Chapter 1. What Content Management Is (and Isn't)

We tend to look at content management as a digital concept, but it's been around for as long as content. For as long as humans have been creating content, we've been searching for solutions to manage it.

The Library of Alexandria (300 BC to about AD 273) was an early attempt at managing content. It preserved content in the form of papyrus scrolls and codices, and presumably controlled access to them. Librarians were the first content managers.

Fast-forward a couple of thousand years, and the Industrial Revolution and the rise of technology increased the accumulation of information exponentially. The problem of managing it became more critical. The early 20th century was full of great thinkers who examined the problem of communicating and managing information: S.R. Raganathan, Vannevar Bush,[1] Paul Otlet, Claude Shannon, and even Melvil Dewey, the father of the venerable Dewey Decimal System.

So, the need for content management didn't begin with the World Wide Web, but simply shifted into fast-forward when the Web was born in the early '90s. At that moment, the ability to create and publish content tumbled down from its

ivory tower and into the hands of the masses. Almost anyone could create a web page about virtually anything.

Content subsequently exploded. I was a college student at the time, and was slightly obsessed with James Bond. Suddenly, I could find reams and reams of information on 007. The sheer amount of trivia was staggering. Of course, I attempted to print most of it, because if it wasn't on paper, how would I manage it?

It wasn't long before I was coediting a popular James Bond website—Mr. Kiss Kiss Bang Bang.[2] I learned that keeping track of content was a challenge. An article only existed as an HTML file in the web server's root directory at any given time. I wasn't an IT professional back then, so the only backup outside that file was the one on my local computer. There was no versioning or access control—one fat-finger mistake and the entire thing could be gone.

This struck me as dangerous. Even then, I knew that a website is essentially a content-based business, and with nothing more than a bunch of files lying around, I was effectively performing without a net. Content was our only business asset, and I remember thinking it was so brittle; one unforeseen problem and it could simply "blow away" like a dandelion in the wind.

Additionally, each HTML file was a mass of mid-'90s-era

markup, complete with nested `TABLE` and `FONT` tags all over the place. There was no way to separate what was content from what was presentation, and each redesign of the site (there were many) involved manually reworking these files. Server Side Includes had helped to a certain extent, but each file was still a massive glob of mixed content and formatting code.

Some time later, I was working for The Microsoft Network as a forum manager for The World of James Bond. We were still writing HTML files in text editors, but Microsoft had introduced its content managers to the wonders of Visual Source Safe, a now long-since-deprecated source code management system. It provided backups, versioning, and file locking.

This clearly made us safer from a risk management perspective, but there was a mental shift too. We had a safety net now. The content we were creating had solidity to it. There was history and context. We were editing and grooming a continuing body of content, rather than just changing it in place. Content didn't exist only in simple files, but lived inside a larger system which provided a set of services to protect and safeguard it. We had gone from hiding money inside our mattresses to depositing it at an FDIC-insured financial institution.

Finally, at some crude level, my content was *managed*. It was

still all mixed up with its presentation, and probably had a host of other problems, but I was at least a couple of steps safer than I had been before. Without me realizing it, Visual Source Safe effectively became my first content management system.

A lot has changed since then, but let's start at the beginning. Along the way, we'll hopefully answer all of the following questions:

- What is *content*?
- What is content *management*?
- What is a content management *system*?
- What are the different *types* of content management systems?
- What does a content management system *do*?
- What *doesn't* a content management system do?

What Is Content?

Many people have tried to draw a distinction between the fuzzy concepts of "data," "information," "content," and even "knowledge." Bob Boiko dedicated the entire first part of his seminal work *The Content Management Bible* (Wiley) to this

question—some 5 chapters and 61 pages.

We're not going to go that far, but we'll summarize by simply differentiating between content and raw data, which is likely the highest-value return we can get out of the question. How is content management any different from managing any other type of data?

There are two key differences:

- Content is *created* differently.
- Content is *used* differently.

Created by Humans via Editorial Process

Content is created through "editorial process." This process is what humans do to prepare information for publication to an audience. It involves modeling, authoring, editing, reviewing, approving, versioning, comparing, and controlling.

The creation of a news article, for example, is highly subjective. Two editorial teams, given the same information, might develop two completely different news articles due to the human factors involved in the editorial process. Whereas a computational function seeks to deliver the same output from the same input every time, an editorial process never quite does.

The creation of content pivots largely on the opinions of human editors:

- What should the subject of the content be?
- Who is the intended audience of the content?
- From what angle should the subject be approached?
- How long should the content be?
- Does it need to be supported by media?

Despite significant advances in computing, these are not decisions a computer will make. These are messy, subjective, imperfect decisions that pour forth from the mind of a human editor sitting behind a keyboard. A small deviation in any of them (the proverbial flapping of a butterfly's wings) can spin a piece of content in an entirely different direction. Content is subjective and open for evaluation and interpretation. It has nuance. It is artisanal.

The editorial process is iterative—content is rarely created once, perfectly, and then never touched again. Rather, content is roughed in and refined over and over like clay on a potter's wheel, often even after being published. Content can change with the passage of time and the evolution of circumstance. What was relevant at one point might need to change or be withdrawn later.

Content is constantly in some state of flux and turnover. It's never "right." It's just "right now."

Compare this process to the creation of the record of a retail sale. There is no editorial process involved with swiping your credit card across a terminal. Furthermore, the data created is not subjected to any other process, will not be reviewed and approved. It is not subjective, it is deterministic. The transaction happened in an instant, a historical record was created, and that's that.

The sales transaction is designed to be as repeatable and devoid of subjective opinion as possible. It will never be edited (indeed, to do so would likely violate a policy or law). It is cold, sterile, and inert by design.

Consequently, management of these two types of information is quite different.

Intended for Human Consumption via Publication to an Audience

Content is data we create for a specific purpose: to distribute it with the intention of it ultimately being consumed by other humans. Sure, it might be scooped up by another computer via an API and rearranged and published somewhere else, but eventually the information is going to make its way to a human somewhere.

Bob Boiko writes:

> *If you strip away all of the technology and terminology that we use to describe information systems, what do you have left? A simple and utterly commonplace idea: Information systems help you talk to people who are not in front of you.*[3]

Our sales transaction from the prior section has no such destiny. It was created as a backward-looking record of a historical event. It will likely not be consumed by someone in the future, except in aggregate through reporting of some kind. It may be retrieved and reviewed individually, but only by necessity and likely on an exception basis.

Our news article, by contrast, was created as a forward-looking item to be published in the future and consumed by humans, through whatever channel (perhaps more than one). It might be repurposed, abbreviated, rearranged, and reformatted, but the ultimate goal for it is to be consumed and evaluated by another human being.

Our content has value in the future. It might be consumed for years (even centuries or millennia), and can continue providing value to the organization far into the future. Every time our article is read, or every time a new employee reads the payroll policy, there is a benefit attributed to the content creator.

Content is an investment in the future, not a record of the past.

A Definition of Content

Bringing these two concepts together, we arrive at our best attempt at a concise definition:

> *Content is information produced through editorial process and ultimately intended for human consumption via publication.*

This definition also points to a core dichotomy of content management: the difference between (1) management and (2) delivery. Content is created and managed, *then* it is published and delivered. The two disciplines require different skills and mindsets, and the state of current technology is creating more and more differences every day.

We'll revisit this two-sided approach to content management throughout this book.

What Is a Content Management System?

A content management system (CMS) is a software package that provides some level of automation for the tasks required to effectively manage content.

A CMS is usually server-based,[4] multiuser software that interacts with content stored in a repository. This repository might be located on the same server, as part of the same software package, or in a separate storage facility entirely.

A CMS allows editors to create new content, edit existing content, perform editorial processes on content, and ultimately make that content available to other people to consume it.

Logically, a CMS is comprised of many parts. The editing interface, repository, publishing mechanisms, etc., might all be separate, autonomous parts of the system behind the scenes. However, to a non-technical editor, all of these parts are generally viewed as a single, monolithic whole: "the CMS."

The Discipline Versus the Software

What's important to note is that a "content management system" is a specific manifestation of *software* designed to enable the *discipline* of content management. Just like a Ford Taurus is a specific manifestation of a device enabling personal transportation, Drupal, WordPress, and Episerver are specific manifestations of software enabling content management.

The discipline of content management—the accumulated

theories, best practices, and accepted patterns of the field—transcends any specific system. In this sense, it's a Platonic ideal: an abstract, subjective representation of how content is to be managed.

The specifics of this ideal can be very different depending on the experiences, preferences, and needs of the observer. This means there's no single, accepted definition for content management as a discipline, just a set of debatable best practices. While people might try to lay claim to a Grand Unified Theory of Content Management, no such thing exists.

Thankfully, this means that skill with a particular content management system can be somewhat transferable. Even if System A differs from System B in extreme ways, they both still need to solve transcendent problems of the discipline, like workflow, versioning, publishing, etc. While specific technical skills might not transfer, working with a content management *system* requires the exercise and development of skills in the content management *discipline*.

So, a CMS is a tool to assist in and enable the theoretical ideal of content management. How well any one CMS successfully brings that ideal to life is the subject of great debate and Internet flame wars.[5]

Types of Content Management Systems

We defined content as “information created through editorial process and intended for human consumption.” Note that there was no mention of the Web in this definition (nor of the Internet itself, really).

However, given that this is a book about *web* content management, it’s probably best that we define some different flavors of content management rather than lumping them into one big bucket.

The “big four” of content management might be identified as:

Web content management (WCM)

> The management of content primarily intended for mass delivery via a website. WCM excels at separating content from presentation and publishing to multiple channels.

Enterprise content management (ECM)

> The management of general business content, not necessarily intended for mass delivery or consumption (e.g., employee resumes, incident reports, memos,

etc.). This flavor was more traditionally known as "document management," but the label has been generalized over the years. ECM excels in collaboration, access control, and file management.

Digital asset management (DAM)

The management and manipulation of rich digital assets such as images, audio, and video for usage in other media. DAM excels at metadata and renditioning.

Records management (RM)

The management of transactional information and other records that are created as a byproduct of business operations (e.g., sales records, access records, contracts, etc.). RM excels at retention and access control.

Clearly, the line blurs here quite a bit. A DAM[6] system is often used to provide content for a website through integration with a WCM. Furthermore, some ECM systems have systems by which they can publish some of their information to the Web.

Software systems are known only through their *intended* use and their perception in the industry. Drupal is well known as a WCM system, but there are undoubtedly organizations using it to manage internal enterprise content. Conversely,

Documentum is an ECM system, but some organizations might use it to deliver all or part of their websites.

DAM is interesting in that it is differentiated primarily on the basis of what it *does* to content. While almost any content management system can store video and image files, and ECM actually excels at it, DAM goes a step further by providing unique tools to render and transform digital assets. Images can be mass-resized and video can be spliced and edited directly inside the system, making a DAM system's point of differentiation one of processes that can be applied to content. Therefore, the core management features of a DAM system overlay quite closely those of an ECM system, with the DAM system layering a level of functionality on top. (Indeed, many DAM systems are sold simply as add-ons to ECM systems.)

There are other, even blurrier shades of gray. Some examples:

Component content management systems (CCMSs)

> Used for management of extremely fine-grained content (paragraphs, sentences, and even individual words), often to assemble documentation or highly technical content.

Learning management systems (LMSs)

Used for management of learning resources and student interaction; most colleges and universities manage syllabi and the learning process via an LMS.

Portals

Used for management, presentation, and aggregation of multiple streams of information into a unified system.

Again, the lines here are very blurry.

Only some of what an LMS does is specific and unique to an LMS. Many different WCM systems, for instance, have add-ons and extensions that claim to turn them into an LMS, and many more are simply used as such out of the box.

In the end, a given software system is mentally classified among the public based on several factors:

- The market in which it promotes itself and in which it competes
- The use cases and examples that the user community creates and promotes
- The specific features designed to meet the needs of a particular user or type of content

CMS software is *targeted* at particular markets or usage scenarios. That has never stopped anyone from using it in

ways outside of the one the vendor designed it for.

For the purposes of this book, we will concentrate on mainstream WCM—that software designed to manage a website intended for public delivery and consumption.

(And, at the risk of beating this subject to death, even this designation is blurry. Organizations commonly power social networking platforms from their WCM systems, managing content that ends up as Facebook updates, tweets, or even emails. While this is not technically "website content," our definition will have to suffice.)

What a CMS Does

Let's break down the core functions of a CMS. In broad terms, what is the value proposition? Why are we better off with a CMS than without?

Control Content

A CMS allows us to get control of our content, which is something you'll understand well if your content is out of control. A CMS keeps track of content. It "knows" where our content is, what condition it's in, who can access it, and how it relates to other content. Furthermore, it seeks to prevent bad things from happening to our content.

Specifically, a CMS provides core control functions, such as:

Permissions

> Who can see this content? Who can change it? Who can delete it?

State management and workflow

> Is this content published? Is it in draft? Has it been archived and removed from the public?

Versioning

> How many times has this content changed? What did it look like three months ago? How does that version differ from the current version? Can I restore or republish an older version?

Dependency management

> What content is being used by what other content? If I delete this content, how does that affect other content? What content is currently “orphaned” and unused?

Search and organization

> How do I find a specific piece of content? How do I find all content that refers to X? How do I group and relate content so it’s easier to manage?

Each of these items increases our level of control over our content and reduces risk—there is less chance that the shareholder report will be released early, or that the only copy of our procedures manual will be deleted accidentally.

Allow Content Reuse

Using content in more than one place and in more than one way increases its value. Some examples:

- A news article appears on its own page, but also as a teaser on a category page and in multiple "Related Article" sidebars.
- An author's bio appears at the bottom of all articles written by that person.
- A privacy statement appears at the bottom of every page on a website.

In these situations, this information is not created every time in every location, but simply retrieved and displayed from a common location.

This reuse of content was one of the original problems that vexed early web developers. Remember the James Bond site I discussed earlier? One of the great frustrations was creating an article, and then adding it to all the index pages where it was supposed to appear. If we ever deleted the

article or changed the title, we'd have to go find all the references and remove or change them.

This problem was mitigated somewhat by Server Side Includes, which allowed page editors to insert a snippet of HTML by simply referring to a separate file—the files were combined on the server prior to delivery. Later platforms tried to automate this even further; Microsoft FrontPage, for example, had a feature it explicitly called "Shared Borders."

The ability to reuse content is highly dependent on the structure of that content. Your ability to structure your content accurately for optimal reuse is highly dependent on the features your CMS provides for you.

Allow Content Automation and Aggregation

Having all of our content in a single location makes it easier to query and manipulate it. If we want to find all news articles that were written last week and mention the word "SPECTRE," we can do that because there is one system that "knows" all about our content.

If our content is structured correctly, we can manipulate it to display in different formats, publish it to different locations, and rearrange it on the fly to serve the needs of our visitors more effectively:

- We can allow users to consume content in other formats, such as PDF or other ebook formats.

- We can automatically create lists and navigation (more generally, "content aggregations"—see Chapter 7) for our website.

- We can create multiple translations of content to ensure we deliver the language most appropriate to the current user.

- We can alter the content we publish in real time based on the specific behaviors and conditions exhibited by our visitors.

A CMS enables this by structuring, storing, examining, and providing query facilities around our content. It becomes the single source of information about our content; the thing that has its arms around the entire repository; the oracle we can consult to find information about our content.

Increase Editorial Efficiency

The ability of editors to create and edit content quickly and accurately is enormously affected by the platform used. It's rare to find an editor who has unconditional love for a CMS, but the alternative, editing a website manually, is clearly much less desirable.

Editor efficiency is increased by a system that controls what type of content editors can and can't add, what formatting tools are available to them, how their content is structured in the editing interface, how the editorial workflow and collaboration are managed, and what happens to their content after they publish.

A good CMS enables editors to publish more content in a shorter time frame (it increases "editorial throughput"), and to control and manage the published content with a lower amount of friction or drag on their process.

Editorial efficiency has a huge impact on morale, which is intangible but critical. Many editors have a historically antagonistic relationship with their CMSs, and nothing destroys editorial efficiency more quickly than a clunky editorial interface and flow.

What a CMS Doesn't Do

Now for the bad news: there are things a CMS doesn't do. More specifically, there are things that a CMS doesn't do *but that people mistakenly assume it does,* which leads to problems and unfulfilled expectations.

Create Content

A CMS simply *manages* content, it doesn't *create* content. It

doesn't write your news articles, procedure documents, or blog posts. You must still provide the editorial horsepower to generate the content that it's supposed to be managing.

Many times, a CMS implementation has ended with a group of people looking at each other and thinking, "So...now what?" Every web development shop in the country can tell you stories about the shiny new CMS that was never once used by the client because they never changed their site after the day it launched. Occasionally, my company has taken calls from clients *years* after their sites launched wanting to know how to log in to their CMS for the first time.

Related to this, a CMS won't ensure that your content is any good, either. Although a CMS might offer several tools to minimize poor-quality content from a technical standpoint (ensuring that hyperlinks are valid, or that all images have `ALT` tags, for instance), a CMS cannot edit your content to be sure it makes sense and meets the needs of your audience.

The best-laid plans to create massive amounts of quality content often fall through when confronted with the hard reality of schedule pressure and business deadlines. You need to ensure that your content creation process exists apart from your CMS.

Create Marketing Plans

Even assuming your content is created consistently and managed well, that doesn't mean it actually provides your organization with any value.

A CMS doesn't "know" anything about marketing. While some systems have marketing tools built into them, they still depend on human beings for direction. Effective marketing is a uniquely human practice involving a combination of aesthetics, sociology, psychology, experience, and intuition. A CMS can make executing your marketing plans easier and more efficient, but those plans still need to be conceived, created, and analyzed by a competent human.

A CMS doesn't take the place of a creative team that understands your marketplace, your customers, your competitors, and what you need to do to differentiate yourself. No software can take the place of a good digital marketing strategy or team.

Effectively Format Content

While a CMS can structure content and automatically format it during publication, there is still an unfortunate amount of room for a human editor to screw it up. Most CMSs have a rich text editor or some other interface element that allows editors to format text and images. This can lead to things like:

- Too much use of **bold** and *italics*
- Inconsistent alignment of content
- Random and inconsistent hyperlinking
- Poor image placement

Editors have never seen a button on an editing interface that they didn't want to press. The only way to limit this seems to be to remove as many editing options as possible, then try to withstand the hailstorm of editor complaints that will inevitably follow.

Provide Governance

"Governance" describes the access to and processes around your content: who has access to what, and what processes/steps they follow to make changes to it. For example:

- If Bob adds a news article, who needs to approve this, and what does that approval look like? Does someone copyedit and someone else edit for quality, voice, and tone? Can you diagram this process out on a piece of paper?
- If John wants to change how the news archives are organized, and the CMS allows him to do this...can he?

What process does he have to go through to do this?

- If Jennifer wants an account on the CMS to start creating content, how does she get that? Who decides who is allowed to become an editor?

Every CMS has some method to limit the actions a user can take, but these limits have to be defined in advance. The CMS will simply carry out what your organization directs it to do. These plans have to be created through human interaction and judgment, then converted into the permissions and access limits the CMS can enforce.

Governance is primarily a human discipline. You are determining the processes and policies that humans will abide by when working with your content. The CMS is just a framework for enforcement.

[1] Bush wrote an incredibly influential essay in 1945 entitled "How We May Think" that lamented the lack of necessary information management tools and envisioned a future that turned out to be remarkably similar to what actually happened over the 70 years following its publication. Bush essentially described the World Wide Web about 50 years before Tim Berners-Lee made it happen.

[2] While now defunct, the site was at *http://ianfleming.org* for many years. There's an archive of screencaps in a public

Facebook group, for those interested.

[3] Bob Boiko, *Laughing at the CIO* (Medford: CyberAge Books, 2007), 99–100.

[4] Early CMSs were not server-based. Some of the first CMSs were client-side templating tools, such as CityDesk, MarsEdit, and Radio UserLand. These were installable software packages that allowed editors to work on content in a desktop interface. The systems then templated that content and transferred the resulting HTML to a server environment, usually via FTP. Today, these are often referred to as "desktop content management" tools. Few of these platforms are actively developed, but most still exist for purchase.

[5] "Flame" was the original term for two people hurling insults at each other in front of a virtual audience through the anonymous magic of the Internet. I think this is now called "Facebook."

[6] An acronym that is the source of countless jokes, which, contrary to what some might say, just never seem to get old.