution guidelines:** These guidelines describe how you intend to cite authors in the finished publications.

- **A tools guide:** This guide describes the tools that you expect the authors to use and how to them set up. In this electronic world, linking the guide directly to a setup and configuration program for the tools shouldn't prove a problem. Such a guide not only makes distributing tools more convenient, but it also makes the documentation a must-have for authors who are just starting out.
- **Workflow descriptions:** You need such descriptions for any workflows in which these authors are to participate. Make sure that you fully describe the conditions that authors must meet for their work to move to the next stage and what they can expect to happen before they see their work come back to them again.
- **Editorial and metatorial guidelines:** If you can manage the personalization issues, you can directly include the parts of the editorial and metatorial guides that apply to this author group. If you can't, you at least need to provide the authors with links or an index to the parts of those documents that apply to them.
- **A guide to the use of their work:** Describe and illustrate what happens to authors' work after they finish it. Include the stages of additional processing and tagging; later reviews, where you show it in each publication (the pages, other content that appears with it, and the surrounding navigational and branding elements); and how you finally retire it. Emphasize that you may add new processes and publications. Your goal with this information is to provide enough context that the authors feels oriented to the later use of their work but not so much that the author expects that context always to remain the same.

Acquisition guides

For each major acquisition source, you want to produce a guide to its processing and tagging. The audience for the guide is all members (and managers) of a processing team who turn to it to learn or review the requirements of the source transformation and their jobs.

All the information for these guides can come from your logical analysis and processing specifications. These guides can include the following items:

- **The source agreement**, which specifies how much of the source to expect, what form to expect, how to expect delivery, and who to contact if you experience any problems with the source.
- **Workflows** that include this source.
- **Processing instructions** that state the exact transformations that must occur and give step-by-step instructions for performing them.
- **Processing tools**, which you can reference or link to in the documentation. You can also provide links to the setup programs of individual tools.

- **Metatorial requirements** that include, link to, or reference (in order of my preference) the parts of the metatorial guide that apply to this source.
- **Editorial requirements** that include, link to, or reference (in order of my preference) the parts of the editorial guide that apply to this source.
- **Samples of the input and output** of the transformation that serve as the official standards to which to compare all other input and output.
- **Processing team structure**, including job descriptions, reporting structure, and escalation procedures.
- **Expected quality and throughput metrics**, which, if you make this document *the* place for your assumptions about quality and quantity, ensures its centrality to the processing team.

After you write (or maybe even conceive) these guides, you want to turn them over to an owner on the processing team for updates and redistribution as necessary.

The metatorial guide

A *metatorial guide* details the ongoing content creation and retirement process. It answers questions such as the following:

- What type is each element (free text, pattern text, open list, closed list, and so on)?
- For each element, what are the allowable values (or maximum character lengths for text fields)?
- What are the rules for changing the allowable values of each element?
- Which elements are authors responsible to fill in? Which do other staffers fill in? Which are filled in automatically?
- What are the review and quality-control processes for the metadata elements of each component class?
- How many of each component class do you expect to add each week?
- Can some component classes have an indefinite number of components, and which ones may reach a maximum limit?
- What tells you that a component is ready for expiration, archiving, or deletion?

Cross-Reference

You can find more information about the metatorial guide in Chapter 20, "Working with Metadata," in the section "Introducing the Metatorial Framework."

If you've completed your design, you already have answers to most of these questions and can simply organize and publish them in an easy-to-use format, such as a Web page or printed guide.

The audience for the metatorial guide is the metator or metators of your CMS. You distribute appropriate pieces of the guide, however, to anyone who touches metadata. Thus you need to write it in a generally understandable manner. I suggest organizing the guide as a reference, listing all elements alphabetically. Include indexes that organize elements by component class and by type.

The fundamental question that each section on an element must answer is, "How do I fill in this element?" Your metator or metatorial team leader should create and maintain the guide.

Tip

Don't forget to account for localization in your metatorial guide. Ask not only how to fill in each element, but also how to fill in each element in each region to which you intend to localize.

The editorial guide

An *editorial guide* is substantially the same in a CMS context as it is in any publication's production environment. Following are some of the differences that may warrant changes to the way that you may otherwise produce an editorial guide:

- **Organize it the same as you do the metatorial guide.** You may consider organizing your editorial guide by element, component, and type. You can organize your comments into three layers. Any style or usage comments that are global to an element can go in the section on the element. Any comments that apply to the usage of an element in the context of a particular component can go in the component. You can create overview sections for comments that apply to the entire system. If this organizational scheme seems too complex for you (and you can't find a database programmer to help), I think you're still best off if you organize by element. That way, people creating content can most easily locate the specific guidelines for that element within the particular components with which they're working.
- **Account for variation in the usage or style per publication and audience.** Creating a single style that works for all audiences and publications is obviously easiest, but doing so isn't always possible or advisable given that different audiences and publications have very different needs. In these cases, listing style variations in families of related elements and explaining them separately may prove your best option. Suppose, for example, that you decide to create a novice summary and an expert summary of a white paper. You can create two overview elements that you fill in separately, basing them on two sets of style guidelines. The system can select which overview to use for which audience and publication, and the author knows exactly how to write each of the two summaries.
- **Distribute it on the spot.** If at all possible, make your editorial (and metatorial) guide information available in the application that authors use to create content. If you use Web-based forms, for example, include a link next to each control on the form to access the metatorial and editorial guidelines behind that control. (As you probably know, each control is the interface to a single element.) You can see why organizing by element is helpful.

- **Account for any localization.** If you plan to localize, include specific guidelines in your editorial guide. Quite possibly, your localization group knows better how your guidelines need modification for other languages and regions, so make sure that you consult with it. In any case, include localization style and usage guidelines per element and region.

The administrator guide

The CMS administrator and her assignees need comprehensive documentation on how you put together and configured the system. In a way, an *administrator's guide* is the sum total of all the system configuration and setup that you do. Thus, even if you do no more than gather this information from the specs and other sources that you may have and then put it all in one place for the administrator to access, you're doing enough. Don't expect authors and processing teams to plow through raw information to learn what they need. You can, on the other hand, expect that a CMS administrator's familiar enough with the source documents and the system itself to find the needle that she needs in the haystack of files. As you gather the administrator's stack, don't forget to include the following:

- Any system diagrams that you may create that show the relation of parts of the CMS.
- The documentation from all third-party software that you include.
- All support agreements that you sign.
- The full documentation set that you prepare for other staff. (The administrator, by the way, is a good point of distribution for this stuff.)

The one thing that you may need to create for an administrator's guide is the list of responsibilities, the reporting procedure, and the escalation path for the administrator to follow. This material should be mostly present in the staffing plan that you create.

The template guide

The *template guide* is the living form of the template specification that you create for implementation. Undoubtedly, when you needed to do the template implementation, things changed and didn't find their way back into the specification, so you're likely to need to do some rewriting. The audience for the template guide consists of the following people:

- **The publication staff member** who wants to know how something was done (usually to know how difficult it is to change it). These people try to match a visual element in the final publication to some sort of definition in the guide.
- **The programmer** who wants to learn the templating system before trying to fix it or change it.

For the publication staff, make sure that you can find your way around the guide by publication, page, and design element. For the programmer, make sure that you feature the code-sharing and version-control system prominently in the documentation so that she doesn't think to subvert it.

For more experienced staffers, the templates speak for themselves, so don't put too much effort into a template guide for them. You can expect the users of this guide to be fairly expert and not need much hand-holding. Still, you want to at least publish and enforce the programming and design standards by which you expect publication staff and programmers to live.

Doing training

The kind and amount of training that you may need to prepare depends quite a bit on the particulars of your situation. If the team that creates the CMS stays on to run it, for example, you need little staff training. If you create a strong user interface for authors, you need little author training. For the purposes of discussion, however, I assume that you need to train everyone to some extent. If you do, the following list describes the sorts of training that you may provide during the deployment of your system:

- **Content management training:** You may want to create a general training course on what content management is, how your system fits together, and what each person is responsible for. This training follows the CMS overview document and serves to reinforce its main points.
- **Author training:** Especially for the novice authors, you may want to offer a short training session on how to use the Web-based forms, workflow interface, and other tools they have. While you have the ear of the authors, delivering them editorial and metatorial training is a good idea as well.
- **Processor training:** Content-processing teams tend to be task-orientated, low in skills, and high in turnover. For processes that require more than a few minutes of training, you're well served by creating some form of training course. These training methods, of course, focus on the particular tasks that a certain kind of processor needs to do and on the tools that she uses.
- **Administrator training:** If you have only one administrator, you hardly need to develop training. If you have multiple sites where content originates or is deployed, however, you have parts of the CMS in locations that are possibly quite far apart. If so, you may want to make sure that each site has some sort of administrator and that each administrator has the same sort of training.
- **Page developer training:** If you have turnover in the page-developer group, you may want to standardize the training that new ones receive. This training follows the template guide and focuses on the overall structure of the template system, the names and positions of all the design elements, and the HTML and other page code behind the templates. (See Chapter 11, "Staffing a CMS," in the section "Page developer," for more information on the job of page developer.)

- **Template developer training:** Just as is true of the page developer, template developers may come and go. If they do, training is a good idea. This training overlaps with the page-developer training at the level of the page code (HTML or other) and goes on to teach the programming methods that the system uses, the code-sharing system, and the version-control techniques that you're employing. (See Chapter 11, "Staffing a CMS," in the section "Template and CMS developer," for more information on the job of template developer.)
- **Localization training:** Translators and other localizers of content need to know enough about the CMS to interact with it. You best hope is that you can extend the CMS to your localizers so that they can work directly with content. If so, their training resembles the author training in the use of CMS Web-based forms, workflow interface, and other input tools. Even if you must export content to the localization group, you still want to train them on how not to mess up the exported text so that you can import it again. Localizing your CMS documentation may even prove worthwhile, although perhaps you can localize just the most basic training material. In most countries, technical people expect to read documentation in English. (You find more information about localization in Chapter 21, "Cataloging Audiences," in the section "Localization.")
- **Deployment training:** If you must deploy your system over time in a variety of locations, you may want to create a training regime for the people who must install and support the system in other locations. This training follows the deployment specification and provides information on system installation, configuration, and integration.

Just because you're training people, you don't need to make it seem like school. The only essential thing about training is that it results in usable knowledge. Assuming that, late in the project, time and money are at a low point, you can scale back your training to the least-effort activity that still gets the result that you want. Following are the kinds of training that I usually see:

- **Read the documentation:** If you do a good job on your documentation, it often suffices for people who already have experience in content management tasks or whose job doesn't require anything more in-depth. A template programmer who's already done a number of template systems using your chosen CMS, for example, usually needs no more than your documentation to get going.
- **Self service:** If you do a good job on your electronic documentation (chunking it into components and making them accessible independently of what surrounds them), creating tutorials or other self-paced learning guides is usually pretty easy. Of course, the more effort that you put into this type of training, the better a learning tool it is, but the minimum is simply to arrange the most important parts of the documentation in a sequence, with comments and supplementary material in between. This approach may prove an appropriate way to train new content processors or authors whom you can move through the process that you expect them to do one step at a time and at their own speeds.

- **Product company training:** Most CMS product companies employ some sort of product training. For the most part, this training is more suitable for your project team before it starts an implementation than it is for your ongoing staff. Still, product-company training provides a convenient alternative to creating or delivering training yourself. Usually, you can get one- to three-day sessions on authoring, configuration, and basic programming techniques. Of course, this training is generic and doesn't target your particular system, so you still may want to augment it (or pay the company to augment it) to deal with your customizations and particular configuration.
- **Apprenticeship:** You may get good results from attaching a new person to someone who's already doing a job. If you pursue this path, make sure that you brief both the existing staff member and the trainee on your learning goals and hold them to these goals. That way, everyone takes the apprenticeship seriously and makes it result in a trained person. Apprenticeships are particularly useful if a lot of informal information is necessary to do a job. CMS administrators and production managers, for example, possess a lot of process knowledge and useful tips in their heads and not on paper. For these positions, an apprentice can learn a lot. If you're smart, you instruct the apprentice to document any information that she gets orally so that as little as possible remains informal.
- **Boot camp:** The boot camp idea involves creating a fast, intense experience where the student works exceedingly hard and comes out with workable knowledge right away. Boot camps are mostly an immersion and not a learning experience. In other words, they produce an attitude more than a set of skills. A person can retain only so much from the masses of information that you present in the one or two days of boot camp. You can, however, present a person all the material that she needs to go back and really learn later on. And, more important, the experience can give the person a mindset that she can take on and use over the course of her job. (Picture the army recruit that a drill instructor strips down and rebuilds in boot camp.) Because CMS jobs often require an attitude as much as they require particular skills, a boot-camp approach can prove useful. You may want to consider creating a boot camp for all full-time staff prior to final launch. You can cover the CMS overview material in more depth than you can in the overview document, introduce all the other key documents to which you expect people to refer, and run exercises that foster teamwork and appropriate divisions of responsibility between collection, management, and publishing staff.
- **Long-term training:** This type of training is the traditional idea of a class, with assignments, texts, and tests. To the extent that you have the time or resources to mount such classes, they're a great way to really teach skills. I haven't seen many CMS organizations that are large enough to warrant creating and running classes. Sometimes, however, you can find classes in colleges and universities that provide enough overlap with the skills that you need that you can support staff members who want to attend them.

Powering up the system

As part of deploying the CMS, you want to power it up slowly. You can begin powering up the CMS at the beginning of implementation. Throughout implementation, you can bring more and more of the system into operation until, during deployment, you have all the CMS systems running.

Note Your CMS product vendor or IT staffers may have different ideas about how to best power up the system. Listen to them, but see how much of what I present here you can work in to their plan. The most essential thing to preserve is the capability to retreat back to a system that works at any point.

Following are the steps that you may want to follow to bring the CMS to full power:

- Get the basic CMS package running in a simple, standalone environment.
- Get the basic package running in your production environment (that is, the full environment in which the CMS eventually must operate).
- Progressively add content and publications to the standalone environment and then add them to the real environment.
- Progressively add third-party software and integrations to the production environment.
- Release a beta system that can do everything that the final system can do.
- Release a final system that you can trust.

I explain each of these steps further in the following sections.

The generic installation

Starting with a generic installation that has as few external complexities to it as possible is always your best course. This type of startup gives you proof that the CMS in its native state can work without difficulty.

Native state, by the way, means the following:

- **Using one or two computers:** The product company may recommend that you use one computer for the CMS server and another one for the database server.
- **Outfitted to the product company's exact system requirements:** These requirements include operating system service packs, patches, and so on.
- **Using whatever sample database structure and content the product company provides:** The samples that come with the system provide a good baseline that you can always return to if the system gets messed up later.

- **Not connected to any external systems except your LAN:** Keep the connectivity as minimal as possible at first to avoid complicating factors. You have plenty of time to complicate the system later.
- **All the features of the CMS installed:** You want to install all features, even if you don't immediately need them, so that you expose as many CMS internal dependencies as possible.

The preceding configuration is your CMS reference platform. Later, if you can't get something to work on your production installation, you can try it on the generic installation to see whether it works if you remove the rest of your infrastructure. The generic installation also gives you practice and confidence installing and configuring the CMS.

Finally, this platform serves as the testing ground for content and publications as you move through implementation.

Cross-Reference

You can find more information about the generic installation in Chapter 17, "Implementing the System," in the section "Installing the system."

The production installation

Soon after you get the generic installation running, you want to start to build your real infrastructure. Your real infrastructure includes the following items:

- The Web, file, and database server hardware and software that you intend to use. Preferably this hardware and software isn't what you currently use, but rather a different set of computers that you eventually use.
- At least one typical end-user machine running the authoring tools.
- An administrator's machine running the full CMS interface suite.
- End-user machines running the operating systems and browsers that you expect your audience members to run.
- The Web server and application server that you intend to use.
- Your determined LAN and Internet connectivity.
- Any firewall and security software that you plan to run.
- Whatever database connectivity that you plan to employ.
- The replication and deployment systems that you intend to use to move and synchronize content at your various locations.

The production installation is also a reference installation. It shows that the selected CMS can run inside your chosen infrastructure before adding the complexity of content, publications, third-party software, and integrations. Make sure that you fully document the installation and configuration of this system. If possible, make disk images of each of the machines that you set up so that you can restore them later if they get really messed up.

Over the course of implementation, you continually add third-party software, system integrations, content, and publications to this system. Make sure that, as you do, you can always back up to the last working configuration.

Content and publications

The generic installation is the one to use for initially loading content and accessing publications. The production installation is much less stable and may go down a lot of the time during implementation. In addition, you can put the generic installation on a computer in someone's office or in a common workspace, where you can reboot and even totally reconfigure it, as necessary, to keep you going.

As you finalize content structure and begin to collect components, load them into the generic installation of the CMS. As you develop and finalize your database or XML structures, this machine contains the master copy of the content structure that you're creating. At each convenient milestone, you can upload the content set from the generic installation to the production installation to give it more content for testing.

Tip As much as possible, keep a copy of content outside the CMS. Make good backups of the content that you process – before loading it into the CMS database. For content that you load directly into the CMS through Web-based forms, back up the database records frequently. If the database becomes corrupted or otherwise unusable, you're always glad that you did the backup.

Build your collection Web-based forms and batch upload tools to access the generic installation at first. Doing so isolates them from any infrastructure complications as you're first creating them. Begin to test them on the production installation as soon as it's ready, but continue to develop them against the generic installation for as long as feasible to keep your collection effort out of the way of the infrastructure effort.

Build and test your publications on the generic installation first. It's easier to manipulate and more robust than the production installation. Of course, you can go only so far on the generic installation. You install none of your publication system integrations or third-party add-ons on the generic machine. Still, you can work through the majority of content access, navigation, and layout issues in the easy environment before moving on to the real world.

Third-party software and integrations

While working through collection and publication on the generic installation, you can work through connection and integration on the production system. Add the third-party products that you intend to use and the integrations that you expect to need to the infrastructure system in whatever order seems to cause the least problems (for the schedule as well as for the system). Bring in only as much content and publication stuff from the generic installation as you need to test the integrations you're doing.

Test as generic a connection as possible to each system first. Then, after you're sure that the generic connection works, complicate the system by bringing in the particular connection that the CMS requires. If you need to connect a publication to an enterprise data system, for example, make sure first that you can get any data in and out of that enterprise system. Then go for the specific data that you need to exchange. Finally, access the data that you need by using the template that accesses it in the final system.

As the system nears full connectivity in the management and publishing system, you're ready to consider switching over fully from the generic to the production installation.

The beta system

After your production installation has a full complement of connections to other products and systems and you test all collection and publication interconnections, you're ready to create a beta CMS.

Cross-Reference For more information on beta tests, refer to Chapter 17, "Implementing the System," in the section "Beta tests."

The beta system is a full working version of the system that you think you're finally going to use. All content is flowing into it, and all publications, fully formed, are flowing out of it. In other words, it's a major cause for celebration!

From the standpoint of deployment, the major issue with the beta system is when to flip the switch on any systems that the CMS is replacing. You're likely not to stop creating your Web sites and other publications the old way until now. Now, however, is the time to flip the switch and start treating the CMS as your only source and destination for content. This move is often scary (especially if you're not quite as organized and fastidious as I recommend). If you're unsure, you can always publish in parallel while you work through the final kinks. If you do publish in parallel, realize the following:

- ✦ **Expect to take significant extra efforts** to collect content and form it into publication pages in two different ways.
- ✦ **Plan on having a difficult time** making sure that the changes that you make in one system end up in the other.

Regardless of the extra effort, supporting two efforts for a while instead of abruptly shutting the old one down, only to find that the new one can't yet make its deadlines, may prove well worth your while.

Another way to approach the change from the old to the new system is to specify a "freeze period," a (short, you hope) period of time when you stop updating content the old way so that the content processors can have a fixed set of content to migrate. Finally, if neither a freeze or parallel publishing works, you can get a “cut” of the content on a certain date and then track every single change to the content after that so that it can get into the new system in the final hours before it goes live.

The full staff may or may not use the beta system, and it may or may not create publications that all audience members see. To power up more slowly, you can release the beta system to one contributor group or one publication at a time. You can use it to create only some of the pages on your Web site, or you can use it to create a separate site to which you route only a smaller percentage of users.

Tip

A simple Web script can route a percentage of your site visitors to a different URL. You can install a script that selects randomly or by any other user-profile data that you can ascertain (audience type is best, if you can detect it) and then redirects the browser of chosen visitors to your test pages.

However slowly you begin, by the end of the test, you should have the complete system in place, accepting all content and creating all publications. Another possibility is that the test fails and you’re back a step or two — but I’m assuming that’s not the case.

The final system

The way that I lay out the deployment process, the final system is no more than the last small step. This approach minimizes surprises and makes sure that the final launch of the CMS is as uneventful as possible.

Besides its benefits for testing and deployment purposes, the incremental process that I propose in this chapter is a good idea for another reason. Nothing particularly is final about the final system. The whole point of a CMS is to give you the capability to change and create content and publications without some arbitrary versioning or release cycle for your publications. Of course, you still may choose to create big launches and release dates for certain publications, but you don’t need to do so. And as the world gets more and more accustomed to the continuous-update model of the Web, you’re called on less and less to create distinct publication versions and releases.

More than an end, the final system is the beginning of a new process. You’re no longer building — you’re maintaining. You’re out of the startup phase and into the run phase of your CMS. You’re now beyond the CMS project and into the CMS as an integral part of the organization.

From now on, your work results in innumerable small improvements to the processes that you initiate rather than in overhauls and complete revolutions. Don’t just sigh and move on, however; take a break to pop a cork or two and sing the praises of your team, which has done an amazing amount of work, and your organization, which let them get away with changing everything!

Preparing content

After the release parties all end, you may find that you already have a backlog of content to get into the system. Did you hold off on new content near the end to make sure that your first set of publications was just right? You can get away with this approach at your first release but not in the longer run. Rather, you must set up the process so that content flows in independently of how it flows out. In fact, if you're smart, you simply queued up new content in a workflow stage rather than stopping its production entirely.

As part of your logical design, in your specs, and finally in your authoring and acquisition guides, you worked through and presented everything that you need to say about moving content into the system. Now the trick is to make it work the way that you planned it. First, prepare to change. Make sure that your guides have an owner who commits to keeping them up to date with the real production rates and processes that the systems uses. As you move from the project phase (where everyone expects extraordinary effort) to the run phase (where you can expect only ordinary effort), you're certain to need to readjust your expectations and find a point of stability where you can really count on the flow of new information. For the long-term viability of your CMS, finding a sustainable production rate is much more important than getting all information into the system as soon as possible. During the CMS project, you want the hare's agility and speed. During the running of the CMS, you want the tortoise's slow but always-steady plod.

Tip

As you move from starting up to running a CMS, expect some of your top talent to leave. The hare that prefers the adrenaline of a fast-paced project may not want to become the tortoise that you can count on for years to come.

Second, adjust your processing team structure to the new environment. You're likely to need only the top talent from your processing teams to handle the smaller quantities of information that you have after you make your way past the backlog. You can use the same processes, but make sure that you review them. You designed them to move information at a high rate. After you slow the rate and make a smaller team, you probably need to rethink some of your processes and job descriptions. You may also want to go from a model of one project team per source to one department of content processing that handles all sources.

Revising the system

As much as you may want it to happen, in the run phase of a CMS, none of the project activities stop; they just slow down.

Of course, you face the inevitable bugs, which may call for a burst of activity that reminds everyone of the bad old project days, but unless you miss a lot in testing, these bugs are normally just short squalls on an otherwise calm sea. I'd advise you to put a good plan in place for quickly discovering the source of any bug and bringing as much power as necessary to bear to quickly address any potential problem.

If your system is a success, you know immediately, because the following things occur:

- **Content appears out of the woodwork:** People you couldn't ever pin down and people you've never even heard of (despite your great search) turn up and want you to "hook up" their content.
- **Publication ideas flourish:** In addition to publication teams approaching you to include their stuff, you find that everyone has a great idea for a new page or publication that you can produce.
- **Personalization moves to the forefront:** Seeing the new possibilities of the system, you get more and more requests to slice content in a particular way for a particular person or group.

These signs of success may seem more the harbingers of your imminent demise if you don't have a way to handle them. The worst thing that you can do is open the floodgates, embrace them all, and get to work. A better move is to establish a clear process for evaluating requests and preparing a measured response.

Cross-Reference

Chapter 12, "Working within the Organization," in the section "Using Functional Collection and Publishing," provides a detailed discussion of how you may structure your organization to handle requests and changes.

As you entertain revisions to the system, I believe that the most important thing to do is to preserve the integrity of the logical design. If you add a new information source, for example, it may introduce changes to your component and element models, author or source assumptions, and publication-template models and personalization routines. If you don't consider the changes carefully and work them through for each change, you eventually end up in a bind, where the new stuff breaks the old stuff. Maybe the new source that you're adding overlaps with the information in one of the existing components, for example, and somehow appears on the same page and looks redundant.

Cross-Reference

Chapter 19, "The Wheel of Content Management," addresses the connections between parts of the logical design. You can use it to create a process for assessing the effects of changes on your CMS.

A question that you eventually must address is, "When's version 2 coming out?" You can handle this issue in either of the following two ways:

- **No version 2 is necessary:** Following the idea that the running CMS evolves rather than revolutionizes, you can say that you make changes as they're suggested and ratified and that the system, therefore, never actually needs a new version.

- **Hold back for a major pass:** Given the realities of budgeting and staffing, you may decide to make only small changes in the normal course of running the CMS, holding back major changes for a defined next CMS project cycle.

The first method makes the most of the CMS and enables it to evolve as necessary. The second approach is more realistic for organizations that can't muster the ongoing budget or staffing necessary to continually review and develop new functionality. You may want to start out on a defined revision cycle and move into continuous revision as the system proves its value. If you do choose revision cycles, you must decide what the difference is between a small and large change. The only natural break that I can think of that works for any situation is as follows: Making changes to existing components and publications are small changes, while creating new ones are large changes.

Of course, continuous change doesn't mean no schedules. It simply means that a particular change happens on its own schedule, which is often independent of any other change.

Taking Stock of the Deliverables

Aside from delivering the CMS itself, during deployment you can make ready a set of plans and specifications that ease the transition into the new world of the CMS. Although I list these deliverables in the deployment stage, you may very well need to begin to create them during implementation to have them ready at deployment time.

The deployment specification

The *deployment specification*, which I describe in this section, you actually want to develop along with the other specifications during implementation. It details the process that you undertake to roll out your system. The deployment specification includes the following parts:

- **The power-up schedule:** This schedule details the plan that you devise for powering up the system. Especially during the earlier stages of development, it's critical so that each team knows your expectations for when you need its deliverables to continue building out the production system.
- **The CMS system architecture:** This architecture includes all the parts that I mention in the section " The production installation," earlier in this chapter. The system architecture should include system diagrams as well as hardware and software requirements for the generic and production installations.
- **Integration plans:** These plans include schedules and techniques to use to connect the CMS to all third-party products and enterprise systems.

- **Staff rollout plans:** These plans detail how and when you deploy the collection system to different contribution groups (author groups and acquisition groups).
- **Publication rollout plans:** These plans describe the order and timing in which CMS-produced publications replace the current publications. This plan shows exactly when the old ones stop and the new ones begin. If you're planning a period during which the old and new publication productions overlap, the plan needs to show how you're handling staffing and synchronization.
- **The beta plan:** This plan describes the timing and participation that you expect in a beta release. If you're including certain audience groups, the plan needs to state how you plan to ask for or notify them of their participation.
- **Expected and allowed volumes:** The deployment plan needs to set the expected range and the maximum values for the page views and other transactions that occur in the CMS at release. It should include contingency plans for how to bring more hardware or bandwidth to bear if the actual volumes (of page requests, database queries, or other publication pages, for example) exceed the maximum.
- **The distributed system plan:** If the CMS system or databases is spread over various locations, the specification should provide a schedule and staffing plan for how the deployment spreads through the set of locations.
- **A deployment staffing plan:** This plan shows how many and what types of people you need to power up and roll out the system.
- **Setup and configuration data:** You need this data for all the end-user tools that you must deploy to contributors and other CMS staff.

The deployment specification dovetails with the implementation specs. Whereas each implementation specification shows how to build a particular part of the CMS, the deployment specification shows how all the parts come together as you build them to form the production system and how you intend to progressively release that production system to staff and audiences.

The staffing plan

The *staffing plan* that you produce for deployment shows the kinds and numbers of people that you need to run your system. All the input for the plan comes from the workflow and staffing analysis that you do during logical design.

Cross-Reference

Chapter 27, "Designing Workflow and Staffing Models," details the logical design of staffing.

The plan should include the following:

- **An assessment of your current content management staffing:** Include all the contributors that formally or informally contribute content now and all the project staff that you've mustered.
- **Job descriptions:** Gather descriptions for all the people that you need to run the system.
- **The gap in staff:** What's the gap between those people you have now and those you still need to run the system?
- **The gap in skills or training:** What skills or training are necessary for the people who are to run the system?

This plan is the culmination of the process that you expect to use to secure CMS staffing and not the beginning of it. If you suspect that you can't turn the project team into the running teams, start as early as possible to secure funding and support for finding your ongoing CMS team.

The training plan

I discuss the training possibilities themselves in the section "Doing training," earlier in this chapter. In addition, you need to plan how you're going to develop and deliver the training materials. In the training plans, make sure that you include the following:

- **Who's to create the training?** You can sometimes use an instructional designer from the product company's professional services division or an outside contractor. Failing that, the people creating the documentation may prove a good choice. I'd go last to the project team members, who probably are busy with other tasks and are unlikely to have any particular skills in creating training.
- **When can you complete it?** Obviously, having the training plans ready to go well in advance of the beta release of the system is your best course. Doing so gives you the most time to train staff before they ever need to do real work. This timing is pretty ambitious, however, given the rest of what must happen before the beta release. The next fall back is to make training ready at the beta release. Then you can train as people ramp into using the beta release. If you simply can't ready the training by the beta release, make other plans for how to train the initial staff.
- **Who's to take the training and when?** Schedule the delivery of the training around the power-up plan, making sure that you train people in advance of calling on them to use the system. Pay attention, too, to the amount of training that each individual needs. More experienced staff can often get ready just by reading the documentation.

You don't need to overplan training. A simple spreadsheet is all that you really need to chart out the creation and delivery of training. The difficult part of the training plan is ensuring that training is ready in time to prove useful at the release of the system. If you can't find a way to train people before they begin work, the value of the training severely decreases and you need to re-evaluate your plan. Less training, delivered earlier, may provide a higher overall value to you.

The documentation plan

In the section "Creating documentation," earlier in this chapter, I discuss the documentation that you may want to create. In addition, you need to plan how to create all the different documents in the specified time with the resources that you have. In your documentation plan, make sure that you include the following:

- **What document to create?** Plot out the migration of information from the specs and other project documents that you've created so far to the documentation set. This exercise tells you how much extra work is necessary to complete the documentation, given the existing information. It also helps you figure out who you need to contribute to each document.
- **Who creates documents?** My first choice for the document writers is the authors of your specifications, augmenting them with a professional writer who can take the raw materials that the authors supply and turn them into readable, accessible publications. This method gives you the most leverage of the limited time that analysts and other authors may have near the end of the project.
- **A publication plan:** This plan needs to include the formats in which the documentation is delivered. I suggest that you focus first on HTML versions, because they give you the most flexibility (in repurposing) for the least effort. As much as you can manage, make this plan a microcosm of the wider CMS — that is, make sure that it covers how to segment the content and how to template and build the documentation.
- **A release plan:** This plan needs to shows when each document is ready. As is the case with training, getting documentation ready well before your beta release is preferable. At beta is fine as well. If you can't manage to get the documentation ready at beta, you want to decide how to plug the information gap with selected parts of the specs. Make sure that you prioritize the creation and release of the documentation to correspond to the major information needs that you expect to have at and after the beta stage.

If you treat your CMS information as content to collect, manage, and publish, you have no problem creating targeted, well-delivered documentation. If you went as far as using a CMS to create the specs, you're already in a position to take complete control of publication. If you didn't use a CMS (or other Web-authoring environment) to create specs, you're not too late to start doing so. Organize the creation and publication of your documentation now to ensure that you can update and repurpose it back into specifications for your next round of changes to the CMS. The generic installation of the CMS is often the perfect server for the ongoing collection and distribution of CMS-related content.

The maintenance plan

A *maintenance plan* details how to administer the system over time. It's the marching orders for the CMS administrator and support staff. Include the following types of information in your maintenance plan:

- **Backups:** How often do you back up the CMS repository and how does this backup occur?
- **Content retirement:** How do you archive or delete content from the repository and what are the procedures for doing so?
- **Revisions:** What are the rules for modifying CMS components, configuration, or publishing routines? Who must agree to a change, who makes the change, and who tests it? Include here, too, the expected and maximum amount of change that your administrators must do. What are your contingency plans if the amount of change surpasses the maximum for which you've planned?
- **Turnaround times:** For each type of request for a change to the system, what's the expected and maximum amount of time until the change is complete?
- **Technical support plan:** This plan can entail an entire document of its own if you're going to create and run your own technical-support group. If so, use the technical-support criteria that I describe in Chapter 16, "Selecting Hardware and Software," in the section "Technical support," to get a start on what you need to design and build. At the very least, your technical support plan must catalog expected problem types. For each type of problem, it needs to provide an initial contact, an escalation procedure, an expected time for resolution, and expectations for logging and reporting on the problem. In addition, the plan needs to overview the kinds, numbers, and distribution of technical support staffing that you expect to engage.
- **Connections:** The maintenance plan needs to include a section on each connection that the CMS makes to outside systems (enterprise data, Web transaction, and other systems). For each connection, the plan needs to state any periodic tasks that must occur, any troubleshooting tips that the implementation teams come up with, and any diagrams or definitions of the way that you make connections to the system.
- **Logs and reports:** The maintenance plan needs to specify logging and reporting requirements such as the following: which logs you must maintain and what data they need to include, what sorts of reports you need to generate from logs and databases, how often you need to produce reports, how often to clear or archive logs, who gets which reports, and what the expected minimum and maximum allowed values are for each key report data field. If a report data field goes below the allowed minimum or above the allowed maximum, the plan must state what the required action is. If, for example, you access a component fewer than two times per month, you may want to trigger a workflow to review its accessibility and desirability.

- **Maintenance:** The plan should contain schedules for routine maintenance tasks, such as clearing out old pages from Web sites, reviewing metadata lists, reviewing personalization rules, and purging bad data from the user profile tables.
- **Maintenance contracts and contacts:** For the parts of CMS maintenance that outside agencies (technical support groups, contract programmers, publication facilities, and so on) do, the plan needs to include or reference the agreement that you have with them and list the contact people for those organizations.

The maintenance plan tries to anticipate any issue that may arise during the running of the CMS. Of course, you're sure to miss a lot, so prepare yourself (or your system administrator) ahead of time to update and maintain the plan.

Summary

Rolling out your CMS is a matter of making it part of the daily life of your organization. In other words, expect a transition between the time before the CMS and the time after which the CMS no longer seems strange or cumbersome. You can ease that transition by doing the following:

- Preparing a complete and easily accessible documentation set.
- Bringing the system into its final production environment in a series of steps.
- Preparing for the ongoing processing of content and publications from the system.
- Figuring out in advance how to support and fix the system after its release.
- Preparing in advance for an inevitable series of enhancements and versions that come quickly on the heals of your first release.

That's it for the project process. In the following part of the book, I take you back into logical design.

✦ ✦ ✦

PART

PART

IV

The Logical Design of a CMS

◆ ◆ ◆ ◆

In This Part

Chapter 19
The Wheel of Content Management

Chapter 20
Working with Metadata

Chapter 21
Cataloging Audiences

Chapter 22
Designing Publications

Chapter 23
Designing Content Components

Chapter 24
Accounting for Authors

Chapter 25
Accounting for Acquisition Sources

Chapter 26
Designing Content Access Structures

Chapter 27
Designing Workflow and Staffing Models

◆ ◆ ◆ ◆

CHAPTER 19

The Wheel of Content Management

The traditional way to create and update publications is to establish a direct connection between an author and a publication. In a CMS, you create an indirect connection between author and publication through a collection, management, and publishing system. This system enables the work of authors to combine with other source materials to create a variety of publications. You can understand a CMS as a set of significant players or entities that all interact to create a living system.

I discuss the content management entities in one form or another all through this book. Now is the time to come directly at them and describe what they are, what they contain, and how they interact. In this chapter, I present an overview of the entities and discuss their major relationships. In the chapters that follow, I describe each entity individually and discuss at length how you may analyze it and use it to construct the logical design of your CMS.

◆ ◆ ◆ ◆

In This Chapter

An introduction to the content management entities

The information that you need to collect on all the entities

The relationships that you need to forge between the entities

How to start collecting information in a logical analysis

An introduction to PLAN International, an example organization

◆ ◆ ◆ ◆

The Content Management Entities

Regardless of any CMS products that you may currently use or how you structure your CMS project, you must face a certain set of issues to make the most of your system. These issues group together into the following eight categories that, by now, probably seem quite familiar to you:

- Goals and requirements
- Audiences

- Publications
- Content component classes
- Authors
- Acquisition sources
- Workflow and staffing
- Access structures

I call these categories the *content management entities* because they're the basic players in a content management system. To create a CMS, you must account for each entity and ensure that you design the system in accordance with it. The entities are significant issues in your system, but they're also forces to be reckoned with. Each not only needs you to address it, but each also affects all the others. Thus the entities are constraints as well as players. They put limits on each other and decide for you what's possible and what isn't. You may, for example, want to serve 10 different audiences until you find out what that means for the complexity of your publications. You may want to acquire content from a certain source but find that forming it into components is too much effort.

Entities broaden as well as constrain the system. The set of content component classes that you choose to create can, for example, give you ideas for new publications that you don't think feasible at the start of your design. Your audiences can suggest access structures that make personalization more natural. Your sources can suggest a set of content components.

The word *entity* has another connotation as well. In relational database design, an *entity* is a table or other significant data structure that you build your design around. In an inventory database, for example, you may use a product entity as well as entities for locations and users. You associate a set of data with each entity. A product gets an ID, a name, a description, a price, and so on. A location gets an ID, an address, a phone number and a size, and so on. A user gets an ID, a name, an e-mail address, a job title, and so on.

Each entity has relationships with others. You warehouse a product, for example, at a location. A location needs a manager, and a particular user of the system is in charge of a product. Figure 19-1 provides an example of how to depict entities and relationships in database diagrams.

Each entity has a table that holds its data set. If its relationship with another entity is one-to-many—one user is in charge of many products, for example—you can include the ID of one entity in the data set of the other. If the relationship is many-to-many—for example, some products may reside in many locations and each location may contain many products—you need an intermediate table to hold the matched sets of IDs. Although it doesn't appear in the example in the preceding figure, the intermediate table can also hold additional data about the relationship. You can, for example, specify whether a certain location is the primary distribution point for a product.

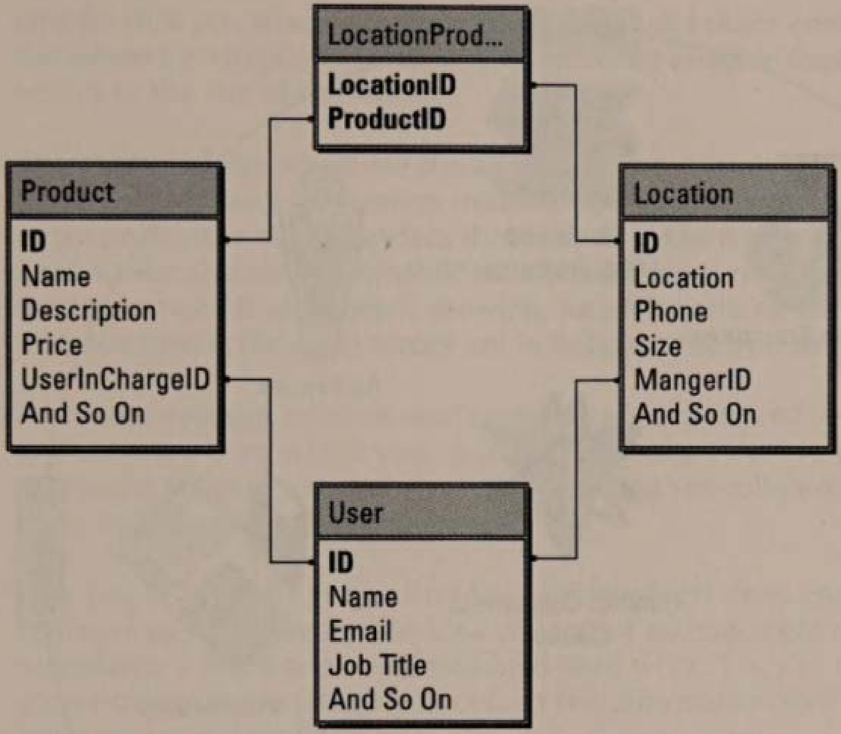

LocationProd...
LocationID
ProductID
Product
ID
Name
Description
Price
UserInChargeID
And So On
Location
ID
Location
Phone
Size
MangerID
And So On
User
ID
Name
Email
Job Title
And So On

Figure 19-1: Entities and relationships in a simple database

Whenever I talk about content management entities, I mean them in the database sense as well as in the logical design sense. And as you can see throughout this chapter, the logical design of a CMS entails defining each entity and determining its relationships to the other entities.

Introducing the CMS wheel

You may recall (or, by now, you may be tired of me saying) that I define a CMS as a system that collects, manages, and publishes information and functionality. I want now to offer another definition—just as valid—that I believe can deepen your understanding of what really goes on in a CMS: A *content management system* is a *balanced interaction* between the forces of the significant entities that control it. Metadata maintains the balance.

If the entities are in balance, they form a stable and effective system for content creation and use. I envision that system as a wheel, as shown in Figure 19-2.

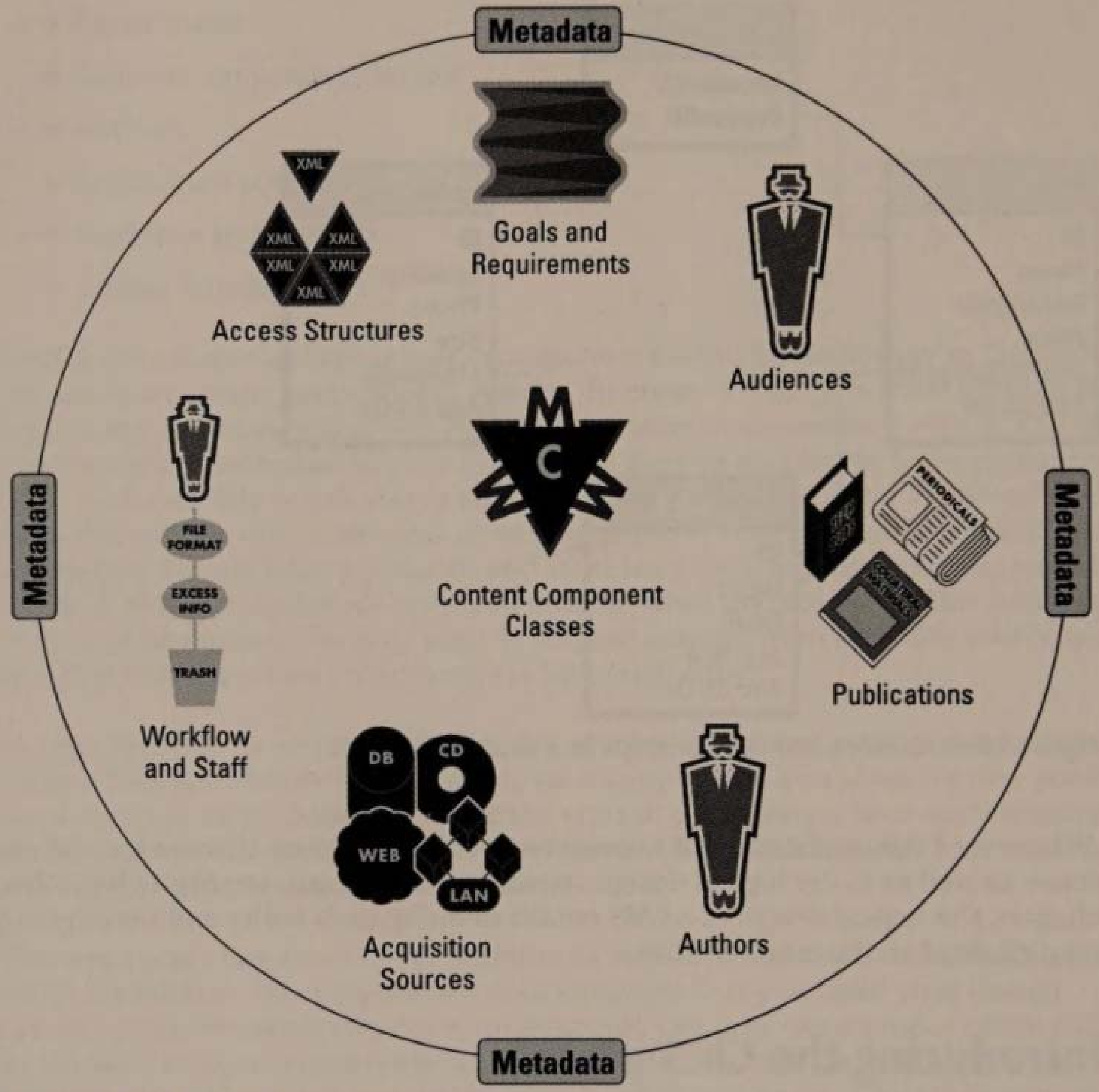

Metadata
XML
XML
XML
XML
XML
XML
Goals and Requirements
Access Structures
Audiences
M
C
Content Component Classes
Metadata
FILE FORMAT
EXCESS INFO
TRASH
Workflow and Staff
PERIODICALS
COLLATERAL MATERIALS
Publications
Metadata
DB
CD
WEB
LAN
Acquisition Sources
Authors
Metadata

Figure 19-2: The wheel of content management

At the hub of the wheel is the set of content component classes that you create. More than any other entity, the components lie at the center of the system. They're the place where all the other entities come together. They're the result of all the knowledge you've gained about the other entities and are the resources from which you draw to provide all the system's value.

The rim of the wheel is metadata. Metadata holds the system together and gives it shape. Any decision that you make ultimately results in some change to the metadata system that you construct. Metadata and components relate closely to one another. The majority of the metadata that you define is stored in the elements of the components that you define. Components are information or functionality chunks that you wrap in metadata. So although I place components in the center of the wheel for simplicity, you can just as easily imagine them spanning out from the center to the rim of the wheel.

The spokes of the wheel are the major entities. Just as do the spokes of a wheel, these entities keep the system from collapsing and from flying apart. Each spoke puts constraints on the system that tend to make it smaller and more well defined. Each spoke also offers possibilities and additional resources to the system that tend to expand it and keep it growing. As with a wheel, if the centrifugal (outward) and centripetal (inward) forces are in balance, the system is in good shape.

From this vantage, content management is independent of the particular hardware and software from which you choose to construct the system. Instead, it's an organizational process to bring a set of competing yet collaborating forces into alignment to amass and deliver value.

Lest you think that I've gone off the philosophical deep end, I can assure you that, abstruse as it may seem, the idea of content management entities, in practice, just represents a lot of good old-fashioned hard work. They're a convenient set of categories that you can use as you collect the information that you need to understand and construct your CMS.

Entities add a layer of abstraction

The traditional way to handle publications is to create a direct connection between an author and a publication. In a CMS, on the other hand, you have an *indirect* connection between author and publication. Authors create components, and publications consume components. The difference isn't that traditional publication systems ignore the content management entities. They don't. Rather, the difference is that, in traditional systems, you consider the entities implicitly, and they're just part of the author's and publisher's mindsets. You also code them directly into the process. In a traditional publication system, for example, authors know their audiences very well and write directly to them. They know how to address the audience and do so very well. In addition, the publication designers keep the audience in mind at every step as they develop publication styles and layouts.

In a CMS, the same process occurs, but it's quite explicit. Rather than residing in the minds of the authors or publication analysts, the author and publication team's understanding is in metadata values and publication templates. In other words, you make the way that you serve an audience so explicit that even a dumb computer can figure out how to serve an audience.

A CMS adds a layer of abstraction between the author and the audience. The author speaks indirectly to the audience through a set of well-crafted content components that she's sure can find their way "somehow" to the right person in the right form. The publication designer similarly designs indirectly. She creates publication structures that "somehow" are filled with appropriate and well-designed content.

The pros and cons here are simple. A direct connection is . . . more direct. It enables an author to see particularly and in detail what he's communicating and how. On the other hand, it's not very efficient and requires an inordinate amount of effort to create multiple or rapidly changing publications from the same content.

The compromise is also simple. Create a system where you can pour your best thinking about your audiences in all at once and then leave it to accomplish the task of sorting through the details of targeting content and formatting publications. And that's exactly what the logical design of a CMS is all about — putting your best thinking in a form that's complete and simple enough for a dumb computer to begin to simulate a very intelligent process.

Goals and requirements

Goals are the business concerns that drive the need for and the major uses of a CMS. You determine these goals in conjunction with your project sponsors — the key players in the organization who support and enable the CMS project. A goal is something that an organization wants to accomplish. A requirement is something that your system needs to do, be, or have to meet your goals. You make goals explicit and document them in your project mandate. In my process, you gather requirements as the first step in logical design by asking people what they suppose that you should include in the CMS. Those people may or may not derive the requirements that they give you from the goals that you provide to them, so you may need to reconcile goals and requirements during logical design.

From one standpoint, you can say that goals and requirements stand outside the CMS and provide an overall set of rules within which the CMS must live. But what if a dispute arises between a goal or requirement and one of the other entities. Which should prevail? If the answer is that the goal should always prevail, goals are indeed outside the logical design of your system. They remain constant throughout the design process. If the answer is that it depends on the dispute, goals shouldn't stand outside your design, but you must include them in your design. They may vary as your design process progresses. Suppose, for example, that you discover that creating a partner Web site (a new publication) is easy given the amount of work that you already must do to create an intranet site. Your goals, however, are to serve only staff. Should you drop the idea of a partner site or widen your goals? Suppose that you're required to show product specifications. None of the audiences that you've identified, however, is technical enough to read a product specification. Should you change the requirement or the audience divisions?

If your answer is, "It depends," or even "The goal or requirement should change," clearly you shouldn't take goals and requirements as a given but rather want to include them in your logical design as simply another kind of entity. To create the best CMS, you must balance goals and requirements with the other entities (audiences, authors, publications and so on) and trade between the two.

I'm not saying that no difference exists between goals and requirements and the other entities. Modifying goals and requirements requires you to go back to your sponsors or requirements group for permission. You can likely adjust the other entities without such permission. After a possibly exhausting mandate process, you may not want to open up discussion again. Moreover, goals and requirements tie in to other organizational initiatives. Changing them may cause an affect beyond the CMS.

So, although goals and requirements form and change beyond the logical design process, you don't need to exclude them from the process.

Cross-Reference

You find more information about sponsors and the project mandate in Chapter 14, "Securing a Project Mandate," in the section "Techniques for Getting the Job Done."

What data do you need to capture about each goal entity? As you can probably cull from my earlier discussions on the mandate process, you want to capture at least the following pieces of data:

- **A goal ID** that you can use to always know what goal you're referring to, even if its name changes.
- **A goal name** that captures the essence of the goal in a few words.
- **A description** that fully elaborates the goal.
- **A way to measure** that you've reached the goal.
- **A target value** that you want to reach for this goal (for example, number of page views, number of components, growth of registered users, and so on).

CMS *requirements* are the tangible statements of what you expect to have in the CMS and what you intend to do to create the CMS. Requirements set the overall scope of your CMS.

Cross-Reference

You can find more information about requirements in Chapter 15, "Doing Requirements and Logical Design," in the section "The requirements process."

Requirements can include the following set of data, which isn't dissimilar to that associated with goals:

- **A requirement ID** that you can use to always know what requirement you're referring to, even if its name changes.
- **A requirement name** that captures the essence of the requirement in a few words.
- **A description** that fully elaborates the requirement.
- **A type** that you can use to categorize the requirements into content, publication, and system (or whatever other scheme you feel is more appropriate).

I spend ample space describing the nature of both goals and requirements in other parts of the book. Here I want to describe a *goal entity* and how you may relate it to the other entities. You gather all the data that you need during your mandate and requirements-gathering processes. To make that data really work for you, however, you need to relate your goals to your requirements and other significant entities, as shown in Figure 19-3.

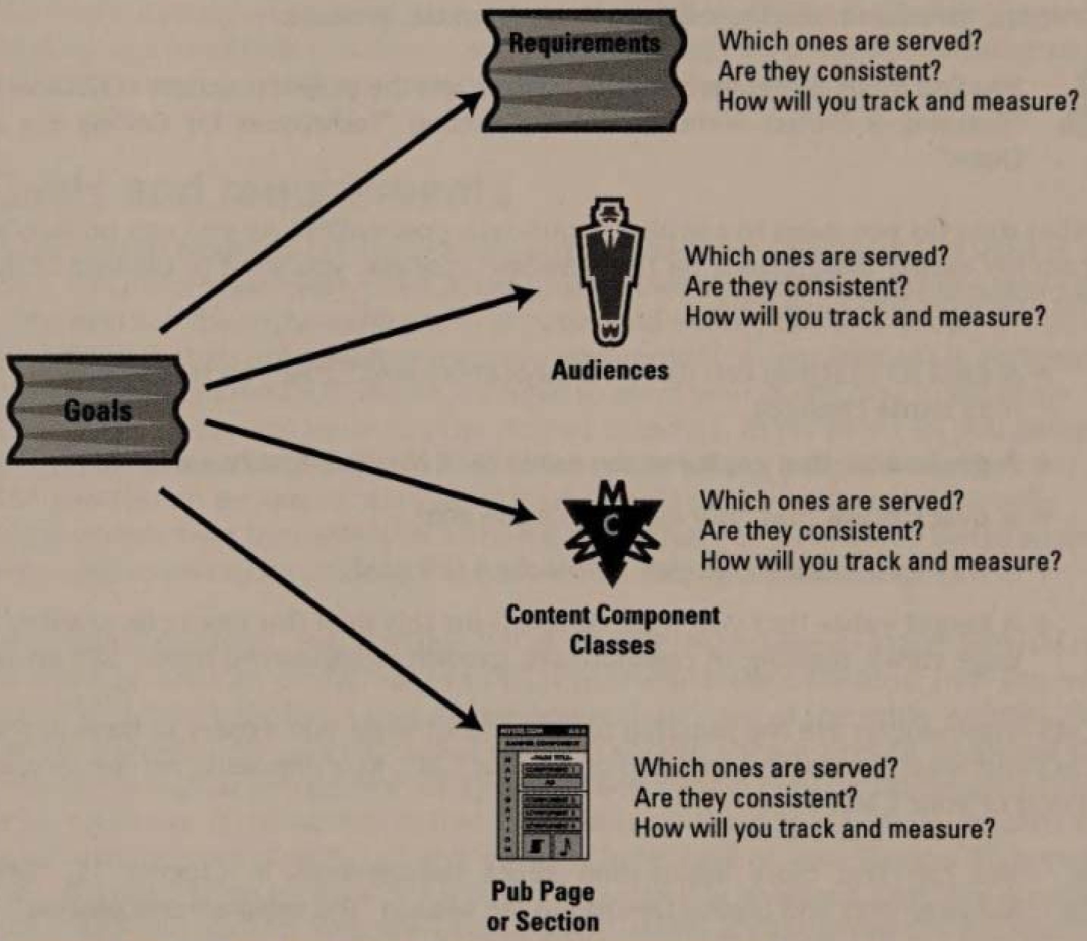

Requirements
Which ones are served?
Are they consistent?
How will you track and measure?
Which ones are served?
Are they consistent?
How will you track and measure?
Audiences
Goals
Which ones are served?
Are they consistent?
How will you track and measure?
Content Component Classes
Which ones are served?
Are they consistent?
How will you track and measure?
Pub Page or Section

Figure 19-3: Goals relate most directly to requirements, audiences, components, and publications.

The relationships that you want to make sure that you cover are those among the audiences, components, and publications. For each of these relationships, answering the following few questions can ensure you that your system is in balance:

- **Which goal does it serve?** Each requirement, audience, component, and publication ought to directly serve at least one goal of the system. You want naturally to prefer those entities that serve more than one goal. On the other hand, be very suspicious of any that serve no goal or for which figuring out a goal is a stretch.

- **Are inconsistencies present?** Do any of the entities directly conflict with a goal? Are any in conflict with each other, given a particular goal? If your goal is to cut down support costs, for example, does a Wireless Application Protocol (WAP) version of your Web site drive costs up? If so, which is more important?
- **How can you track and measure your goals?** Extend your goal-measurement criteria out to the other entities to give more substance and detail to the measurements. If you need to drive up the conversion rate (the percentage of those interested who ultimately buy) from 10 to 15 percent, for example, you must determine from which audiences the increase comes. Which components are key? Through which publications does the increase occur?

In addition to correlating entity by entity, look at the entire system. Do the most important goals get the most coverage in the other entities? Do the goals suggest entities that you never thought to include before? Do the entities compel you to revisit and perhaps modify your goals or, especially, your success metrics? Overall, does serving these audiences, this content, or these publication channels seem to ensure that your organization is going to see the benefits that you promise in the goals?

Tip

If you want to go farther, you can bring your sponsors into the picture as an entity by cross-analyzing sponsors and goals, requirements, publications, and components. Ask how well the sponsors understand and support the other entities.

Audiences

Audience entities contain information on the constituencies to which the CMS provides content. The basic information to collect on your audiences is as follows:

- **Identification**, including an ID and name for each audience group.
- **Demographics**, which describe the personal data and habits of the audience.
- **Attitudes**, which tell you what responses to expect from the audience.
- **Comparisons**, which tell you what other publications the audience is likely to compare to yours.
- **Profiles**, which combine all the information that you gather on an audience into a set of measurable and storable traits.

Cross-Reference

You can find more detail about the preceding items in Chapter 21, "Cataloging Audiences," in the section "Analyzing audiences."

As you may expect, audiences relate in some way to all CM entities. Their most important relationship, however, you can map to the following CM entities, as shown in Figure 19-4:

- **Are inconsistencies present?** Do any of the entities directly conflict with a goal? Are any in conflict with each other, given a particular goal? If your goal is to cut down support costs, for example, does a Wireless Application Protocol (WAP) version of your Web site drive costs up? If so, which is more important?
- **How can you track and measure your goals?** Extend your goal-measurement criteria out to the other entities to give more substance and detail to the measurements. If you need to drive up the conversion rate (the percentage of those interested who ultimately buy) from 10 to 15 percent, for example, you must determine from which audiences the increase comes. Which components are key? Through which publications does the increase occur?

In addition to correlating entity by entity, look at the entire system. Do the most important goals get the most coverage in the other entities? Do the goals suggest entities that you never thought to include before? Do the entities compel you to revisit and perhaps modify your goals or, especially, your success metrics? Overall, does serving these audiences, this content, or these publication channels seem to ensure that your organization is going to see the benefits that you promise in the goals?

Tip

If you want to go farther, you can bring your sponsors into the picture as an entity by cross-analyzing sponsors and goals, requirements, publications, and components. Ask how well the sponsors understand and support the other entities.

Audiences

Audience entities contain information on the constituencies to which the CMS provides content. The basic information to collect on your audiences is as follows:

- **Identification**, including an ID and name for each audience group.
- **Demographics**, which describe the personal data and habits of the audience.
- **Attitudes**, which tell you what responses to expect from the audience.
- **Comparisons**, which tell you what other publications the audience is likely to compare to yours.
- **Profiles**, which combine all the information that you gather on an audience into a set of measurable and storable traits.

Cross-Reference

You can find more detail about the preceding items in Chapter 21, "Cataloging Audiences," in the section "Analyzing audiences."

As you may expect, audiences relate in some way to all CM entities. Their most important relationship, however, you can map to the following CM entities, as shown in Figure 19-4:

- **Goals and requirements** relate to audiences as I describe in the section "Goals and Requirements," earlier in this chapter.
- **Publications** obviously relate to audiences the way that they always have. You must decide what the audience is for each publication. Then you need to know what your audiences expect for structure and format in a publication such as yours and what other publications they're likely to compare yours to. In addition, you need to do a lot of work to figure out exactly how to personalize a publication for the various audiences that it's to serve.
- **Component classes** relate to audiences in that you target your components to particular audiences as the components aggregate into your CMS. If a component serves no audience, it doesn't belong in your system. The most (or best) content needs to apply to the most important audiences. Audiences may have expectations about the structure or format of a particular kind of content, based on past experience. In addition, they assign certain values or priorities to the different kinds of content that you provide, and you want to try to assess these values and priorities beforehand to better know how to order and select components for particular audiences. Finally, you need to determine how to represent the different audiences that a component must serve in the structure of the component class. Do you need special component elements, for example, that recast some of the element's content for particular audiences?
- **Authors** produce content for audiences. Figuring out which authors are going to write for which audiences is essential (as is making the authors aware of what you decide). Do these authors understand the audiences that they're writing for? Can the authors communicate well with their audiences? Because authoring is an expensive endeavor, are you spending your limited authoring budget on the most important audiences? Do the audiences respect the authors that you're providing? Do your authors have credibility with the audiences they're writing for?
- **Acquisition sources** are similar to authors in their relationship to audiences. In fact, an audience may not know or care whether you author or acquire the information that you show them. On the other hand, if they perceive that you're just borrowing a lot of generic information, they're likely to value it significantly less. So you must determine the perceived value of each source for the audiences that you intend it to serve. Does an audience respond better if you cite the source name or if you leave it out? How close is the original audience for a source to those for whom you intend it? Do you need to change (reword, add, delete) anything about the content to accommodate the needs and characteristics of your audiences?

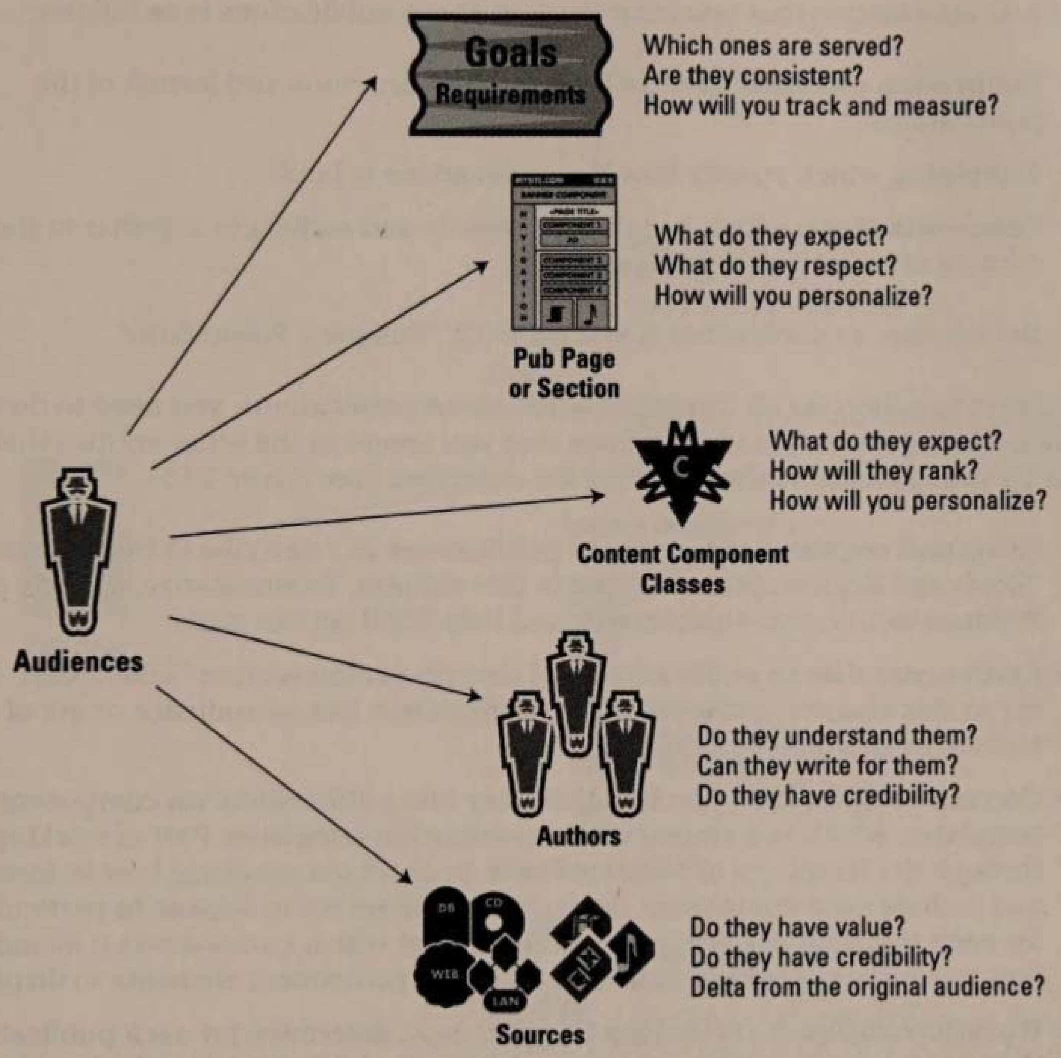

Goals
Requirements
Which ones are served?
Are they consistent?
How will you track and measure?
What do they expect?
What do they respect?
How will you personalize?
Pub Page
or Section
What do they expect?
How will they rank?
How will you personalize?
Content Component
Classes
Audiences
Do they understand them?
Can they write for them?
Do they have credibility?
Authors
Do they have value?
Do they have credibility?
Delta from the original audience?
Sources

Figure 19-4: Audiences relate in a big way to most of the major content management entities.

Overall, always ask yourself, "How are my audiences likely to perceive this information? If you ever come up with the answer, "Not well," you need to do some rethinking.

Cross-Reference Many of the issues that I raise in the preceding list I cover again in Chapter 21, "Cataloging Audiences."

Publications

Publication entities contain information about the publications that you want to create, how you plan to personalize them, and how you're going to create them from the CMS by using templates.

The basic information that you must capture about publications is as follows:

- **Publication qualities**, such as the purpose, structure, and format of the publications.
- **Templates**, which specify how the publications is built.
- **Personalizations**, which match components and audiences together in the context of a particular publication.

Cross-Reference The full story on publications is in Chapter 22, "Designing Publications."

In addition to collecting all this information about publications, you need to do a lot of work relating the publication entities that you create to the other entities that make up your system, as the following list describes (see Figure 19-5):

- **Goals and requirements** relate to publications as I describe in the section "Goals and Requirements," earlier in this chapter. To summarize, specific publications must meet requirements and help fulfill certain goals.
- **Audiences** relate to publications as I describe in the section "Audiences," earlier in this chapter. Obviously, every publication has an audience or set of audiences to which you direct it.
- **Content components** must find their way into publications via component templates, which you embed within publication templates. Part of working through the templates of your system is to chart out precisely how to format and include each component on each page where it's to appear. In particular, for each publication page, you must figure out which components it includes, how to format these components, and which component elements to display.
- **Workflow and staff** are entities that you must determine for each publication that you create. You may need any number of workflows. In particular, for each publication, you must determine which tasks are necessary and then how to group them by person into jobs and by time into workflows.
- **Access structures** are another entity that you must determine for each publication. You must map between the general access structures that you store in the CMS repository and the navigation that you produce to move users through the publication. In particular, you need to decide how to transform the hierarchy, indexes, cross references, and sequences that apply from component to component in the repository into page or section navigation after you create a publication.

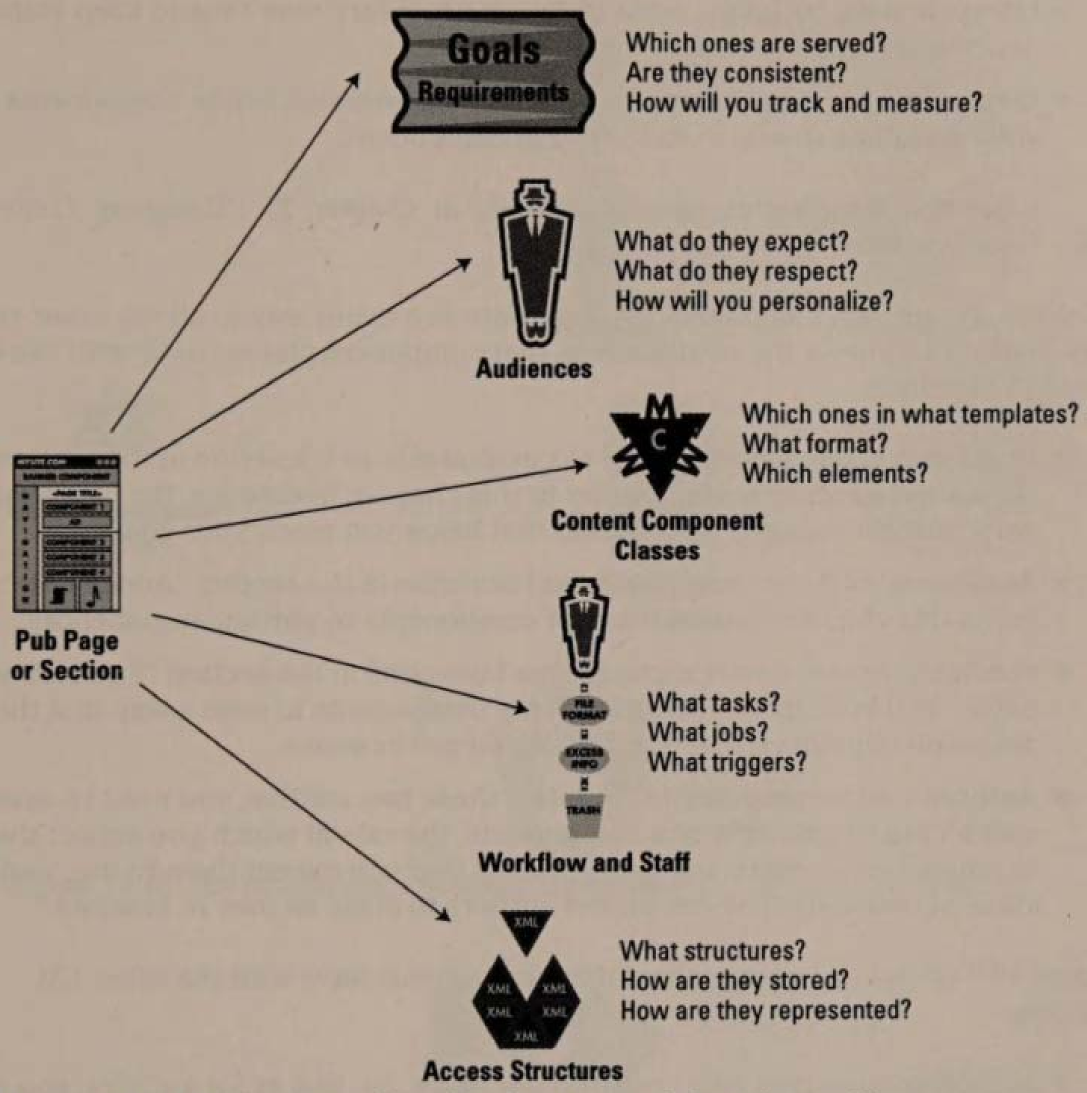

Goals
Requirements
Which ones are served?
Are they consistent?
How will you track and measure?
What do they expect?
What do they respect?
How will you personalize?
Audiences
M
C
Which ones in what templates?
What format?
Which elements?
Content Component Classes
Pub Page or Section
FILE FORMAT
EXCESS INFO
TRASH
What tasks?
What jobs?
What triggers?
Workflow and Staff
XML
What structures?
How are they stored?
How are they represented?
Access Structures

Figure 19-5: Publications relate directly to most of the other entities.

Component classes

Component class entities contain information about all the content that you collect and manage. The main areas of information that component entities contain are as follows:

- **Identification**, including the names, IDs, and overall priorities of the components in your system.
- **Creation**, where you figure out exactly how to collect components of each type and aggregate them into the CMS.
- **Quantities**, where you figure out how many of each type of component that you intend to gather initially and over time.

- **Lifecycle data**, including what tasks are necessary over time to keep components up to date and, finally, retire them.
- **Elements**, including the way to divide the information inside components and what metadata to add to each type of component.

Cross-Reference I describe components more completely in Chapter 23, "Designing Content Components."

As you may suspect, component classes relate in a major way to all the other entities. Figure 19-6 shows the relationships that component classes have with the following CM entities:

- **Goals and requirements** relate to components as I describe in the section "Goals and Requirements," earlier in this chapter. In essence, the components must contain required information that helps you reach your goals.
- **Audiences** relate to components as I describe in the section "Audiences," earlier in this chapter. You must target components to particular audiences.
- **Publications** relate to components as I describe in the section "Publications," earlier in this chapter. You must create components in such a way that they're automatically selected and laid into your publications.
- **Authors** create components. To relate these two entities, you need to assess which authors create which components, the rate at which you expect them to create components, the kinds of tools that you expect them to use, and the kinds of metadata that you expect authors to enter as they're creating.

Figure 19-7 shows the relationships that components have with the other CM entities:

- **Acquisition sources** also create components. So, just as for authors, you must figure out which sources create which components and at what rate. In addition, you need to assess how well marked components are in each source (that is, can you get a program to pull out components automatically?). Finally, you must suss out and account for any editorial or metatorial changes that you must make to each source to yield its target components.
- **Workflow and staffing** are entities that you must determine for the creation and subsequent life cycle of content components. You must arrive at a set of tasks, jobs, and triggers that you use to manage components.
- **Access structures** relate to components in a deceptively simple way. Access structures often become metadata elements and values in a content component. Your components may have an element that you call *Keywords*, for example, which includes the set of index terms under which you index that component. Assigning a component a Keywords element is the easy part. Filling in the keywords for thousands of components is the hard part. In addition, you must figure out how to store your various access structures in the CMS repository and display them in its interface.

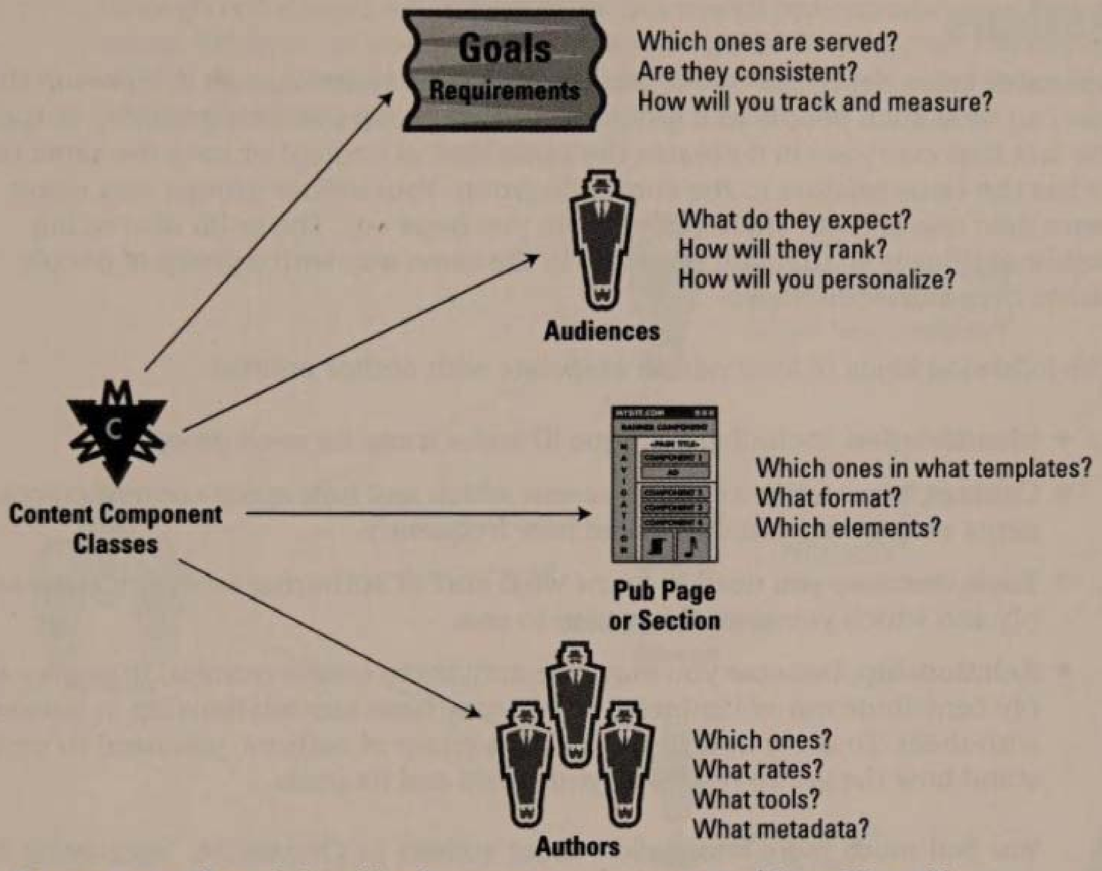

Goals
Requirements
Which ones are served?
Are they consistent?
How will you track and measure?
What do they expect?
How will they rank?
How will you personalize?
Audiences
Content Component Classes
Which ones in what templates?
What format?
Which elements?
Pub Page or Section
Which ones?
What rates?
What tools?
What metadata?
Authors

Figure 19-6: The relationships between components and four CM entities

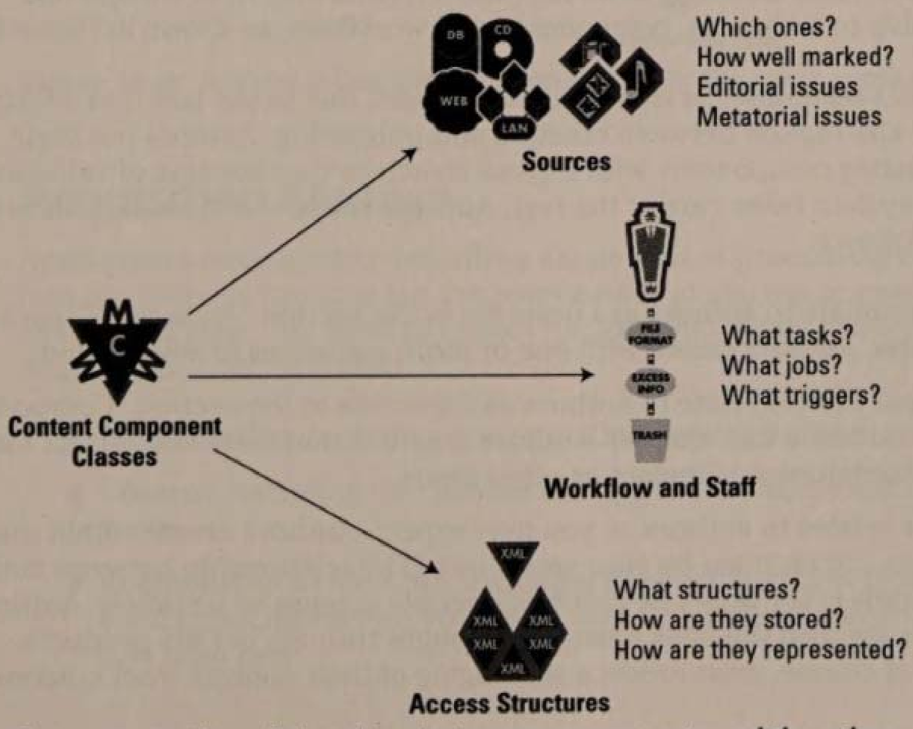

DB
CD
WEB
LAN
Which ones?
How well marked?
Editorial issues
Metatorial issues
Sources
Content Component Classes
FILE FORMAT
EXCESS INFO
TRASH
What tasks?
What jobs?
What triggers?
Workflow and Staff
XML
What structures?
How are they stored?
How are they represented?
Access Structures

Figure 19-7: The relationships between components and the other three CM entities

Authors

An *author entity* describes one or more people who share enough in common that you can treat such people as a group. An author group can be a group by virtue of the fact that everyone in it creates the same kind of content or uses the same tools or has the same relation to the core CMS group. Your author groups may share more than one of these traits (or, at least, you hope so). The point of creating author entities is so that you can work in the same way with a group of people rather than with individuals.

The following kinds of information associate with author entities:

- **Identification**, including a unique ID and a name for each group.
- **Content**, because you need to assess which and how many content components each group can create and how frequently.
- **Tools**, because you need to know what sort of authoring tools you must supply and which you want this group to use.
- **Relationship**, because you may hire authors to create content; they may simply contribute out of kindness; or you may have any relationship in between with them. To know how to work with a group of authors, you need to understand how the group relates to your team and its goals.

Cross-Reference

You find much more information about authors in Chapter 24, "Accounting for Authors."

Although authors relate indirectly to all the other entities, they have a major and direct relationship to audiences, components, and workflows, as shown in Figure 19-8.

The lack of direct relationships is, as you may expect, due to the fact that a CMS adds a layer of abstraction between creation and publishing. Authors put their energy into creating components with a given structure that become of value to an audience. The system takes care of the rest. Authors relate to the entities shown in Figure 19-8 as follows:

- **Audiences** relate to authors as I describe in the section "Audiences," earlier in this chapter. Authors create with one or more audiences firmly in mind.
- **Component classes** relate to authors as I describe in the section "Component classes," earlier in this chapter. Authors create content as components rather than as standalone documents or other items.
- **Workflow** relates to authors as you may expect. Authors create within the constraints put on them by your workflows. The relationship between authors and workflow is the only one that most people discuss as workflow. Author workflows are also the ones most well thought through in CMS products. Authors, of course, must follow a set staging of their content from creation

through review and into production. But workflows can associate with every entity. Publication workflows, for example, guide the creation and deployment of Web sites. Component workflows determine when you need to archive and review components.

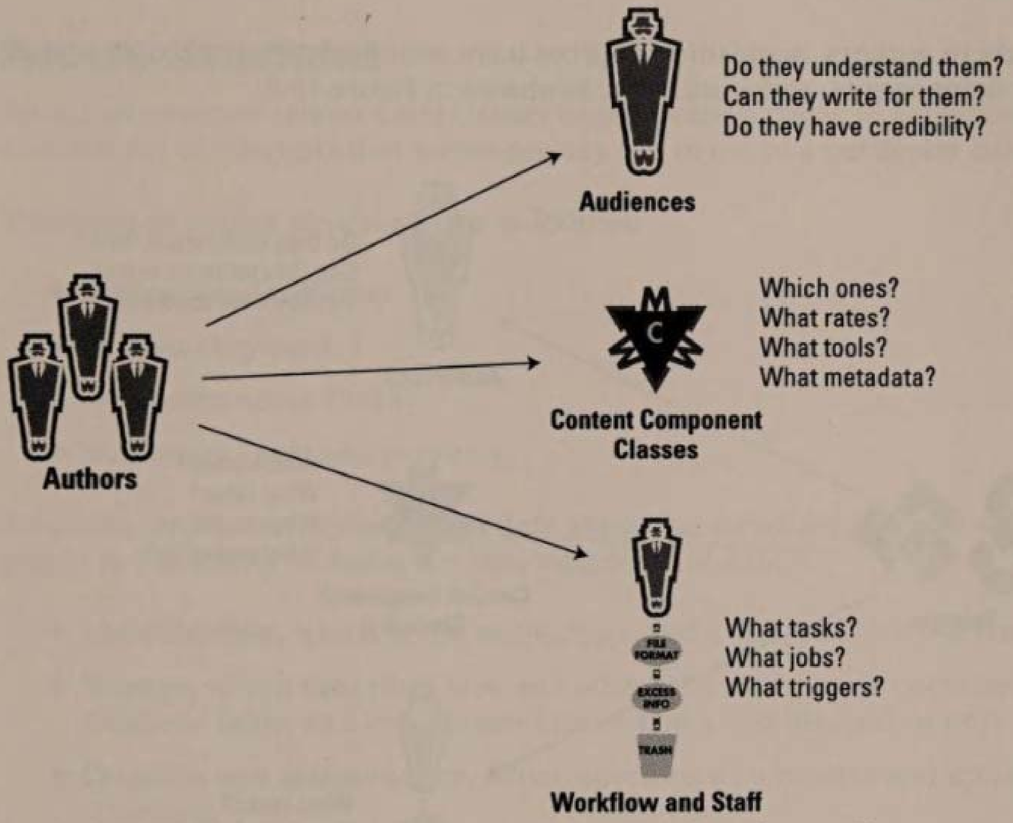

Do they understand them?
Can they write for them?
Do they have credibility?
Audiences
Which ones?
What rates?
What tools?
What metadata?
Content Component Classes
Authors
What tasks?
What jobs?
What triggers?
FILE FORMAT
EXCESS INFO
TRASH
Workflow and Staff

Figure 19-8: Authors relate directly to only three of the other entities.

Acquisition sources

Acquisition source entities describe a single kind of preexisting content that feeds into the system. You may tap the source once at startup or consistently over time as the CMS runs. You need to collect the following kinds of information about your acquisition source entities:

- **Identification**, including a unique ID and name for each source.
- **Content**, including the number and types of components that the source yields.
- **Relationship to the CMS**, including the person in charge of the acquisition source, the agreement you reach with her, and the way that you get information from her.

- **Conversion**, including an analysis of how to distinguish each component within the source and how to locate and tag component elements in the source information.

Cross-Reference Chapter 25, "Accounting for Acquisition Sources," contains the full story on acquisition sources.

Similarly to authors, acquisition sources have major relationships only to audiences, components, and workflows, as shown in Figure 19-9.

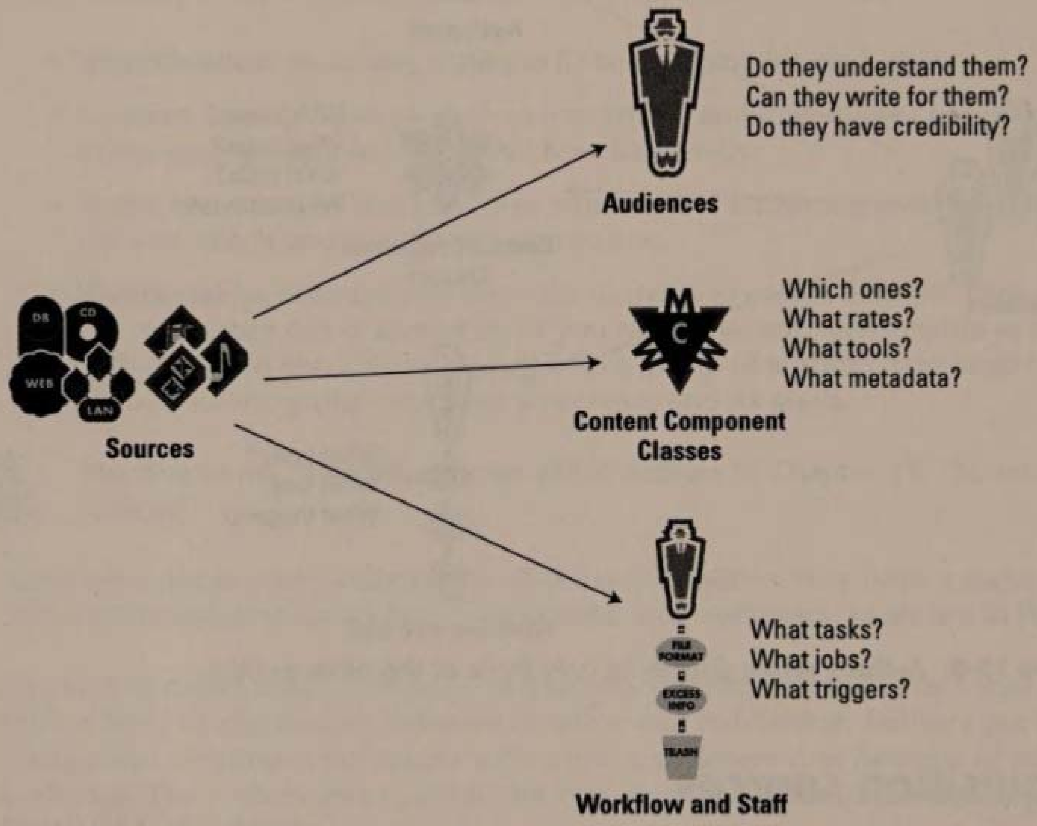

Do they understand them?
Can they write for them?
Do they have credibility?
Audiences
DB
CD
WEB
LAN
Sources
Which ones?
What rates?
What tools?
What metadata?
Content Component Classes
FILE FORMAT
EXCESS INFO
TRASH
What tasks?
What jobs?
What triggers?
Workflow and Staff

Figure 19-9: Sources relate most directly to audiences, components, and workflows.

Components insulate acquisition sources from most of the other entities (as they do authors). After the source transforms into components with a complete set of well-produced elements, its job is done. The components then enter into relationships with the structures and publications of the wider CMS, as follows:

- **Audiences** relate to acquisition sources as I describe in the section "Audiences," earlier in this chapter. You must make sure that the source is appropriate to your audiences or that you can change it to make it so.
- **Component classes** relate to acquisition sources as I describe in the section "Component classes," earlier in this chapter. You must figure out how to chop each source into components and elements.

- **Workflow** you can apply to acquisition sources prior to the CMS. A conversion team, for example, may perform an extensive set of staged tasks on the source before anyone logs any content into the CMS repository. I don't mean here that the CMS workflow tools can't extend into the conversion process — just that they probably don't.

Access structures

An *access structure* relates components or publication pages to each other or to an external set of concepts that someone may use to get to a particular component.

The types of access structures are as follows:

- Hierarchies (outlines)
- Indexes (keywords)
- Cross-references (links)
- Sequences (next and previous)

An *access structure entity* is a set of data about the structure and how you intend to create it. The entity includes the following sorts of data:

- **Identification,** which is the name, type, and a unique ID for the structure.
- **Storage,** which describes how and where the structure is contained (in a database table, as a component element, in a text file, and so on).
- **Creation and maintenance,** which specifies who creates and updates the structure.

Cross-Reference

Chapter 26, "Designing Content Access Structures," provides the complete scoop on access structures.

Access structures apply to components in the CMS repository and publications that the CMS produces. Thus the following entities are the only two others to which access structures have a direct relationship, as shown in Figure 19-10:

- **Component classes** relate to access structures as I describe in the section "Component classes," earlier in this chapter. In the case of components, access structures become metadata.
- **Publications** relate to access structures as I describe in the section "Audiences," earlier in this chapter. In the case of publications, access structures become navigation for finding and moving to the content that you want to view.

In either case, you must decide what structures to support, how you intend to store them in the CMS repository (or perhaps in publication templates), and how to represent them to the user so that she can best access content.

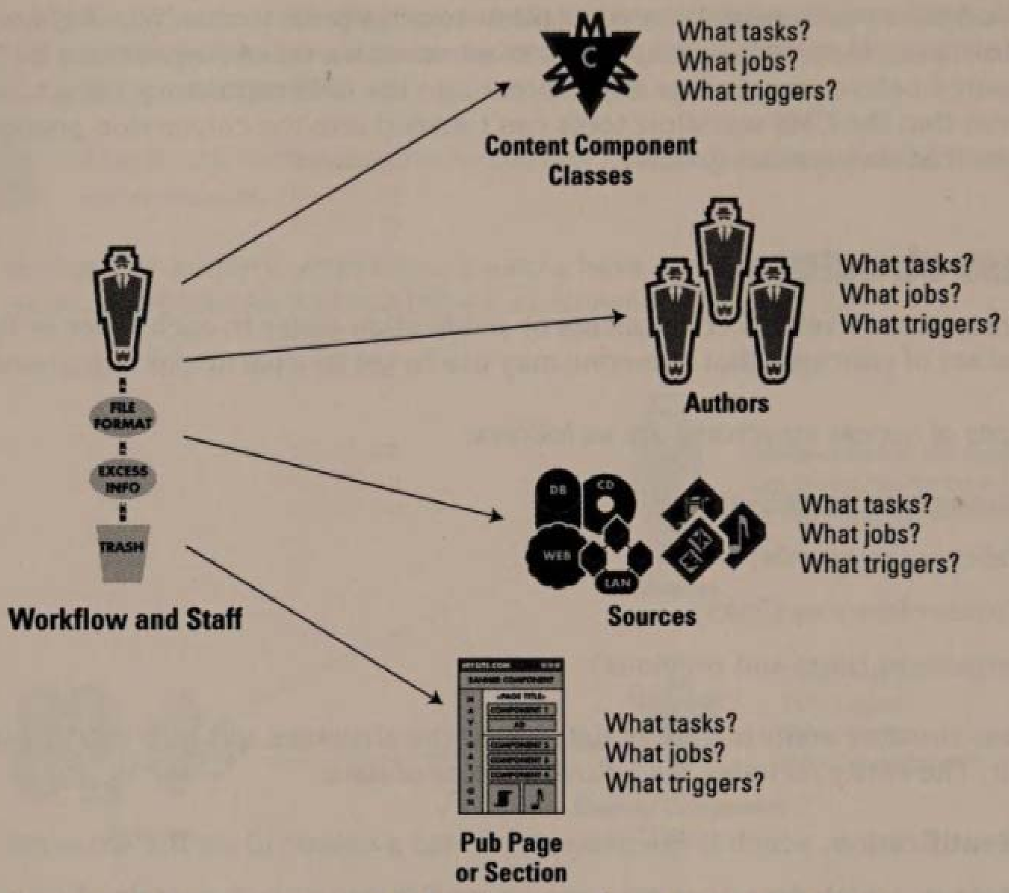

What tasks?
What jobs?
What triggers?
Content Component Classes
What tasks?
What jobs?
What triggers?
Authors
FILE FORMAT
EXCESS INFO
TRASH
What tasks?
What jobs?
What triggers?
Workflow and Staff
Sources
What tasks?
What jobs?
What triggers?
Pub Page or Section

Figure 19-10: Components and publications have access structures.

Workflow and staffing

A *workflow*, as I present it in this book, is a sequence of tasks that a staff member performs and that some event triggers. A *workflow entity* is a collection of data that specifies the tasks, staff, and triggers for a particular workflow.

Workflows involve steps that form a set of sequenced operations. For each step, you must collect the following kinds of data:

- **An object**, which is the part of the CMS that the step affects.
- **A task**, which is a discrete activity that makes some change to content, publications, or the CMS structure or configuration.
- **A staff member**, who performs the tasks as part of her job.
- **A trigger**, which is an event that signals that the step is to begin.
- **A duration**, which says how long the step is to take.

Cross-Reference I present the complete model for workflow in Chapter 26, "Designing Workflow and Staffing Models."

Workflows relate to all the other entities that can serve as objects, as shown in Figure 19-11.

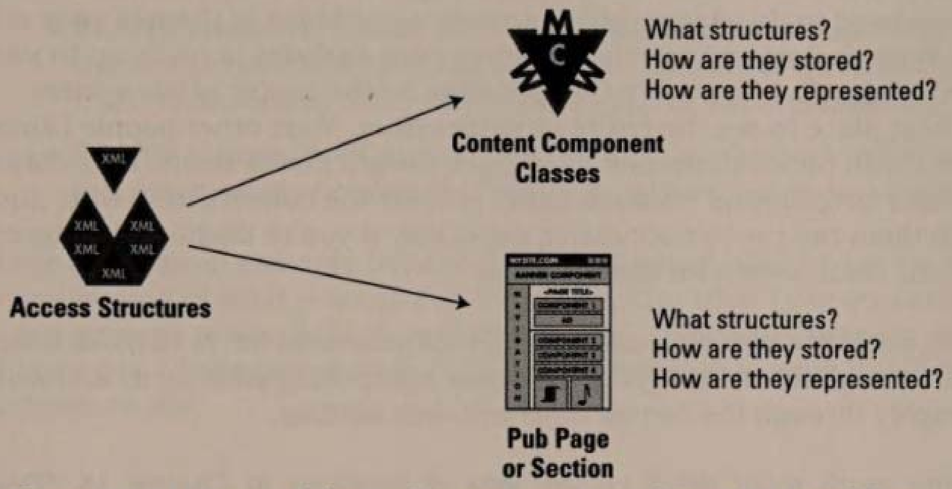

What structures?
How are they stored?
How are they represented?
Content Component Classes
XML
Access Structures
What structures?
How are they stored?
How are they represented?
Pub Page or Section

Figure 19-11: Many of the entities can serve as objects in a workflow.

The relationship between a workflow entity and any other entity is always the same. The other entity serves as the object that a workflow step changes in some way. For each step and object, you must also arrive at a task, a trigger, and a staff member who is to complete the task.

I discuss the particulars of the relationship between workflow and the other entities in the sections earlier in this chapter for each entity (authors, component classes, and so on).

Think, Plan, Integrate

In this chapter, I give you an overview of the entities that I consider most central to the design of a CMS. In addition, I hope you can see that the relationships between these entities are as important as the entities themselves in designing a system. The chapters that follow go into greater detail about each entity.

As you can probably already guess, logical design involves a sort of "chicken and egg" problem. Which comes first? Publications or the components that create them? Do you design your access structures around your components or your components around your access structures? Do you look for sources that make good components or components that the sources you want can create? As with all such processes, no single answer is right — only an ongoing process for the two sides of the issue to push on each other.

I sequence the following chapters as I do as much for convenience as for their true order. I do believe that an audience analysis is always a good first step. For one thing, you then have a set of audiences that are right for your organization, regardless of the CMS that you create. For another thing, the audiences are your ultimate judges, so you need to do all your planning with a good idea of them in your mind. After that, where to start and how to sequence your analyses is really up to you. My personal preference is to start with components. As the center of the system, they're the best place to see the entire structure from. Most other people I know prefer to start with publications and work back toward components. Publications give you a lot of tangible and visual material to start the conversation with, and starting with them can prove a lot easier, especially if you're dealing with the concept of content components for the first time.

However you decide to proceed, understand that your best bet is to make a series of iterations throughout the analysis, each time sharpening your focus and working more thoroughly through the connections between entities.

Cross-Reference

I go into much more detail on the idea of iterations in Chapter 15, "Doing Requirements and Logical Design," in the section "Iterating through your design."

To give you a head start iterating around the wheel of content management entities, I break each analysis section in the following chapters into the following subsections:

- **Think** sections get you thinking about a particular aspect of planning. The questions in these sections are working if they're really difficult to answer and blur your ideas about what you need to do. They ask you open-ended questions that tie the particular subject of analysis to the others and to the organization as a whole. After you beat these questions to death, you're ready to start planning.
- **Plan** sections detail the specific data that you must collect. The questions in these sections are the deliverable analysis that you need to do. The lists that I provide in this chapter for the content of each entity I elaborate on extensively in the Plan sections of those chapters.
- **Integrate** sections help you to relate one area of analysis to the others. The questions in these sections are working if they force you to rethink assumptions and come to better answers in Plan sections. Many of the relationships that I present in this chapter I cover again in more detail in the Integrate sections.

PLAN International — An Example Organization

PLAN International is a wonderful organization that helps children worldwide. PLAN, similar to any organization, has information and functionality that it wants to publish. I use the organization as an example throughout the following chapters to illustrate concepts and give tangible examples of the issues that I raise. Here is some information about PLAN that I got from its Web site (at `www.plan-international.org/`) to give you a flavor for the organization:

- **PLAN's identity:** PLAN is an international, humanitarian, child-focused development organization without religious, political, or governmental affiliation. Child sponsorship is the basic foundation of the organization.
- **PLAN's vision:** PLAN's vision is of a world in which all children realize their full potential in societies that respect people's rights and dignity.

I use PLAN as an example but not as a literal one. Rather, I base my examples on a combination of what you can see on its Web site, what I can guess or make up about what it wants to accomplish, and what I need to stipulate to make a good example. Please don't believe that what I say has any relation to what PLAN believes or intends to do.

Summary

You can think of a CMS as a set of software programs running on a set of computers, or you can think of a CMS as a dynamic balance of influential entities in an organization. If you're smart, you think of a CMS in both ways. As a set of interacting entities, a CMS balances the following elements:

- The goals of the system against the realistic means by which you can accomplish those goals.
- The original sources of content (authors and acquisition sources) against the content components that they must create.
- The publications that you want to create against the components that you have available to create them and the audiences that have an interest in them.
- The staff and processes (workflows) that you can manage to put together against the labor and processing needs of the other entities.
- The access structures that you can manage to create and maintain against the navigation that your publications need and the accessibility that you want in your CMS repository.

In the next chapter, I deal directly with the thing that holds this balance together—metadata.

✦ ✦ ✦

CHAPTER 20

Working with Metadata

◆ ◆ ◆ ◆

In This Chapter

The meaning of meta, metadata, and metatorial

A way to categorize the various kinds of metadata

How to collect and present metadata in a CMS

The metatorial framework, metatorial processing, and the job of metator

Issues in metadata localization

◆ ◆ ◆ ◆

Metadata is the small snippets of information (or data) that you attach to content so that you can more easily catalog, store, and retrieve it. A coherent system of metadata draws diverse classes of content into a coherent scheme in which content components relate to each other as well as to the collection, management, and publishing systems that you devise.

In this chapter, I dive into the concept of metadata, discussing its meaning, types, and uses.

What Is Metadata?

More than anything else, the metadata system behind a CMS is what defines the system. The set of names and relationships that a metatorial framework contains are the skeleton on which you hang the content. Without this structure, the content is as formless and flaccid as a body without bones.

What does meta mean?

The word *meta* itself has no meaning except (according to the Oxford English Dictionary) as a column that marks the boundary of a circus. In more general usage, *meta* is a *prefix*. It modifies the meaning of the words that it precedes.

Merriam-Webster OnLine (at `www.m-w.com`) states that, as a prefix, *meta* means the following:

- Occurring later than or in succession to.
- Situated behind or beyond.
- Change or transformation.
- Later or more highly organized or specialized form of.
- More comprehensive or transcending.

All these definitions give depth to the idea of meta that I have in mind in this book. Meta-things come after the things themselves to add context and organization to them. They go beyond the things themselves to a higher level of abstraction, where you see the things in a wider and different light.

I first became enamored by the concept of meta while reading Douglas Hofstadter's book, *Godel, Escher, Bach: An Eternal Golden Braid.* This most wonderful work uses meta as one of its most central and subtle threads. In the following dialog, Achilles (a character Hofstadter borrowed from the Greek philosopher Zeno) meets GOD who turns out to be a never-ending hierarchy of genies::

> *Achilles:* Tell me, Mr. Genie—what is a meta-wish?
>
> *Genie:* It's simply a wish about wishes. I am not allowed to grant meta-wishes. It's only in my purview to grant plain ordinary wishes, such as wishing for ten bottles of beer, to have Helen of Troy on a blanket, or to have an all-expense-paid weekend for two at the Copacabana. You know—simple things like that. But meta-wishes I cannot grant, GOD won't permit me to.

You must read the book to find out exactly who GOD is and get Hofstadter's deep and wide view of metaphenomena. For the present, I need take only one word out of his work—*about.*

To my mind, the best way to understand the prefix *meta* is to substitute the word *about.* The word *metadialog*, for example, is a dialog about a dialog. A *metatorial* (not my usage of this word, but another common one) is an editorial opinion about editorial opinions. *Metadata* is data about data.

The word *about* captures the large majority of the dictionary definitions of meta. If you're talking about a thing, you must be coming after the thing, and you must be speaking at a transcending or higher level of organization than that of the thing itself. Most important, by talking about the thing, you're changing its meaning. My recent favorite, which I use on every new kid I meet, is a statement that is its own metastatement: "Everything that I say is a lie!"

Note You may be familiar with the word *meta* from HTML where you have a `<META>` tag. This tag is now a catch-all for any information that people want to put in it, but its intended use is quite consistent with its name. You're supposed to use the `<META>` tag to list information *about* the HTML file.

What does metadata mean?

Just saying that metadata is data about data isn't much of a description. The term is in wide enough usage these days to have a lot more behind it.

You can get a fair feeling for how people use the word by looking at the following few sentences from the home page of the Metadata Coalition at `www.mdcinfo.com/`:

> "The Coalition allies software vendors and users with a common purpose of driving forward the definition, implementation, and ongoing evolution of a metadata interchange format standard and its support mechanisms. The need for such standards arises as metadata, or the information about the enterprise data emerges as a critical element in effective data management. Different tools, including data warehousing, distributed client/server computing, databases (relational, OLAP, OLTP . . .), integrated enterprise-wide applications, etc. . . . must be able to cooperate and make use of metadata generated by each other."

This excerpt, one among many that you can find on the Web, shows the main characteristics of metadata as people most commonly use the term, and the following list describes those characteristics:

- **Sharing:** Metadata provides the capability to share data across applications. By employing data that describes the use and meaning of the data that it surrounds, one system can interpret and translate the data that it receives from another. In a content management context, metadata enables publications that each need a somewhat different form of the same data to draw from a common repository.
- **Standards:** Metadata is a set of standards that groups agree to for information definitions. Standards, which are the basis of any kind of data sharing, bring the possibility of large-scale efficiencies in information interchange among groups that don't even know one another. In the content management context, the standards may be mostly internal today, but they serve the same purpose. Standards ensure that others can automatically reuse the efforts of one person or group if they all follow the same standards.

- **A focus on databases:** The main reason today for an interest in metadata is the sharing and standards behind "standard" database applications. Data warehousing and interapplication data transfer are huge concerns for organizations with an enormous amount of data trapped in databases and other files that no one can interpret except by using the application that created them.
- **An awareness of the wider world:** Today, metadata is mostly used in data systems. A vague but growing understanding exists, however, that metadata isn't just for data. The previous quote from the Metadata Coalition mentions "integrated enterprise-wide applications." A CMS is an integrated enterprise-wide application. Metadata is critical, not only so that a CMS can integrate with other enterprise data sources, but so that the CMS itself can unify and make the best automated use of the information and functionality that it manages.

Note

By the way, metadata, apparently, is a registered trademark of the Metadata Company, headquartered in Long Beach, California (and at `www.metadata.com/` on the Web).

Merging this information with the definition of the prefix *meta* can give you a deeper understanding of what metadata is.

First, metadata is what you need other than the data itself to understand and use that data. Metadata acts as the instructions that come with the data. In addition, metadata is what isn't there if you look at the data itself. Metadata exists in addition to or after the data. It adds context and a wider interpretation to the data. Consider, for example, the following piece of content that has no metadata:

```
Cinder Riley is the anti-Cinderella. She wants nothing to do
with fame and riches and longs to run away with a pauper . . .
```

Without any other information surrounding this content, knowing what to make of it or what you may use it for is hard. By adding some metadata, however, its meaning and use becomes clear, as the following table shows.

Metadata element	*Value*
Author	Claire Taylor
Publication date	January 1968
Source	Plays Magazine
Reviewer	Bernard Lewis
Audience	Children, primary school teachers, and librarians
Title	The Adventures of Cinder Riley
Review	Cinder Riley is the anti-Cinderella. She wants nothing to do with fame and riches and longs to run away with a pauper . . .

Now you can tell that the content is a review of a play called "The Adventures of Cinder Riley." All that extra information isn't the content; it surrounds the content and tells you *about* it. In addition, the chunk of text alone is hard to make use of. But add metadata and the chunk becomes part of a wider system of shared attributes that help to store, locate, and present it in the context of the other content that you may have.

The metadata isn't the content. I make this claim because it exists apart from the content. Contrast the preceding table with the following representation of the same basic information:

In January 1968, Claire Taylor published "The Adventures of Cinder Riley" in *Plays Magazine*. Bernard Lewis provided this review of the play, which is of most interest to children, primary school teachers, and librarians. Cinder Riley is the anti-Cinderella. She wants nothing to do with fame and riches and longs to run away with a pauper . . .

In the table form, *Claire Taylor* is a piece of metadata. In the paragraph form, it's not. What makes *Claire Taylor* metadata not only is the fact that it's descriptive of some other content, but also that it's separate from that content. As a content manager, you choose what information to separate out and make into metadata and what remains part of the content. This distinction is a practical, not a philosophical one. If you use the words *Claire Taylor* as metadata, you must somehow name and separate it so that a computer can find it and use it to categorize the content that it surrounds.

As a final point, notice that nothing is exclusive about putting metadata in a table that extracts it from the content that it surrounds. The following paragraph, for example, uses XML tags to distinguish metadata from the content it's embedded in:

In `< PublicationDate>`January 1968`</PublicationDate>`, `<Author>`Claire Taylor`</Author>` published `<Title>`The Adventures of Cinder Riley`</Title>` in `<Source>`Plays Magazine `</Source>`. `<Reviewer>`Bernard Lewis`<Reviewer>` provided this review of the play, which is of most interest to `<Audience>`children, primary school teachers, and librarians`</Audience>`. `<Review>`Cinder Riley is the anti-Cinderella. She wants nothing to do with fame and riches and longs to run away with a pauper . . .`</Review>`

What does metatorial mean?

Over the years during which I've been building electronic publication systems, I've hit again and again on the problem of metadata. At first, I didn't think of metadata as a problem unto itself. Rather, I confronted related issues such as, "How do I know what content is ready for an audience to see?" or, more widely, "How do I pack a bunch of data in with the content, knowing that this data isn't going to appear but that I need it to manage the production and display of the content?" Over time, I began to standardize the ways that I dealt with this "bunch of data" so that it became easier and easier to find, parse, and manipulate. For some time, I used a "dot" system, where each line of the file that I needed to process began with a period and the name of the data field. (See Chapter 25, "Accounting for Acquisition Sources," in the section "Syndicated content," for an example of a "dot" system.) The systems of standardized tagging that I was working with made isolating metadata from the information it surrounded and doing what I needed to do with it (usually find it and load it into database fields) easy. At about the same time (I would guess about 1996), the term *metadata* became fashionable in the electronic publications arena as a way to describe "data about data."

As is apt to happen, over time I began to notice some patterns, as the following list describes:

- **Metadata consistently fell into a few distinct categories.** These categories include *navigational* (TOC, index, cross reference, and browse), *management* (author, create date, last edit, and so on.), *content type* (what I now call *components*), *internal structure* (title, abstract, body, and so on), and *inclusion* (add media here, add a banner here, add standard text here, and so on).
- **Without a rigorous consistency and careful attention, metadata becomes useless.** What good is sometimes including a chunk of standard text and sometimes typing it in from scratch? How useful is a TOC if every submitter has a different notion of how organize content and where it belongs? (You can guarantee that every submitter's own content always comes out on top!) Someone or, better, some system is necessary to ensure that people handle metadata thoroughly and consistently.
- **A different set of skills is necessary to deal with this metadata.** Authors, who are, generally, subject-matter experts and not experts at the system for managing their expertise, are often unable to add this extra data. One very fundamental piece of metadata, for example, concerns where, in the overall outline of content, a particular piece of information belongs? In larger systems, authors who have no problem creating content often don't know enough to decide where to put it. The problem is worse for cross-references. Authors rarely have the wherewithal to discover what else others are submitting and exactly how to relate it to what they submit. Overall, you need a type of person who's a cross between an editor, a librarian, and a database administrator to do a good job creating and maintaining metadata.
- **To do metadata well requires a lot of human energy.** Simply writing some scripts and pulling out what's easy to find isn't sufficient. A perfect example of this idea is keywording. I've seen many systems fail miserably at indexing because people thought that all they needed to do was create a program to find and mark every example of a small set of words and phrases.

began to standardize the ways that I dealt with this "bunch of data" so that it became easier and easier to find, parse, and manipulate. For some time, I used a "dot" system, where each line of the file that I needed to process began with a period and the name of the data field. (See Chapter 25, "Accounting for Acquisition Sources," in the section "Syndicated content," for an example of a "dot" system.) The systems of standardized tagging that I was working with made isolating metadata from the information it surrounded and doing what I needed to do with it (usually find it and load it into database fields) easy. At about the same time (I would guess about 1996), the term *metadata* became fashionable in the electronic publications arena as a way to describe "data about data."

As is apt to happen, over time I began to notice some patterns, as the following list describes:

- **Metadata consistently fell into a few distinct categories.** These categories include *navigational* (TOC, index, cross reference, and browse), *management* (author, create date, last edit, and so on.), *content type* (what I now call *components*), *internal structure* (title, abstract, body, and so on), and *inclusion* (add media here, add a banner here, add standard text here, and so on).
- **Without a rigorous consistency and careful attention, metadata becomes useless.** What good is sometimes including a chunk of standard text and sometimes typing it in from scratch? How useful is a TOC if every submitter has a different notion of how organize content and where it belongs? (You can guarantee that every submitter's own content always comes out on top!) Someone or, better, some system is necessary to ensure that people handle metadata thoroughly and consistently.
- **A different set of skills is necessary to deal with this metadata.** Authors, who are, generally, subject-matter experts and not experts at the system for managing their expertise, are often unable to add this extra data. One very fundamental piece of metadata, for example, concerns where, in the overall outline of content, a particular piece of information belongs? In larger systems, authors who have no problem creating content often don't know enough to decide where to put it. The problem is worse for cross-references. Authors rarely have the wherewithal to discover what else others are submitting and exactly how to relate it to what they submit. Overall, you need a type of person who's a cross between an editor, a librarian, and a database administrator to do a good job creating and maintaining metadata.
- **To do metadata well requires a lot of human energy.** Simply writing some scripts and pulling out what's easy to find isn't sufficient. A perfect example of this idea is keywording. I've seen many systems fail miserably at indexing because people thought that all they needed to do was create a program to find and mark every example of a small set of words and phrases.

With all these patterns in my head and with clients becoming ever larger and more demanding about the quality of their publications, I became creative. What's needed, I reasoned, is something like an editorial framework for metadata. It needed the right kind of leader (such as an editor), an overall metadata framework that everyone can buy into and apply, and a set of rigorous metadata guidelines (such as an editorial guide) and a metadata process that everyone must follow (such as an editorial process). Thus the *metator*, the metatorial guide, and metatorial processing were born.

As much as I like coining words, I recognize that it isn't the particular names that you choose that matter. Rather, it's the fact that you name a thing at all that's important. The most important thing for me in coming to these names was that doing so gave me a reason and a way to begin thinking about metadata systems as separate entities. Metadata became, for me, not just a part of gathering and tagging content but a subject of concern and importance on its own and apart from any particular content management process. For me, this turn represented a critical change from doing content management systems to *understanding* them.

The narrow view of metadata

The definitions of meta and metadata that I describe in the preceding section clearly point to a wide-ranging and deep notion of metadata. A more narrow view of metadata is nonetheless the most well-understood and accepted version that I've encountered. As best I can tell, this notion arose from the document-management industry and now informs most of the uses of metadata in content management products and literature. The following is my admittedly ill-informed interpretation of how the term evolved to its current usage.

First came file properties that the operating system exposed and put a user interface around. File name, size, creation date, and file type were some of the first kinds of metadata available. Although they weren't called metadata, these file properties clearly were metadata. They weren't the files themselves but rather data about the files.

After the document-management industry came along, one of the things that product companies did was to extend these file properties to include all manner of other data with which you could tag a file to describe it. Over time, people began to apply the term *metadata* to the set of properties that they could assign to a file. Document-management systems used these properties to help store, manage, and retrieve files.

Today, people still most commonly use the term *metadata* to describe a set of management data that you can use to categorize and manage complete chunks of content. In a CMS, the chunks may not be files, but the concept is the same. Metadata is administrative data about the content that enables you to manage its use.

The wide view of metadata

The narrow view of metadata corresponds best to my notion of the management component elements. *Management elements* are the administrative data (such as author, create date, and status) that you attach to a component to help you keep track of it, find it, and know what to do next with it.

Cross-Reference You can find more information about component elements in Chapter 23, "Designing Content Components," in the section "Components Have Elements."

But aren't the body elements of a component metadata? And how about the tags within an element that tell you what a piece of text is or does? Why wouldn't you call that metadata? In addition, why can't you call your access structures metadata? Don't hierarchies and cross-references offer data about the content in your system?

In fact, my view of metadata goes far beyond administrative attributes to include anything in a CMS that's data about the content you're storing. To some extent, the difference between the narrow and wide views of metadata is a consequence of a general principle of metas: *What is the thing itself and what is the meta-thing depends on your level of awareness and concern.*

For someone concerned with entire files, who never cracks those files to see what's inside, the thing itself is a file. The metathing is data about the file. In a CMS, you're concerned with smaller content chunks (which I call component elements), and you're concerned with the parts that make up the chunk (the formatting and structural tagging that lie within an element). So a wider view of metadata assumes that you need to mark entities smaller than an entire file with metadata.

A wider view also assumes that you need to manage metadata at a level larger than a file. Someone whose last task is to retrieve a file generally has no need to capture metadata on anything larger than a file. In a CMS, you need to name and track a number of things larger than files. These things are, for example, publications and access structures that include components.

I'm not saying that the narrow view is wrong and the wide view is right but to take the view that gives you the most advantage in the task that you're trying to accomplish. In a CMS, you need to take a view of metadata that's wide enough to encompass all the levels of content that you must manage.

Metadata and content management

If content management is the art of naming information (as I claim in Chapter 5, "But What is Content Really?"), metadata is the set of names. In other words, content management is all about metadata. Recall the diagram that I present first in Chapter 19, "The Wheel of Content Management." I reproduce it here for your convenience (see Figure 20-1).

Metadata forms the rim of the wheel. It holds together the entire system and ensures that it neither collapses in on itself nor expands out of control. Control is the essence of metadata in a CMS. If you name and account for everything, you're organized enough to exert complete control over your content.

Understanding the Types of Metadata

In the spirit of casting a broad metadata net, I offer the following list of the types of metadata you're likely to encounter in your travels through the domain of content management:

- **Structure metadata:** The ruling monarch of metadata. It precedes most other kinds of metadata by creating structural divisions in your content.

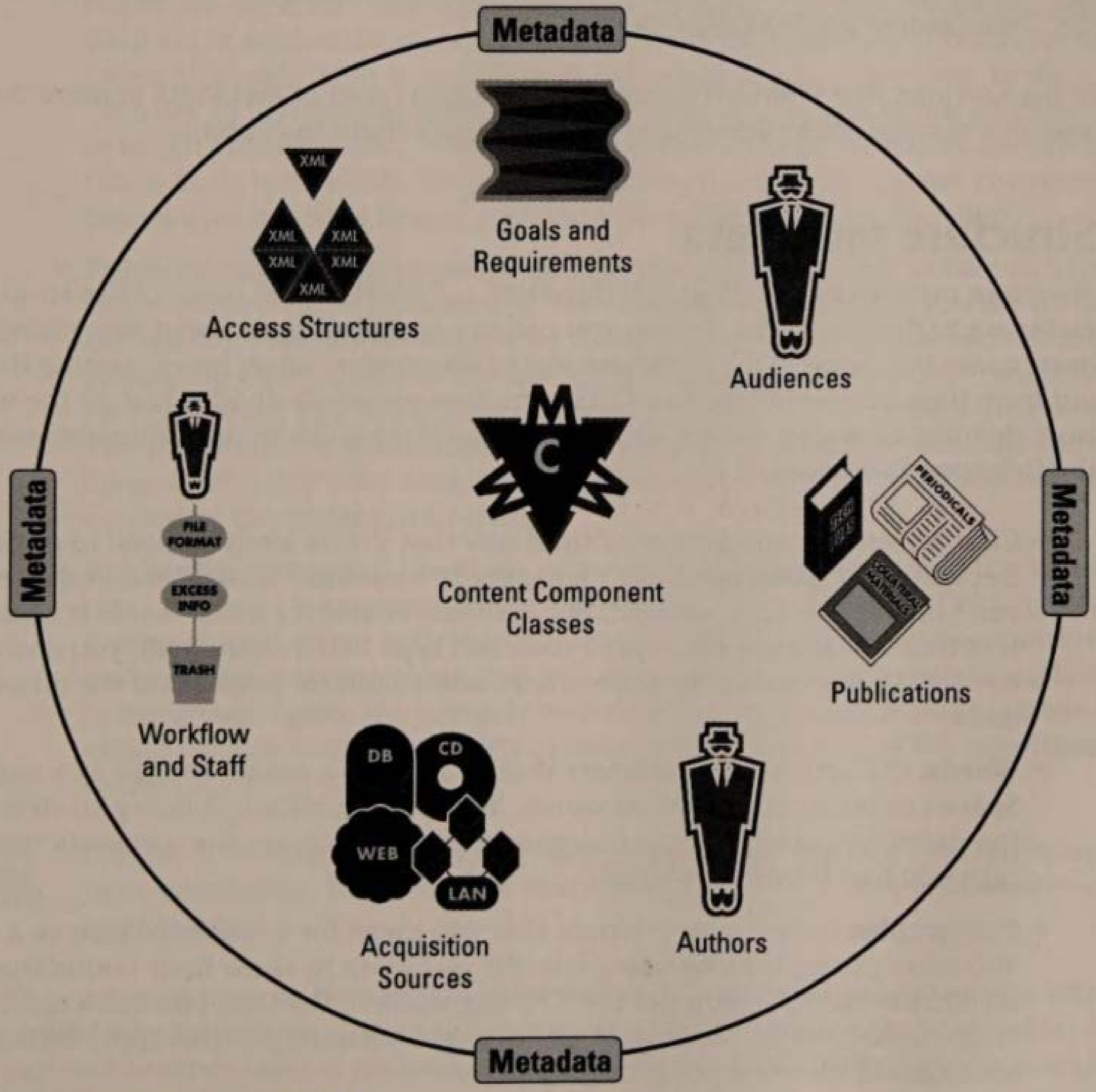

Metadata
XML
XML
XML
XML
XML
XML
Access Structures
Goals and Requirements
Audiences
Metadata
M
C
Content Component Classes
Metadata
PERIODICALS
Publications
FILE FORMAT
EXCESS INFO
TRASH
Workflow and Staff
DB
CD
WEB
LAN
Acquisition Sources
Authors
Metadata

Figure 20-1: The wheel of content management

- **Format metadata:** Applies to any level of structure that you define and marks how you intend to render that structure.
- **Access metadata:** Organizes the structures that you create into hierarchies and other access structures.
- **Management metadata:** The data that you attach to structures to administer and track it.
- **Inclusion metadata:** Stands in for external content. It marks the place where the external content is to go.

One of the best ways that I can think of to get quickly immersed in the wide view of metadata is to look at some good markup. Markup languages (such as XML) are the major way that you apply metadata to content. The other major way that you apply metadata is in database rows and columns.

Cross-Reference

You find much more information about markup languages in Chapter 28, "What Are Content Markup Languages?"

In the sections that follow, I discuss each of these types of metadata in more detail and give markup examples of how you may apply them to content.

Structure metadata

Structure metadata says, "You call this stuff . . ." It's the most basic of the kinds of metadata in the sense that, before you can say anything more about something, you must name that something. Structure metadata creates things by separating them out from their surroundings. In a CMS, structure metadata divides text all the way from defining character boundaries to dividing large collections of publications, as the following list details:

- **Characters:** The smallest structural unit that you're likely to need to notice. Because formatting metadata can apply to individual characters, you often can't ignore this smallest unit. Because one character is the smallest unit of text that a computer can store (you can't type half a character), you don't need a specific marker to show where one character begins and the other ends.
- **Words:** Collections of characters that you intend a reader to take as a unit. Spaces or punctuation set off words. You can say, although doing so stretches the definition a bit, that the spaces and punctuation are the metadata that tells you that a word is a word.
- **Paragraphs:** Collections of words that you mean for a reader to take as a unit. You mark paragraphs by using specific metadata to show their boundaries. (In HTML, for example, you use the `<P>` tag, while in flat text, you use a carriage return). Paragraphs are important to notice because you can apply format metadata at the level of a paragraph.