The Branches of Content Management

A CM Domain White Paper

By Bob Boiko

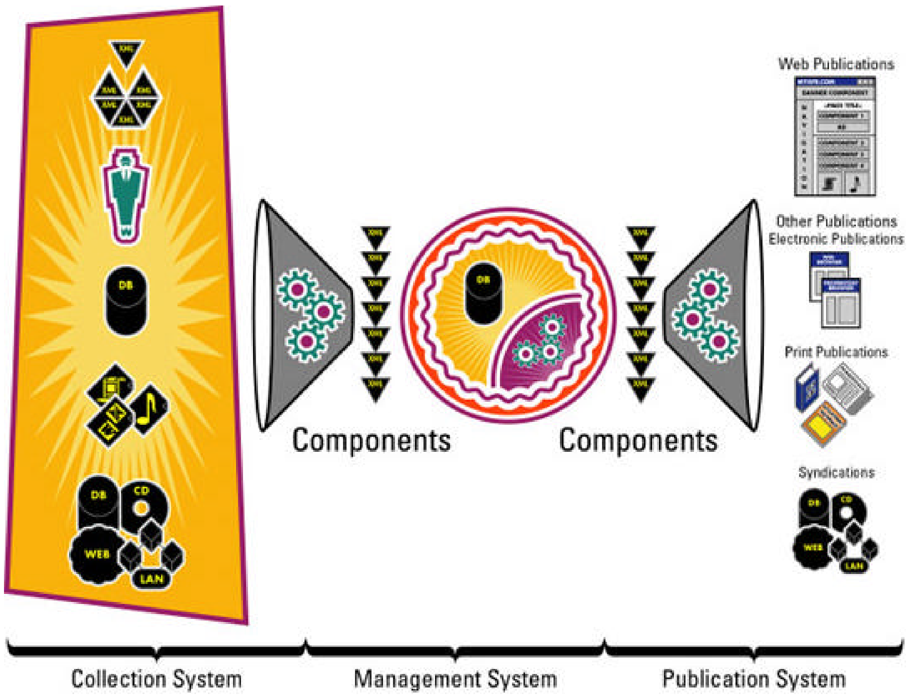

Table of Contents

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 white paper 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 rather 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, then 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 the last piece of management that happens to content before you lay it into a publication.

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 any 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, then 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.

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.

✍✍ **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.

✍✍ **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.

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 white paper 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: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

> *"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 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. The process such a company might use is shown in Figure 1. 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.

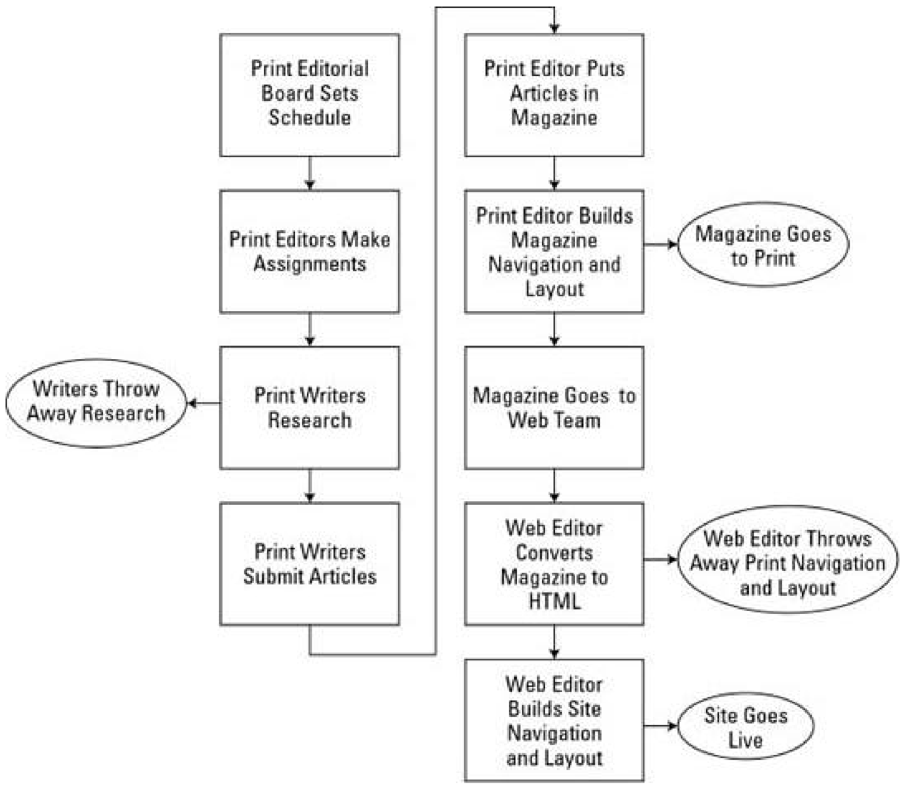
Print Editorial Board Sets Schedule
Print Editors Make Assignments
Print Writers Research
Writers Throw Away Research
Print Writers Submit Articles
Print Editor Puts Articles in Magazine
Print Editor Builds Magazine Navigation and Layout
Magazine Goes to Print
Magazine Goes to Web Team
Web Editor Converts Magazine to HTML
Web Editor Throws Away Print Navigation and Layout
Web Editor Builds Site Navigation and Layout
Site Goes Live

Figure 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 2.

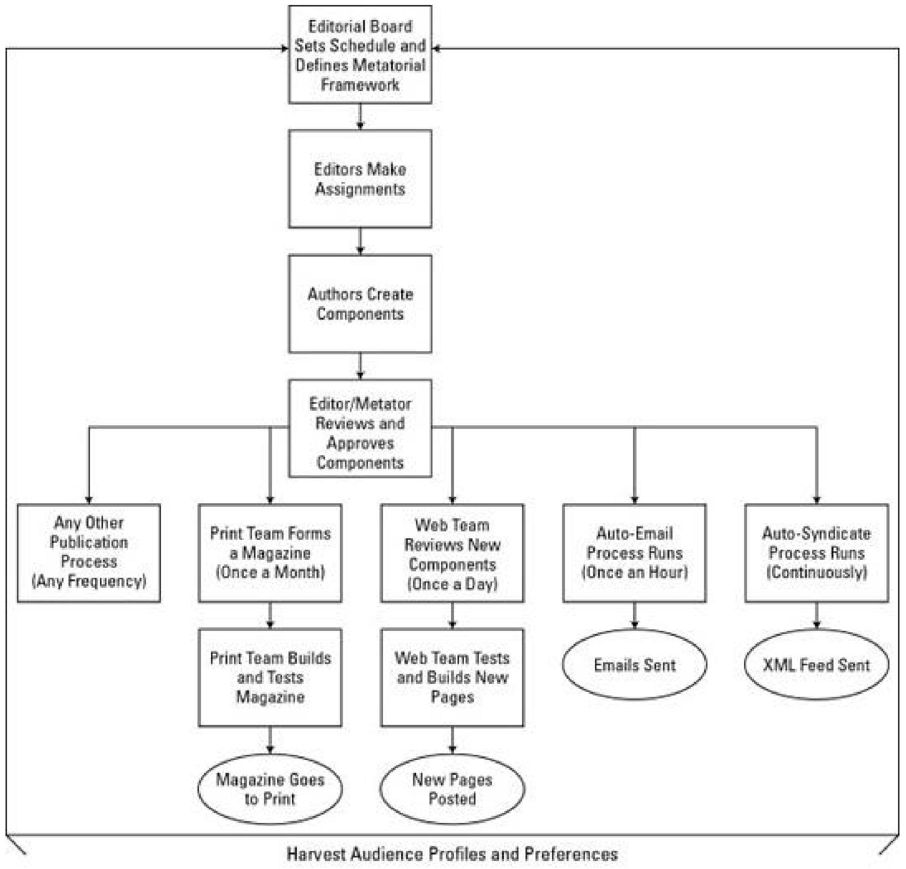
Editorial Board Sets Schedule and Defines Metatorial Framework
Editors Make Assignments
Authors Create Components
Editor/Metator Reviews and Approves Components
Any Other Publication Process (Any Frequency)
Print Team Forms a Magazine (Once a Month)
Web Team Reviews New Components (Once a Day)
Auto-Email 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 2**: The publication process if a CMS is present*

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.

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, timelier 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.

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.

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?

The major components of an online community system are shown in Figure 3.

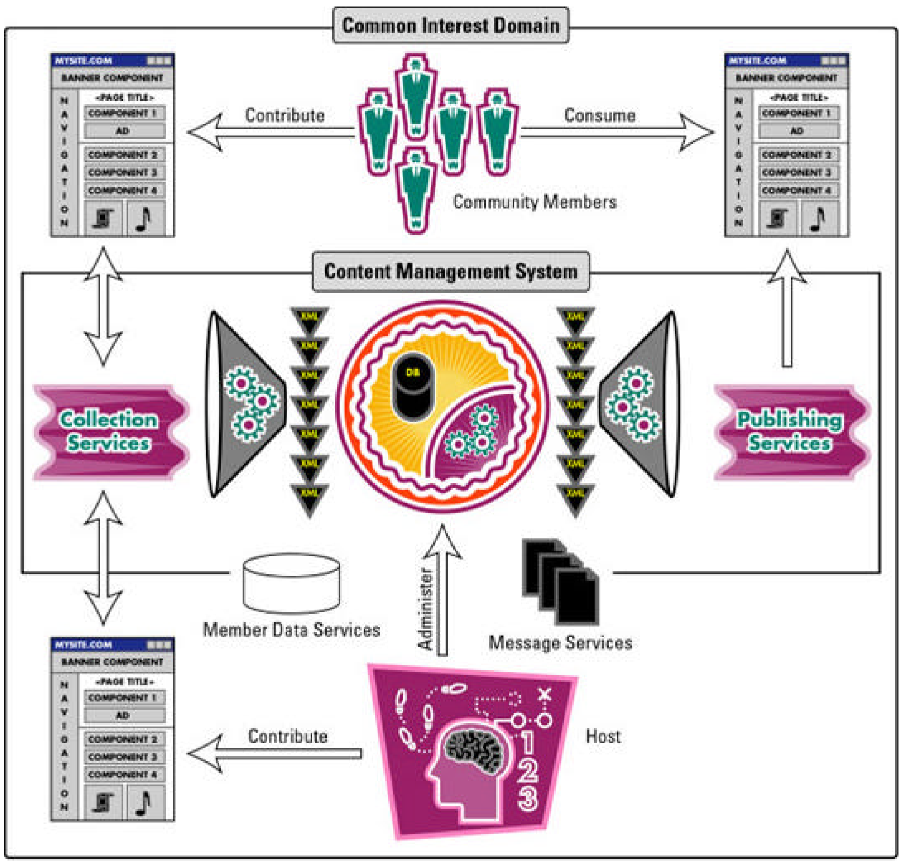

***Figure 3**: An overview of the CMS behind a community Web site*

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 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.

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.

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, then 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 white paperhas 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.