CONTENT MANAGEMENT

Bridging the Gap Between Theory and Practice

Edited by
George Pullman and Baotong Gu

Baywood's Technical Communications Series
Series Editor: Charles H. Sides

BAYWOOD PUBLISHING COMPANY, INC.
AMITYVILLE, NEW YORK

CHAPTER ONE

Experiences with Building a Narrative Web Content Management System: Best Practices for Developing Specialized Content Management Systems (and Lessons Learned for the Classroom)

Rudy McDaniel

In this chapter, I begin by examining the process of creating a specialized online content management system (CMS) and conclude by applying the techniques and lessons learned from this experience to classroom pedagogy. Specifically, I consider the development of a Web-based CMS that was created using stories as the raw material for propagating organizational knowledge (a more detailed description of this process is found in McDaniel, 2004). While the theoretical basis for such an effort is an interesting study in its own regard (see Denning, 2001; Post, 2002; Smart, 1999 for studies of storytelling at work in organizations such as the Bank of Canada, the World Bank, and NASA; or Kim (2005) for a discussion of narrative as it applies to the field of technical communication), the issues involved with the construction of such an interface deserve their own unique discussion. In addition, this humanities-friendly data model presents an opportunity for studying the implications of using content-compatible CMS design methodologies in a classroom with advanced writing, communications, or digital media students.

The chapter is organized into three sections. In the first section, I examine the fundamental components of a CMS and present several theoretical considerations for building a specialized CMS. For example, construction of a *narrative* CMS relies on research from organizational and business communication (Denning, 2001, 2004), from cognitive psychology (Bruner, 1991), and from computer

science (Minsky, 1985; Schank, 1990). The process of building a narrative CMS is detailed in terms of a standard software development lifecycle, which begins with an abstract requirements and specification phase and gradually moves toward a more concrete implementation. This process can be adapted to other specialized CMS designs simply by transforming the content base, the encapsulating unit for this content, and the types of design decisions that will eventually be built into an interface. Many of the complex technological processes may require interdisciplinary collaboration from other fields such as information technology, engineering, computer science, or digital media.

I claim that this type of practice-oriented and interdisciplinary synthesis is precisely the type of activity that knowledge management researchers (Hughes, 2002; Wick, 2000) argue is necessary in order to empower technical communicators and situate them in more desirable positions within their organizations. In addition, I assert that writers and digital media specialists are well suited to be operators, administrators, or developers of such systems—that a knowledge and background supplemented with humanities expertise may improve the sterile and positivist pathways and applications of modern content management techniques. This is especially true given the tendency of IT firms to overemphasize technology, to misinterpret (or to wholly ignore) the needs of their audience, and to underemphasize the humanistic components of information applications (see Davenport & Prusak, 1997).

Next, I will detail the practical aspects of constructing CMS technologies and describe some of the decisions that were made during development of my own CMS. In the case of a narrative-based CMS, a system such as the one I discuss can be built using a modicum of student talent, open-source software, and a metadata classification system such as XML. As I discuss the process involved in building my own CMS, I will write about lessons learned from this experience with regard to user privacy; the selection, storage, and classification of relevant information; the use of open-source versus proprietary software; and the decision-making process behind building a genre-specific content management system.

The final portion of the chapter will focus on ideas for teaching by using CMS. My argument is that a thorough understanding of topics such as CMS, XML, and the practices for building such technological systems is essential for the students graduating with technical writing or new media degrees. Incorporating these topics through classroom exercises, readings, and discussions equips students to be better prepared with both the fundamental skills necessary for survival in the industry with the critical thinking skills necessary to truly innovate in the field.

As Zimmerman (2001) writes, it will not be long before autonomous computer agents write their own software documentation. We already have mechanisms in place for computer programming languages such as Java (JavaDoc) and PHP (PHPDoc), but such documentation is rather mechanical at this point and of use mostly only to other programmers and developers. The day in which computer

algorithms begin to write useful instructions for end users is still some time away, but it is likely to be inevitable. When that day comes, the ability to translate core technical skills into other types of innovative ideas and projects will be essential for newly trained technical communicators and new media practitioners entering the workforce. CMS technologies provide a nice starting point for examining these sorts of issues. Their reliance on core Internet technologies, their ability to generate multiuse content for a variety of audiences and contexts, and their compatibility with metadata classification languages are all characteristics that position these complex systems as natively and inherently useful teaching tools for those courses in which the study of complex information is integral.

PART I: LIBRARY (COMPONENTS OF A SPECIALIZED WCMS)

The process of designing, implementing, and managing CMS systems in online environments has been described as Web content management (WCM) (Yu, 2005). We can therefore describe the technologies supporting this task as Web content management systems (WCMS). In its most general form, a WCMS is nothing more than a database-driven Web site. Though some claim such a comparison is a bit oversimplified for CMS in general (see Goans, Leach, & Vogel, 2005), it is certain that the architectural basis for any online CMS generally involves a robust database paired with an interface capable of communicating with this data source. Such a scenario is also common for systems that *do not* rely on the Internet; complex configurations of content are generally only possible through the use of database technologies capable of sorting and shifting information in response to user commands. In a non-networked environment, such commands are simply generated by an interface from workstation application software rather than from a Web browser.

After these basic database and interface technologies have been configured, a WCMS can be customized for a variety of pedagogical and industrial applications, from monitoring and adapting student learning outcomes in fields such as mathematics and writing (Deacon, Jaftha, & Horwitz, 2004) to modularizing content in complex writing scenarios (Farkas, 2005; Goans et al., 2005; Surjanto, Ritter, & Loeser, 2000). A basic premise leading to the success of early WCM systems is the idea that a finite number of efficiently constructed information units, or nodes, can be coupled with database support to dynamically generate a large number of content configurations and documents. This emergent complexity is based on relatively simple changes to the requests made to the database server. The small number of template pages is then much easier to manage and maintain than vast collections of documents with individualized headings, inconsistent content, and stylistic disparities.

A specialized WCMS can be very useful for solving many of the issues encountered in large-scale Web development projects, especially when content is

added in a distributed fashion. As Goans et al. (2005) note, such environments often rely upon multiple authors, each with their own technological backgrounds and ideas about how given content should be represented on the Web. While such stylistic and structural differences are interesting to observe, particularly in terms of emergent properties that may form when multiple authors consider a single topic from different angles, they also lead to serious problems when individual pages are collated and represented under a common organizational framework. A primary problem here is inconsistency, which contributes to an overall "lack of organizational voice and credibility" (Goans et al., 2005, p. 30). The architecture of a WCMS, which imposes a higher-level order and structure upon the data through the use of form fields and data verification tools, often alleviates many of these inconsistency issues. Furthermore, these architectures improve maintenance and administration of the content collection by trimming the number of editable files to a manageable number.

When a CMS system is housed on the Internet—thereby becoming a WCMS—additional server software is needed to manage communications between client and host computers. Using an interface, a Web server, and a database management system, an online content management system provides a dynamic and interactive alternative to information access and retrieval as opposed to traditional and static HTML delivery systems. The core of a general Web-based content management system is configured as shown in Figure 1. In any given transaction, a request for information is first relayed from the interface to the Web server, which sends any database queries on to the database server. Appropriate subsets of information are then returned to the interface, which displays records accordingly and filters information down to what is hopefully an absorbable and manageable level of granularity for the end user. Additional searches or sorts requested by the user will then be executed immediately, either with no additional requests to the two servers (client-side processing) or with additional round-trip visits to one or more servers (server-side processing).

The transformation of the relevancy of digital information as it moves in either direction through this system is worth noting. Using Davenport and Prusak's (1997) distinction between data, information, and knowledge, we see the process

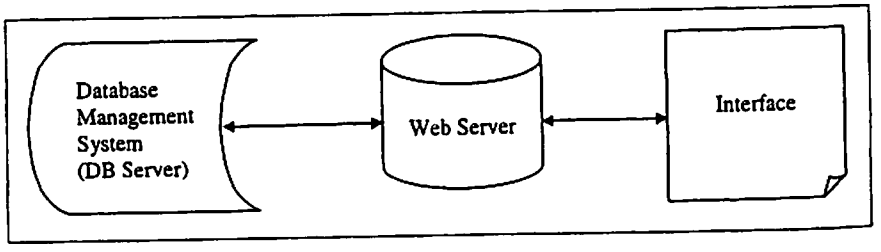


Figure 1. General CMS architecture.

of moving from raw data to usable knowledge as a filtering process, with the most useful, relevant, and appropriate sources representing information that will eventually be internalized and encoded as knowledge. This stored knowledge can then be transformed and adapted as new situations emerge that may require access to the same sorts of cognitively encoded memories and observations in a slightly different context. In Figure 1, assuming that data entry also occurs from within the interface, data primarily flows from the right to the left, starting in the interface and making its way into the database server. Information, though, will flow from the opposite direction, beginning its journey in the database server and eventually filtering through the Web server to arrive in the interface. Assuming that the WCMS is doing its job correctly and that the user has formed appropriate search parameters, the filtering process will be successful in separating data, or world observations, from information, or contextually relevant and useful world observations (Davenport & Prusak, 1997).

Given these three general requirements, it is quite possible for an entire WCMS to be developed and deployed on a single computer. In a testing environment, for example, it is not uncommon for developers to run Web server software, database server software, and programming or interface design software concurrently on a desktop or laptop computer. While a technical discussion of this installation and configuration is beyond the scope of this chapter, the point here is that the tools for constructing and creating a specialized WCMS are readily available and configurable for a wide variety of needs. In addition, many of these robust tools are available for free. Both commercial and open-source tools (see Deacon et al., 2004 or Goans et al., 2005, for a discussion of some of these applications) are available to support configurations of both specialized and general content management systems.

Developing a specialized WCMS, then, primarily involves manipulation of the data situated at either end of this general WCMS architecture. In other words, specialization occurs at both the database level, with the insertion of specialized sets of data appropriate for a given domain, and at the interface level, with the user-centered design of a product that meets the needs of those searching for information within that domain. The Web server, which coordinates the flow of specialized information in both directions, is largely unaffected by specialization. Examples of one such specialization technique—for both database and interface designs—will be discussed in Part II of this chapter.

Given this general architecture, the development of a specialized online CMS is largely concerned with soliciting the appropriate sources of specialized content. We can refer to this database collection as the *library* of the CMS. For example, a CMS concerned with automotive repair needs to gather and store data related to vehicle models and part numbers, service locations, service histories, technician personnel, and other types of data associated with vehicles and their parts. A CMS used by environmentalists for environmental awareness campaigns would instead populate their library with industrial propagation information,

environment incidents, and public relations contact information, among other things. The key, of course, is to gather enough data so that a wide variety of problems can be addressed with the correct sources of information from the library. While information overload or lack of focalization can pose troublesome issues, these problems can generally be dealt with through the creation of a user-centered interface. Duplicitous or overlapping information can also be handled through database optimization techniques.

A primary concern with the library/database is describing its contents in a manner such that users without an expert knowledge of taxonomy will be able to find an appropriate number of usable sources. For example, a user of the automotive CMS looking for a specific recall part for the Ford Motor Company will want to peruse only vehicles from the particular year during the period of time in which this particular part was manufactured and installed. If the CMS were to classify each vehicle record as only "foreign" or "domestic," the results returned from a query would likely be too broad, and the user would be faced with a situation in which information overload was a significant possibility.

On the other hand, if the CMS library classified each record at too precise a level, perhaps by using the actual Vehicle Identification Number (VIN) of each Ford vehicle, the user would be subjected to the cumbersome task of navigating through each and every affected vehicle on an individual basis. For a specific task, then, such as mailing out letters to the owners of these vehicles, the operation would be far too time consuming to be used in a realistic scenario. The task would be much better suited to a batch mode type of operation in which letters were automatically generated based on a range of specific vehicle models and a filtering of relevant dates.

Fortunately, when constructing a CMS library, it is relatively easy to perform these types of sorts due to the inherent structure of databases in general. Specialized collections of content are generally grouped together in tables, in which each table will usually have a unique identifier that separates one row in the table from another. In a relational database, where various database tables and entities can have precise and defined relationships with other tables and entities, each column within a table serves as a potentially searchable entity within the database. For instance, a table in the database containing vehicle models, owners, and VINs can be searched by any of those three characteristics, thus enabling a variety of different search techniques depending on the particular data being sought. In addition, a second table might contain VIN-specific information, such as the year in which the vehicle was manufactured, its base configuration, and the manufacturing plant where it was produced. Fortunately for the user, complicated queries to these distributed, relational databases can be hard coded into the interface so that a given user will need to know only which button to click rather than the specific syntax and particularities involved with linking and selecting appropriate data and tables from the CMS library.

To summarize, building a specialized online CMS involves gathering the appropriate collection of specialized data (building the library) and then implementing access to this customized collection through a special interface. Queries, or commands designed to solicit specific subsets of information from the library, are designed to be attached to specific objects (buttons, scrollbars, links) that exist within the interface. In the next portion of this chapter, I describe a specific instance of this process through my experience in designing and building a narrative-based WCMS.

PART II: LAB (CONSTRUCTION GUIDELINES)

Given the general requirements outlined in Part I of this chapter, we can now apply these design specifications to the construction of a specialized WCMS. The WCMS discussed in this section was developed as an accompanying software product for my dissertation (see McDaniel, 2004), and its purpose was to demonstrate how socio-organizational approaches to knowledge management might be embedded in positivist technologies, which generally focus more on the codification and transmission of data rather than on the influence of the environments and communities from which this data emerged. Wick (2000) provides a useful overview of several approaches to knowledge management (KM), which range from fully document-centered KM to technological and socio-organizational approaches. The most desirable approach is found in the practices of the "knowledge organization," which "emphasizes knowledge as the core source of competitive advantage" and "approaches knowledge management as a mission-critical issue affecting all areas of organization" (p. 520). The goal of this particular WCMS was to help encourage organizational growth toward a level of KM maturity by providing a means for collecting, storing, and disseminating tacit knowledge in the form of occupational narratives.

This WCMS operates by using a specialized database composed of narrative "objects," or textual stories represented in computational structures. In this context, the terms "narrative" and "story" are used interchangeably for simplicity's sake, though this practice deviates from the discussion of terms by some literary theorists who may choose to differentiate between the process of telling a story and the story itself, or to impose multiple and layered definitions on narrative (see Bal, 1997; Genette, 1980). Both terms in this chapter are used in the operational sense; they refer to the expression of a causal event sequence recounting some experiences of a primary character (protagonist) working within some environment to overcome some obstacle or obstacles (antagonists) in pursuit of some goal or goals. While this general definition is flexible enough to accommodate specialized forms of narrative (epic poetry, lyrical or musical storytelling, oral storytelling or story circles, etc.) the implementation of a CMS generally requires stories to be expressed in a format that can easily be encoded

into a text-based (e.g., hypertext) medium. It is important to note that such encoding does not exclude multimodal storytelling, as images and audio information can also be accommodated by hypertext markup.

A narrative CMS is useful for many different types of applications, particularly in knowledge management scenarios in which specialized types of knowledge require discursive movements from subject matter experts to those audiences unfamiliar with its intricacies. Such applications are especially well suited to first-person-perspective experiential stories, in which a writer explains his or her experiences in dealing with some type of problem by telling a story about his or her situation. This technique is well known from Orr's (1996) observations of expert Xerox technicians and their tendency to swap stories about challenging technical problems during coffee breaks. Post (2002) writes of NASA's investigations into using the same type of narrative exchange in a written form, by having expert project managers compose narratives based on their knowledge and experience. These stories were then compiled into an internal newsletter and circulated to appropriate employees. Even highly technical concepts were able to be successfully shared through the careful use of storytelling and the creative use of figurative language and metaphor.

Stories are also useful for KM due to their native similarity to our own mental processes. Bruner (1991) writes of narrative's abilities to facilitate and engage on a cognitive level, citing such properties as normativeness (the ability to mirror reality), canonicity and breach (being able to surprise), and narrative diachronicity (following chronological patterns familiar to the reader from their experiences in the physical world). Others in the cognitive sciences have found that the scriptlike structure of narrative is especially compatible with the ways in which the brain processes information and scans for patterns (Fiore, Johnston, & McDaniel, 2007). Kim (2005) notes that narrative has some additional self-organizing characteristics, which can be helpful when deciding how to represent these structures in a CMS library. These characteristics include

- A reliance on causality and logical sequencing between various events and agent (character) actions, which, when taken collectively, represent the plot of a narrative,
- An anchoring to contextual information that influences both how the story is written and how the story is perceived by an audience,
- An environment regulated by boundaries such as time and place, and
- A definite beginning and ending, which provide impetus and closure for the narrative.

From these features, it is possible to develop several different methods for classifying and organizing this specialized CMS library. Stories can be classified according to environment, such as the time, place, or location of either the recounted events or of the narration itself—the environment of the story or the

environment of its telling. They can be catalogued and sorted according to their obvious plot characteristics or their subtextual elements such as political or moral undercurrents. They may be classified by genre; for example, folklore stories might be separated into stories based around riddles, legends, or superstitious tales. Narrative characteristics provide a rich and varied source of potential organizing methodologies and are well suited for certain types of classification exercises within a CMS.

In the specialized WCMS I created, I chose to classify narrative using five primary features: the central character present within the narrative, the forces (human or otherwise) this character was matched up with and struggled to overcome, the environment in terms of both time and place, and the central theme or concern present in the story. To account for a variety of central themes, or plots, I also included a thematic dictionary system, which included topical listings of keywords. These keywords could then be searched for within narratives and, if a programmed threshold was exceeded, trigger an automatic classification of the story into one of the predefined topics. The final application was named EDNA-E, the Event-Driven Narrative Analysis Engine, and is described in more detail in McDaniel, 2004. Here I provide only a high-level overview of the process in order to relate how closely related the design process lies to fundamental CMS processes and procedures. The WCMS application is available online at: http://www.textsandtech.org/~rudy/edna-e/index.php.

Applying the traditional software lifecycle model (Hamlet & Maybee, 2001) yields five phases of development for a software product: requirements analysis, specification, design, coding, and testing. Developing a robust CMS, then, begins with a clear outline of requirements from a user-centered perspective. What exactly will users need to accomplish with this system? How will administrators enter data, and how will visitors or regular users then access it? Next, during specification, user requirements are implemented according to current process-oriented and technological capabilities. For instance, decisions must be made as to how the interface will function on a technological level and how the database and server will communicate with one another based on existing protocols and software configurations. During design and coding, the system is gradually transformed first into a formalized model, in the design phase; and then into a working prototype, during the coding phase. Testing then occurs to refine the system, and the cycle is repeated as necessary in an iterative fashion. Such development is often known as iterative software development or rapid application development (RAD).

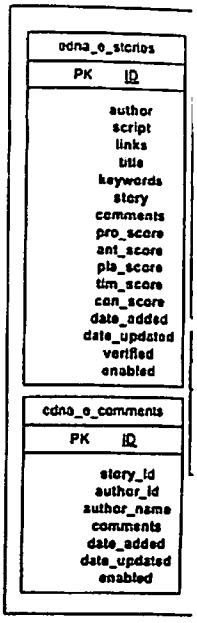
As the content for this particular WCMS would be made up of employee generated stories, there were several considerations that needed to be addressed; namely the issues of scope, security, and quality. Scope concerned the process of soliciting appropriate stories that would be relevant to a given organizational problem, since this narrative WCMS was designed to be used for knowledge management. In other words, if a manager or subject matter expert were

interested in reading stories pertaining to experiences with outsourcing or globalization, then employees who were instead writing stories about technological issues would not be contributing valuable data. Scope, security, and quality were therefore driving factors throughout the development of the product and led to the creation of features such as

1. A *permissions system* to ensure that stories could be accessed by only those users with appropriate permissions or organizational status. This was used to tighten security in the WCMS.
2. An *event generation system* used to produce a database of organizational events open for soliciting stories at any given moment. In other words, only approved organizational events could be responded to by a contributing author. Sample events would include tasks related to major organizational change such as the adoption of new technologies, relocation to a new geographical environment, or perhaps the launch or release of a new product or documentation set. This was used to provide focus and control scope in the WCMS.
3. A *scripting and story administrator panel*, which was used to both define parameters upon the process of contributing new stories as well as to evaluate, edit, or delete stories already submitted. This allowed a "story administrator" to enforce some measure of quality control upon the WCMS.

With the narrative system described here, I began development with a requirements analysis phase, during which I assessed and attempted to anticipate user needs. The database structure, or library, for this WCMS is composed of nine individual database tables (see Figure 2). The tables collect and store information related to both the content itself and the users and administrators who interact with and manage the system. A total of nine tables were designed in order to collect various types of data necessary for successfully soliciting and distributing narrative materials. These nine tables contained several important subsets of data that were critical for the operation of the WCMS:

- A *stories* table was used to encapsulate user-contributed stories and contained additional keyword metadata useful for searching.
- A *comments* table enabled user-contributed comments for particular stories based on reader interactions with stories.
- Both contributors and readers were assigned to the *users* and *groups* tables, allowing for complex permissions to be assigned according to different schema (organizational departments can be broken into different groups, for example).
- The *scripts* table was used to store data that communicated to a potential story author how their story should be created. Scripts were controllable in



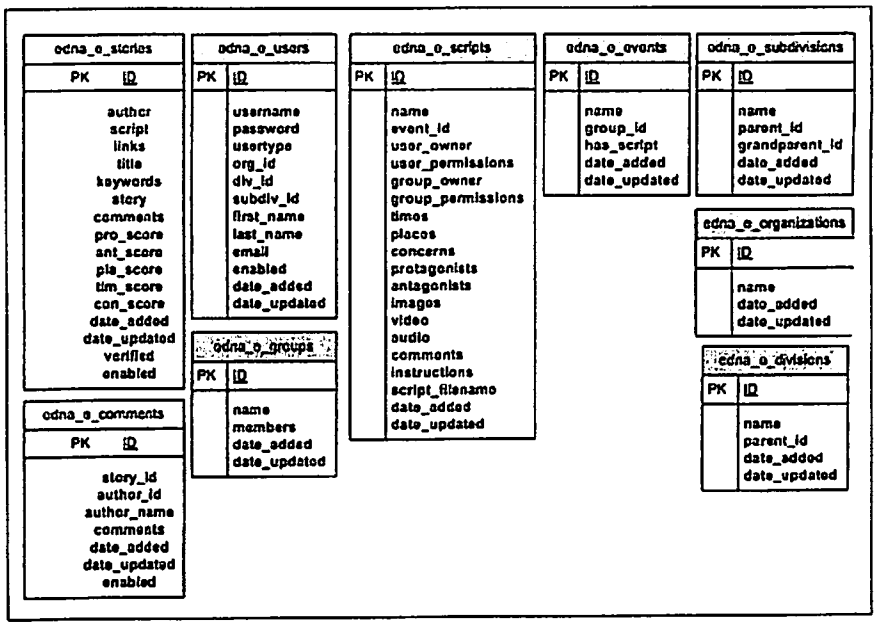


Figure 2. Narrative CMS library configuration.

terms of protagonist (what perspective should the story be told from?), antagonist (what is the opposing force this central character is dealing with?), environment (when and where did the story take place?) and central theme (what is the primary plot of this story and how does it relate to the organization?).

- The *events* table functioned as a library for organizational events and was queried during the story contribution process in order to find an appropriate topic and script.
- The *subdivisions*, *organizations*, and *divisions* tables were primarily used to support the permissions system in enforcing security for access to particular narrative objects. These default tables can be used to automatically generate groups and users based on their membership with default groups in the organization (such as the departments or divisions in which they worked).

Narrative scripts were created based on Schank's (1990) work with narrative and cognitive structures as well as Minksy's (1985) ideas about terminals, which are generic memory structures that he theorizes are filled in with specific details as we encounter variations of these objects in the real world. His idea of a story terminal, then, is a generic framework for a story with placeholders for protagonist, antagonist, place, time, and concern; this directly informed the

construction of the scripting mechanism during the design and coding phases of WCMS development.

Finally, after the database had been implemented and populated, a secondary level of encoding was imposed upon the data using the eXtensible Markup Language (XML). XML, which is useful for documentation specialists in a variety of applications (Applen, 2002; Johnsen, 2001), provided a metadata framework to classify stories and dictionary items in a semantic fashion. A simple Web-based XML editor for configuring script files and editing thematic dictionaries was also included for convenience. Screenshots from the final application, which reveal selected components of both the content library management system and the administrative management interface, are shown in Figures 3 thru 10.

So far we have explored specialized Web content management systems from both theoretical and technical perspectives. What is particularly interesting about such systems, though, is that they also have uses as pedagogical tools.

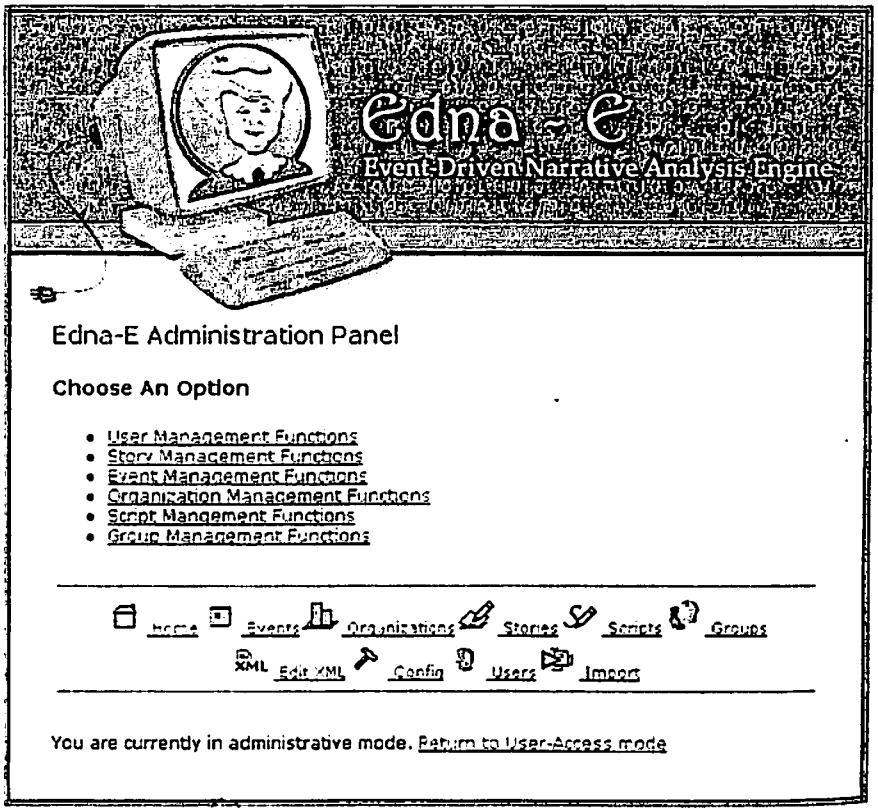


Figure 3. Administrative control panel.

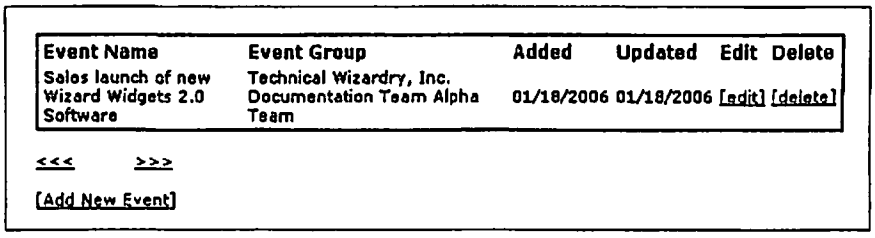


Figure 4. Event management panel.

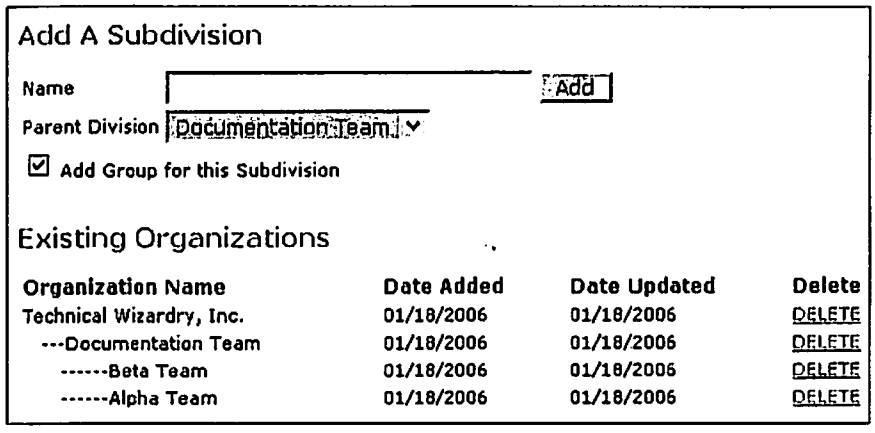


Figure 5. Organizational management panel.

In Part III, I discuss several ways in which designing and evaluating specialized CMS applications can be incorporated into targeted learning objectives for the classroom.

PART III: LEARNING ENVIRONMENT (CONNECTIONS TO THE CLASSROOM)

Content management systems have interesting properties that make them suitable for certain types of learning, particularly in those courses dealing with information complexity, software documentation, usability, or project management. For one thing, they are technologically sophisticated and include many of the same types of technologies that graduating students will encounter in industry. For another, they provide an example of metadata classification systems at work, and they operationalize a concept that is oftentimes difficult for students to grasp without seeing it in action. Lastly, the ability of CMS implementations to

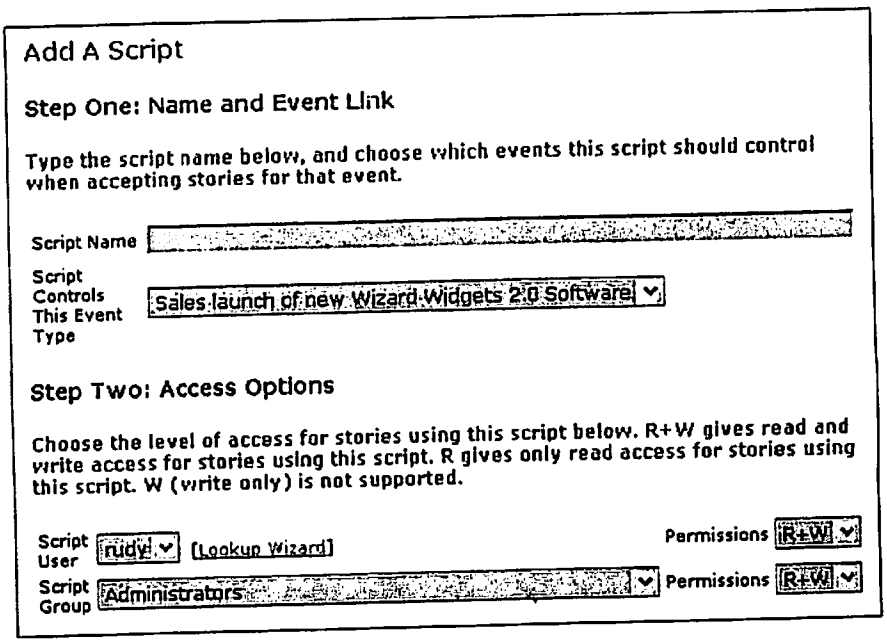


Figure 6. Scripting panel.

cater to a wide variety of audiences and to meet varied informational needs using a single centralized data source seems representative of the trends in which practicing technical communicators and multimedia writers find themselves immersed in within industrial practice: trends like single sourcing and modular documentation designs.

The most challenging aspect of considering a CMS as a classroom tool is devising a framework for managing the complex technologies that make this type of system possible. It is relatively easy to see the benefits of having students design and construct specialized systems of their own: they learn about user-centered design, data flow, process management, audience analysis, and computational data structures, to name just a few topics, all in an operationalized context. Such tasks can be completed either individually or in group scenarios. It is very difficult, however, to provide students with a general framework for building such a system without investing large amounts of time and effort into the process of building a customized delivery system that needs only minor adjustments by each individual student or group of students. Even open-source portal software, freely available on the Internet, is often cumbersome to install, configure, and manage.

A more feasible goal, then, is to divide the construction of a hypothetical content management system into several smaller assignments, or perhaps into a

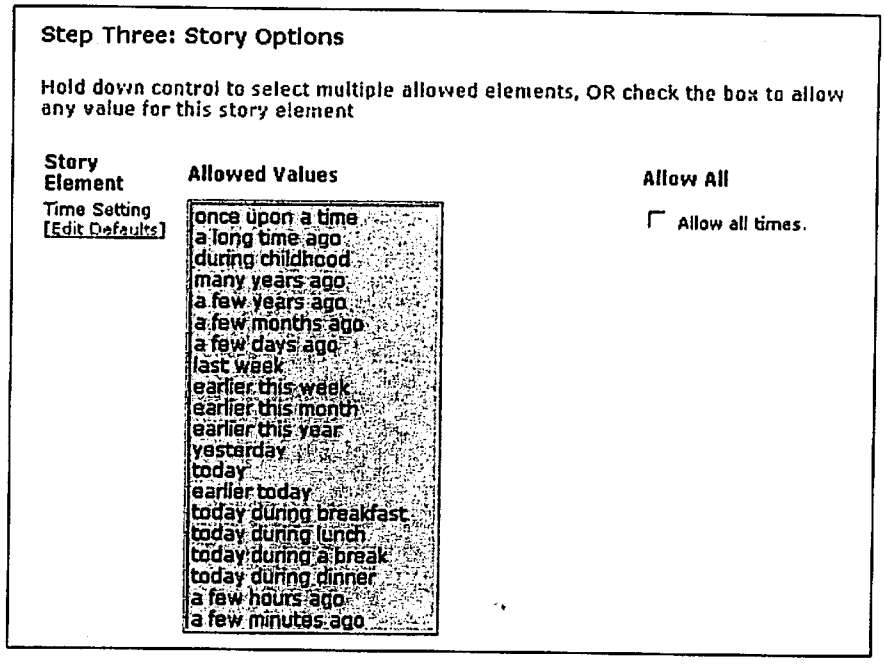


Figure 7. Scripting panel (time setting).

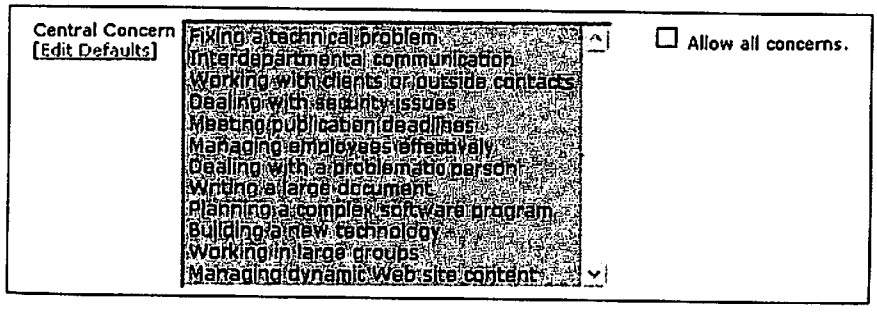


Figure 8. Scripting panel (central concern).

final group project for a semester-length computer software documentation course. Such division mirrors the decomposition model found in the software engineering process, wherein five or more phases occur during the completion of a large, team-produced software deliverable. During each phase, deliverables, or artifacts, are produced that reflect the progress of the CMS at that particular point in development. This division of process can be particularly useful for teaching

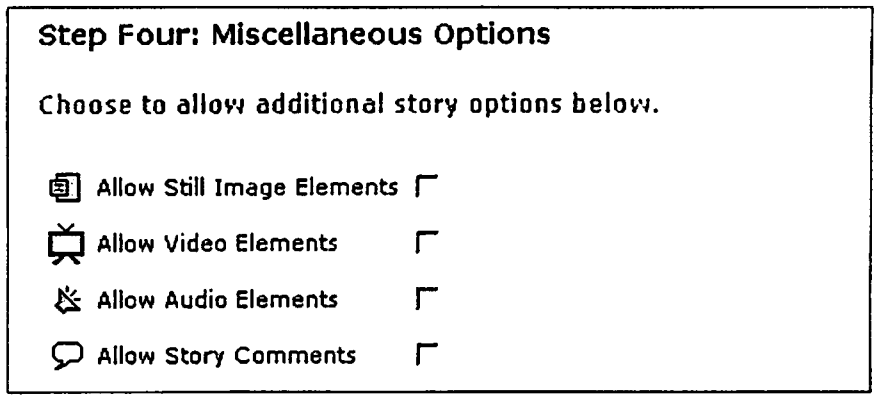


Figure 9. Scripting panel.

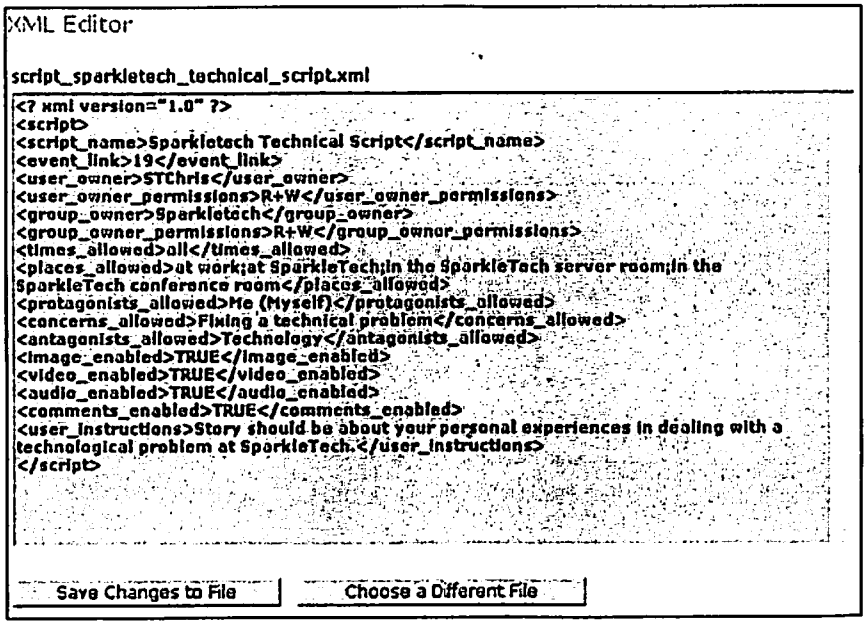


Figure 10. XML editor with sample script loaded.

fledgling writers about the documentation or design process. Since each phase of development is concerned with different primary audiences, writing tasks will vary greatly from one phase to another due to the different informational needs of these audiences.

Table 1 shows one of several possible decompositions of a CMS development process in terms of potential writing tasks. These types of tasks are suitable for use in an upper-level computer documentation or writing for media course. In this table, development phases are indicated along with their primary milestones and some sample writing tasks. In addition, sample audiences are provided for each row and can be used to specify or customize the particular types of writing assignments requested for each phase.

An obvious benefit of this particular exercise is that it includes multimedia authors and technical communication students in the beginning of a complex design process rather than simply handing off a previously completed design to students and asking them to produce its documentation. This facilitates a process in which all team members are involved in all phases of development, and it minimizes confusion about how the product works or how it was designed during latter phases of production. This in turn leads to a truly reader-centered dynamic, in which writers can anticipate how a user or content contributor might feel when encountering the CMS for the first time. Such reader-centered expertise is critical for the development of both software and documentation that are more user-friendly as compared with items produced in a scenario in which documentation and design are fully independent.

An even more ambitious exercise would be to pair a software documentation course with a computer science, digital media, or engineering course in which the

Table 1. CMS-Related Assignments Useful for Software Documentation Courses

Phase	Documentation milestones	Document artifacts	Audience(s)
Requirements	Achieve shared understanding between development team and client	Requirements analysis report	Client/Customer Design Team
Specification	Make technology choices based on capabilities and experience as well as on audience characteristics	Audience analysis, competition analysis	Design Team Project Manager
Design	Create blueprints for software model	Modeling documents and feature lists	Design Team Project Manager
Coding	Develop a working prototype from the abstract design	Prototype of software documentation	Design Team Project Manager
Testing	Refine prototype based on usability and quality assurance testing	Release version of software documentation	Design Team End User

software is fully programmed and implemented. This could be done either for a short period of time (perhaps for a final project) or in a full semester cooperative model, in which instructors collaborate throughout the development process. In either scenario, each course would exchange deliverables periodically—documents created by students from the writing course would go to the programmers for implementation and analysis, and beta software coded by the software engineering students would be delivered to the writing students for documentation.

While there is much potential for using a CMS in this type of software documentation course, these types of courses constitute only a small subset of what is generally available in technical writing and digital media departments. In more general types of courses, and in introductory technical communication courses, incorporating a WCMS into an existing course curriculum is more difficult, but not impossible. By separating the content of such courses into a set of modules, and by further imposing a hierarchical ranking of topics upon each module, it is possible to devise a simple pedagogical matrix that can be used for generating course ideas related to a given topic.

An example of this framework is shown in Table 2. In this structure, categories have been defined that list important skills and learning objectives one might find in an introductory technical communication or writing for media course. Sample categories for this table are defined for theoretical, technological, and professional development types of learning materials. Tiers are numbered according to increasing task complexity and represent various entry points for integrating traditional assignments with CMS development tasks. Each tier includes a sample (and arbitrary) task, topic, or activity related to a broader category of technical communication and could easily be expanded in either direction depending on the needs and objectives of a particular course. These tasks can then be wrapped around the design or *evaluation* of a CMS or WCMS application.

It is worth mentioning that Table 2 is not intended to represent discrete skillsets or categories; writing can be considered a type of technological skill, for instance, and knowledge of the eXtensible markup language (XML) could also be considered as representing a type of writing skill. XML might also be used to demonstrate an application of the theoretical topics related to the granularity or fragmentation of data in a CMS. "Theoretical topics" might be better described as "advisory frameworks," and so on. The idea is not to provide a rigid rubric for relating classroom activities to the design and production of content management systems, but rather to suggest some ways in which humanities expertise might be applied to the design of such systems by breaking down skills into certain topics related to the field that are especially pertinent. Many of these topics also generalize well to the broader discipline of software design. Additional categories could also be generated rather easily depending on the nature of the course (e.g., a category not mentioned here is business management skills,

Category
Theoretical topics
Writing and rhetorical skills
Technological skills
Professional development skills

Table 2. Skills Involved in CMS Design

Category	Topical task areas for generating classroom activities
Theoretical topics	Tier 0: Reader-centered design Tier 1: Audience analysis Tier 2: Granularity and fragmentation Tier 3: Knowledge models (logocentrism, constructivism) Tier 4: Digital rhetoric
Writing and rhetorical skills	Tier 0: Content generation Tier 1: Standardization of content and editing techniques Tier 2: Evaluating source credibility in WCMS Tier 3: Modular writing techniques Tier 4: Writing for extreme programming (XP)
Technological skills	Tier 0: Brainstorming and modeling tools in Word Tier 1: Working with collaborative writing software Tier 2: Database schema and design Tier 3: Producing eXtensible markup language (XML) Tier 4: Translation of unified modeling language (UML)
Professional development skills	Tier 0: One minute oral reports on brainstorming ideas Tier 1: Hybrid teams and team writing Tier 2: Analysis of job search tools as WCMS

which would include digital asset management, or the economic implications of reusable and accessible digital content, as a potential topic area).

An example of a Tier 4 theoretical topic that has some connections to CMS design is the branch of inquiry that has come to be known as "digital rhetoric." Zappen (2005), discussing Manovich's (2001) work, writes of the flexibility of new media methodologies when combined and arranged using online databases. He writes that new media "can appear in different versions (variability) so that a media database, for example, can produce an almost infinite variety of end-user objects, which can be customized for different users, manipulated through hyperlinks, periodically updated, and scaled upon demand" (p. 321). As this is precisely the type of scenario made possible by a WCMS, what better way is there to familiarize students with the potential power of new media configurations? A classroom discussion of digital rhetoric as it applies to WCMS technologies is likely to be interesting and productive, especially if students are asked to bring in several examples of WCMS technologies that they use on a daily or weekly basis. Discussions can then range from the properties of WCMS to the rhetorical characteristics that enable them to shape the same source of content for different audiences.

A Tier 4 technological skill would be translating unified modeling language, or UML (see Stevens & Pooley, 2006), documents from conceptual diagrams, such as use cases, activity diagrams, and sequence diagrams, into textual descriptions and features lists. These lists are important as they are the software design equivalent to the executive summary in the business world, enabling administrators to quickly absorb the most important parts of the design without wading through pages of documentation. In addition, translation allows technical concepts to be understood by other design team members not familiar with UML syntax. These individuals would include project managers and those individuals responsible for meetings with clients. Such descriptions might even reach end users or those responsible for adding content to the system, if the particular design features are stable enough to make it into the final release of the CMS. This skill is particularly valuable for all types of software documentation, and there is likely to be a significant increase in quality and disambiguation if technical writers and multimedia authors become attached to these types of design phase procedures.

An interesting exercise is to pair the most complicated theoretical exercises with the least complicated technological exercises during early development phases, then gradually adjust the intensities of each to maintain a stable level of difficulty throughout. For example, during the requirements phase, students can be asked to apply their knowledge of constructivist theories in order to produce a blueprint for a CMS design that will account for the needs of a wide variety of different discourse communities and subject matter experts. In this phase, a very low level of technological understanding is necessary, and ideas can be communicated using simple brainstorming software such as *Inspiration* or *Brainstorm* or even typed into a Word document or Excel spreadsheet for comparisons. As the designs move from the abstract realm to more specific ideas, the level of technological complexity will increase, but the theoretical material has been previously applied and can be tapered off or used in a postproduction context in order to produce evaluative usability feedback. By balancing the degree of technological difficulty with the degree of theoretical complexity for any given activity, the tool remains useful, but it is also easier to access and apply by students and to manage by instructors.

The activities shown in Table 2 do not take into account other ways in which CMS or WCMS technologies might be used in the classroom. For example, an instructor teaching in a multimedia classroom might find it meaningful to bring up examples of genre-specific CMS tools in order to demonstrate particular concepts such as information complexity, usability (or the lack thereof) of engineering systems, or concrete examples of single-sourcing methodologies that are used in industry or in open source applications. This would provide an example to *show* the types of systems being discussed. Evaluating existing CMS technologies therefore generates the opportunity for additional classroom connections in terms of classroom participation (discussion and evaluation), writing assignments

(rhetorical criticism or evaluative essays), and even field work (usability studies or interviews with CMS personnel).

In my own teaching, I have used CMS examples in several different courses with varying degrees of success. As I am appointed both in an English department and in a digital media department, I generally teach at least one technical communication course each semester along with one or more digital media design courses. In English, I teach both introductory and graduate level technical communication courses, and in digital media, I teach courses in Internet software design, media for e-commerce (which focuses more on the application of technologies rather than on electronic business models), and interactive media.

At our university, digital media students, who come from humanities backgrounds and rarely have any type of background in engineering or computer science, have proven quite adept at designing and implementing their own CMS solutions. These students generally enjoy working with both artistic and Web-related content and appreciate the flexibility offered by CMS architectures in allowing them to create their own dynamic Web sites based on their own academic and professional interests. For example, in a course entitled "Media for E-Commerce II," offered in the spring semester of 2006, students designed CMS-like applications that we called "information fluency kiosks." In this particular course students were exposed to design tools such as Visual Studio .NET and PHP, and although most of these students did not emerge from programming backgrounds, they were able to develop skills sufficient to create basic CMS applications of their own. They could choose their own content for their information fluency kiosks as long as their designs met three objectives: they were required to be educational in some fashion, they were required to be interactive and to allow users to add content, and they were required to use database-driven technology.

These student-developed CMS kiosks were designed to draw from work done in the library sciences (primarily searching and database taxonomy) and digital media (interactivity and graphic design for interfaces) and used specialized content collections for pedagogical purposes. One student designed a CMS for students of beginning Spanish; the system would present data from the library in the form of virtual flash cards in which definitions and conjugation information could be layered and selectively shown based on whether or not a user was interfacing with the system in a "studying" or "quizzing" mode. Several others designed database-driven CMS wikis; rather than attempting epistemological ubiquity, though, they instead focused on specialized subsets of information specific to groups of students (film terminology or collections of tips and tricks for specific software applications, for example).

Using open-source programming and database languages such as PHP (http://www.php.net) and MySQL (http://dev.mysql.com/), students were able to produce and document these CMS applications over a period of four weeks

(Figures 11–14). While these applications are likely not full fledged WCMS applications due to their lack of advanced indexing and search features, they do demonstrate the basic characteristics of such systems in terms of database-driven design, interactivity, usability, and interface design.

In my English courses, I have had better luck dividing CMS design procedures down into smaller assignments as opposed to asking students to participate in building an entire system (although I still contend that this can be done and with valuable results). In a graduate-level professional writing course, one of my modules focuses on the use of XML in technical communication, and students are given the opportunity to examine modules of information (taken from a CMS and encoded in XML) in terms of their rhetorical efficacy. In another module, students are given the choice to either design a Web site system adhering to the principles of user-centered design or to evaluate an existing site through a rhetorical lens. Many students choosing the latter option have honed in on large commercial Web sites and examined the ways in which source credibility can be even further diminished by content management systems (see Warnick's 2004 discussion of online ethos and the complications involved with source assessment in "authorless" environments).

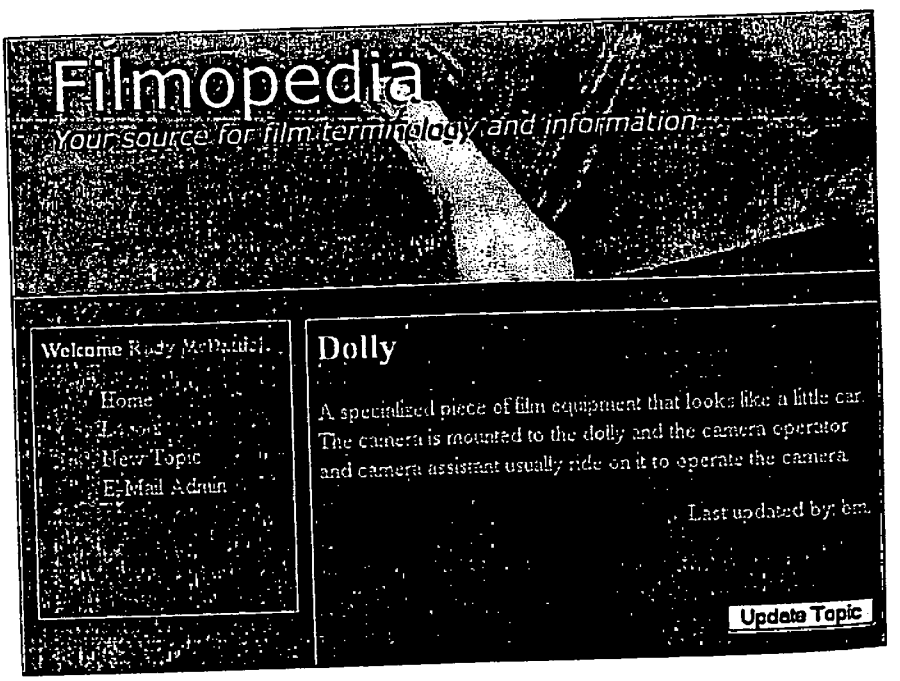


Figure 11. Student produced film terminology wiki CMS.

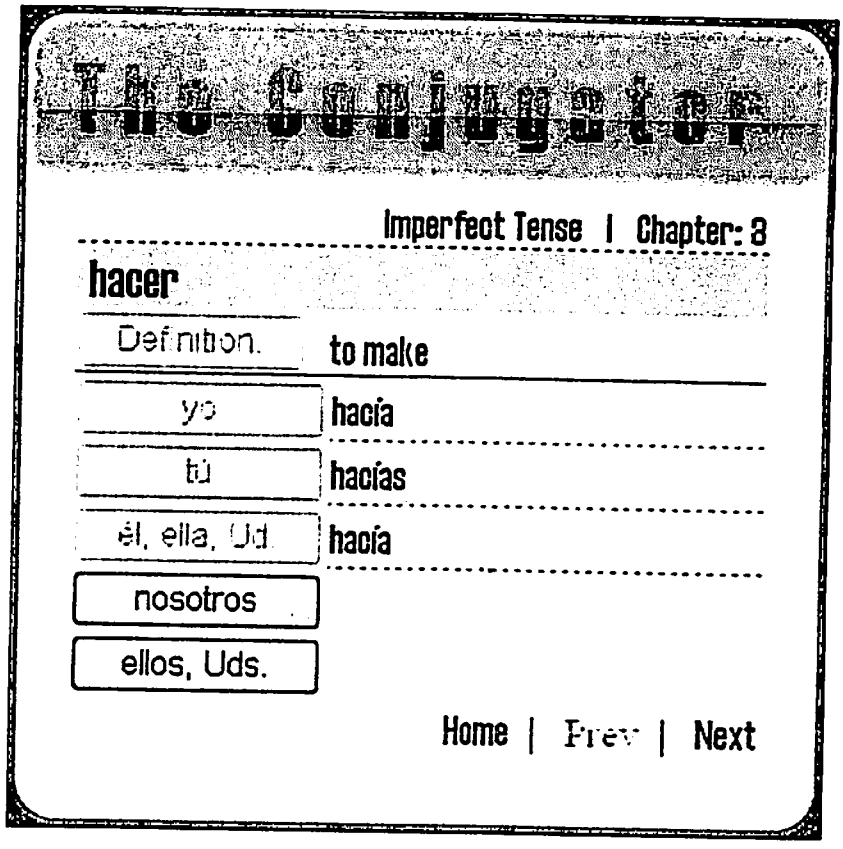


Figure 12. Student produced Spanish language study tool CMS.

CONCLUSION: IMPLICATIONS FOR WRITING IN NETWORKED ENVIRONMENTS

An understanding of specialized CMS processes and architectures is essential for technical communicators and multimedia authors hoping to position themselves as knowledge managers—or simply as effective and technologically literate documentation specialists—within their organizations. Luckily, such expertise is not difficult to obtain. As Hughes (2002) notes, those technical communicators who routinely practice user-centered writing are already working within a knowledge domain rather than an information-centric domain; he writes, "By reinterpreting technical information in user contexts, they are creating new knowledge by presenting that information in actionable terms and relating it to

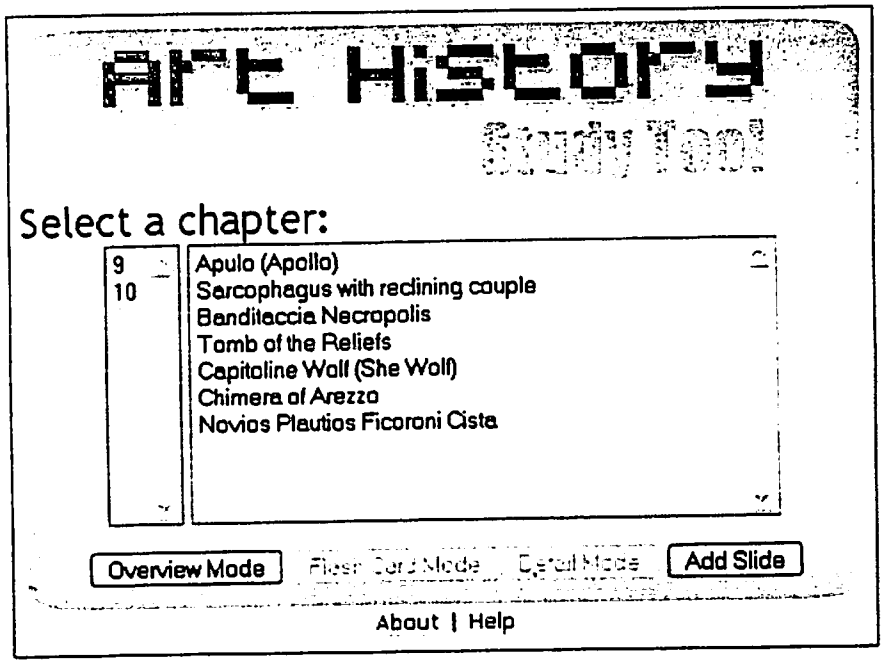


Figure 13. Student produced art history study tool CMS (chapter view).

specific applications" (p. 276). By guiding the types of automated reinterpretation being done by CMS tools, technical communicators contribute valuable usability expertise to this process.

Likewise, the experience gained from working with CMS and WCMS technologies is highly desirable for those students studying in digital media and new media programs. By engaging with CMS frameworks in both directions—forward when designing and in reverse when evaluating—students are given the opportunity to interact with digital assets in a dynamic fashion while learning about critical concepts such as genre analysis, information granularity, modular design, and digital asset management. As digital media itself is such a new and emerging field—digital media programs across the country are being formed from programs as diverse as English, engineering, and art—CMS technology is rich enough to be studied from *all* of these perspectives, yet straightforward enough so as not to be overly intimidating and inaccessible.

Instructors also benefit from knowledge of CMS technology by better positioning themselves for interdisciplinary collaboration; such collaborations have the potential to yield measurable and tangible results. With a computer science and technical communication course partnership, for example, computer science

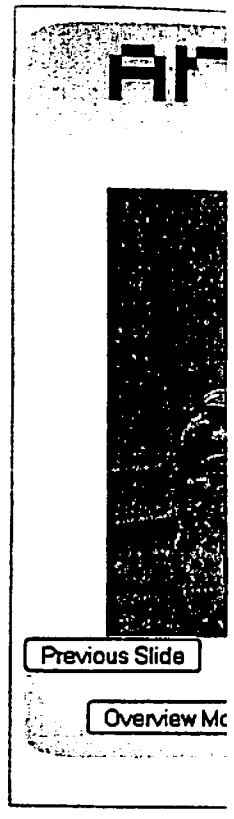


Figure 1

students are likely to
while technical comn
in a project that is si
Faculty from both d
by other disciplines.
see value in the coll
writing skills of thei
administrators can h
problems in the class

In addition to indu
handy for use in the v
with access to those
of data and corporate

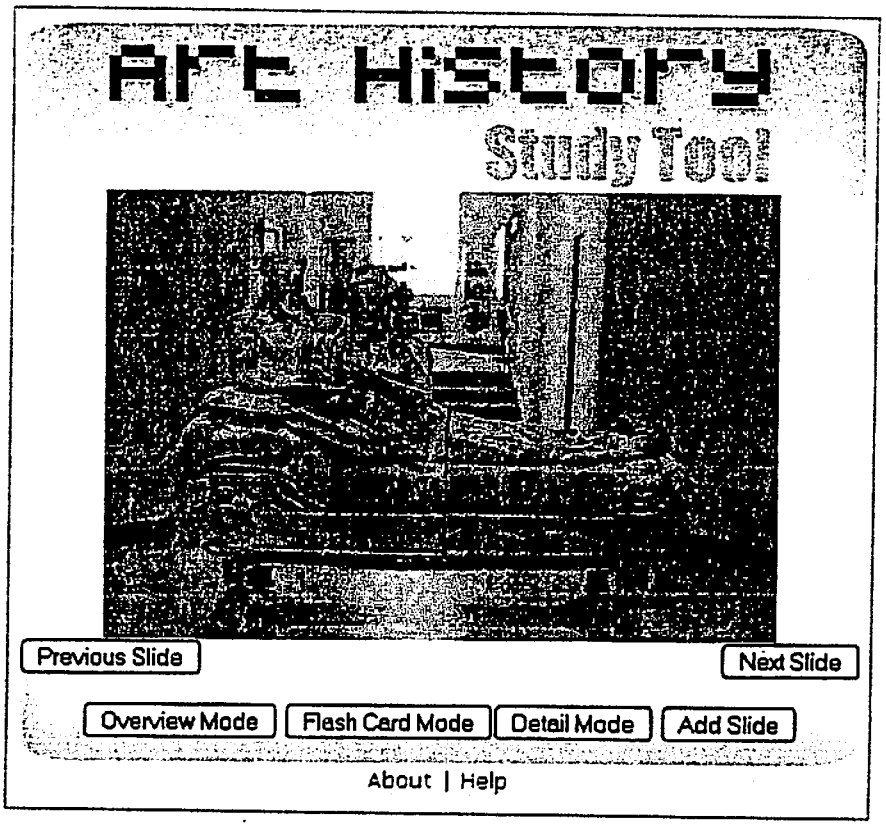


Figure 14. Student produced art history study tool CMS (image view).

students are likely to appreciate the assistance with documentation and planning, while technical communication students will appreciate the chance to be involved in a project that is similar to what they will find in industry when they graduate. Faculty from both departments will appreciate the skills brought to the table by other disciplines. Administrators from cooperating departments may also see value in the collaboration; computer science Chairs often bemoan the poor writing skills of their students while technical communication and digital media administrators can have a difficult time replicating real-world technological problems in the classroom.

In addition to industrial utility, a knowledge of CMS applications can also be handy for use in the writing or communications classroom. By providing students with access to those principles guiding the development of industrial collections of data and corporate assets, these students are likely to recognize the value and

necessity of fundamental communication tasks as they are applied to a more fully realized development cycle. Incorporating CMS-related activities into a technical communication, digital media, or information technology course is likely to generate at least two levels of benefit: at the short-term level, students will be exposed to complex technologies and gain some familiarities with the inner workings and applications of such technologies. At a more sustained (and arguably more valuable) level, students will also engage with these technologies on a critical level in order to reexamine and question the ways in which information technology networks operate in broader discursive contexts.

As we slowly but inevitably move toward the next-generation Internet technologies such as Web 2.0 and the Semantic Web, information overload will continue to be a very real problem, perhaps *the* problem professional communicators and managers of complex and specialized information are most unprepared to deal with. Technologies such as intelligent agents (Quesenbery, 2002; Soller & Busetta, 2003) may eventually mature to the point of viability as they take up residence in the pathways between interfaces and content databases and help users to construct accurate and focused searches through vast libraries of information. Regardless of new technologies, though, it is likely that students and faculty exposed to CMS tools and technologies in *any* context will be better prepared to deal with the continued growth and complexity of not only the Internet, but of distributed information systems in general.

ACKNOWLEDGMENTS

The author would like to thank several students in the Spring 2006 Media for E-Commerce II course at the University of Central Florida for agreeing to share their information fluency kiosks. In particular, Bryan Miller designed the kiosk shown in Figure 11, Paul Cox's work is shown in Figure 12, and the screen captures in Figures 13 and 14 are the work of student Gabriel Mariani. He also wishes to thank Carole McDaniel for her graphic design work on EDNA-E and her assistance in preparing Figures 1–10.

REFERENCES

Applen, J. D. (2002). Technical communication, knowledge management, and XML. *Technical Communication, 49*(3), 301-313.

Bal, M. (1997). *Narratology: Introduction to the theory of narrative* (2nd ed.). Toronto: University of Toronto Press.

Bruner, J. (1991). The narrative construction of reality. *Critical Inquiry, 18*, 1-21.

Davenport, T. H., & Prusak, L. (1997). *Information ecology: Mastering the information and knowledge environment*. New York: Oxford University Press.

Deacon, A., Jaftha, J., & Horwitz, D. (2004). Customising Microsoft Office to develop a tutorial learning environment. *British Journal of Educational Technology, 35*(2), 223-234.

Denning, S. (2001). *The springboard.* Woburn, MA: Butterworth-Heinemann.

Denning, S. (2004). *Squirrel Inc: A fable of leadership through storytelling.* San Francisco, CA: John Wiley & Sons.

Farkas, D. K. (2005). Explicit structure in print and on-screen documents. *Technical Communication Quarterly, 14*(1), 9-30.

Fiore, S. M., Johnston, J., & McDaniel, R. (2007). Narratology and distributed training: Using the narrative form for debriefing simulation-based exercises. In S. M. Fiore & E. Salas (Eds.), *Where is the learning in distance learning? Towards a science of distributed learning and training* (pp. 119-145). Washington, DC: American Psychological Association.

Genette, G. (1980). *Narrative discourse* (J. E. Lewin, Trans.). Ithaca, NY: Cornell University Press.

Goans, D., Leach, G., & Vogel, T. M. (2005). Beyond HTML: Developing and re-imagining library Web guides in a content management system. *Library Hi Tech, 24*(1), 29-53.

Hamlet, D., & Maybee, J. (2001). *The engineering of software: Technical foundations for the individual.* Boston, MA: Addison Wesley.

Hughes, M. (2002). Moving from information transfer to knowledge creation: A new value proposition for technical communicators. *Technical Communication, 49*(3), 275-285.

Johnsen, L. (2001). Document re(presentation): Object-orientation, visual language, and XML. *Technical Communication, 48*(1), 59-66.

Kim, L. (2005). Tracing visual narratives: User-testing methodology for developing a multimedia museum show. *Technical Communication, 52*(2), 121-137.

Manovich, L. (2001). *The language of new media.* Cambridge, MA: MIT Press.

McDaniel, T. R. (2004). *A software-based knowledge management system using narrative texts.* Orlando, FL: University of Central Florida.

Minsky, M. (1985). *The society of mind.* New York: Simon & Schuster.

Orr, J. E. (1996). *Talking about machines: An ethnography of a modern job.* Ithaca, NY: ILR Press.

Post, T. (2002). The impact of storytelling on NASA and Edutech. *KM Review, 5*(1), 26-29.

Quesenbery, W. (2002). Who is in control? The logic underlying the intelligent technologies used in performance support. *Technical Communication, 49*(4), 449-457.

Schank, R. C. (1990). *Tell me a story: Narrative and intelligence.* Evanston, IL: Northwestern University Press.

Smart, G. (1999). Storytelling in a central bank: The role of narrative in the creation and use of specialized economic knowledge. *Journal of Business and Technical Communication, 13*(3), 249-273.

Soller, A., & Busetta, P. (2003). An intelligent agent architecture for facilitating knowledge sharing communication. *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems* (94-100).

Stevens, P., & Pooley, R. (2006). *Using UML: Software engineering with objects and components* (2nd ed.). New York: Addison-Wesley.

Surjanto, B., Ritter, N., & Loeser, H. (2000). *XML content management based on object-relational database technology.* Paper presented at the First International Conference on Web Information Systems Engineering.

Warnick, B. (2004). Online ethos: Source credibility in an "authorless" environment. *American Behavioral Scientist, 48*(2), 256-265.

Wick, C. (2000). Knowledge management and leadership opportunities for technical communicators. *Technical Communication, 47*(4), 515-529.

Yu, H. (2005). *Content and workflow management for library Websites: Case studies.* Hershey, PA: Idea Group.

Zappen, J. P. (2005). Digital rhetoric: Toward an integrated theory. *Technical Communication Quarterly, 14*(3), 319-325.

Zimmerman, M. (2001). Technical communication in an altered technology landscape: What might be. *Technical Communication, 48*(2), 200-205.