# Using the Zope Web application framework to build and manage a large encyclopedia of scientific knowledge

JOHN JORDIN JR.*, WILLIAM HUBBARD+, DEBORAH KENNARD‡, WILLIAM MILNOR§, MICHAEL RAUSCHER‖, AND BRYAN VEAL'

## INTRODUCTION

Many social and economic institutions in the Southern Appalachians depend on the various benefits provided by its forests, such as abundant, high-quality timber; plentiful and diverse fish and wildlife; extensive recreational opportunities; and, a variety of nontimber forest products. These benefits take on added value because of their proximity to human population centres and the strong social and cultural heritage of rural and indigenous populations. This region is experiencing increasing pressure to provide this wide diversity of resource values to millions of people.

These socio-economic concerns have driven substantial research efforts in the southern Appalachians. As a result, an overwhelming body of information exists covering many aspects of forest ecosystems in these mountainous areas. For example, the Coweeta Hydrologic Laboratory, established in 1934, has been a centre of forestry research in the region-nearly 900 publications from that site alone had been produced by 1994 (Stickney et al. 1994). Scientists at the U.S. Department of Agriculture Forest Service, Southern Research Station's Bent Creek Experimental Forest, established in 1927, have produced 287 publications. In 1993, Nodvin et al. published a list of some 2500 publications associated with the Great Smoky Mountains National Park. The Southern Appalachian Assessment (SAA) generated nearly 3 gigabytes of information about the status of resources in the southern Appalachians (SAMAB 1996).

Despite the accumulation of this large body of research knowledge, a gap exists between what scientists know and what the management community is able to apply on the ground. Most research knowledge is neither easily accessible nor readily useable because it has not been synthesized and integrated into a coherent, meaningful knowledge structure. In most cases, this knowledge base retains the fragmented nature of the many separate publications that compose it. What should emerge as an integrated and coherent body of knowledge appears instead to managers as disconnected pieces of the "whole" that they need for applied problem solving. In such a situation, knowledge gaps are not easily identified, important knowledge developed a decade or more in the past is unknown, and the interesting, but relatively unimportant, research problems cannot be distinguished from those that are both interesting and critically important.

Because land managers deal with forest resources in aggregate, they need knowledge that captures the integrative nature of ecosystems and management. Moreover, as natural resource management moves from a multiple-resource management paradigm to an even more challenging ecosystem management paradigm, the need for powerful knowledge management aids becomes urgent.

CITATION

Jordin, J. Jr., W. Hubbard, D. Kennard, W. Milnor, M. Rauscher, and B. Veal. 2003. Using the Zope Web application framework to build and manage a large encyclopedia of scientific knowledge. In Natural resources information management forum: Putting knowledge to work. T. Innes (editor). FORREX–Forest Research Extension Partnership, Kamloops, B.C. FORREX Series No. 8. pp. 135–145.

To address this need for more accessible, understandable, and condensed research knowledge, we began the Encyclopedia of Southern Appalachian Forest Ecosystems (ESAFE)-a hypertext-based encyclopedia system available on the Internet. This project aims to synthesize what we know scientifically about the management and ecology of Southern Appalachian forest ecosystems, organize it logically, and make it universally available at no cost to users. The encyclopedia is designed to be dynamic, so that new or revised content can be submitted directly through the Internet, resulting in a continuously updated, expanded, and improved knowledge base.

The main objectives of the encyclopedia are to:

- organize research knowledge about southern Appalachian forest ecosystems;
- present scientific information on the Internet in a form that is accessible to a wide variety of users; and
- provide research information in a form that is easy to apply to the daily work of forest managers, landowners, and researchers.

In this paper, we will discuss the development history and hypertext preprocessor (PHP) infrastructure problems, provide a brief introduction to Zope, and describe how we utilized Zope to create a hypertext encyclopedia with a dynamic content management system. A companion paper (Kennard et al. in press) describes the development of scientific content more fully from an author's perspective.

### Development History and PHP Infrastructure Problems

The current version of the Encyclopedia of Southern Appalachian Forest Ecosystems (ESAFE)[1] was released for evaluation in June 2002. It is composed of original summaries of hundreds of topic areas compiled from thousands of literature sources. Content is organized in a hierarchical format so that each page has only one parent page. It is possible for multiple pages to designate the same page as the parent, thus creating sections or a tree-type data structure.

The infrastructure for this version of the encyclopedia was originally created using static HTML publishing methodology. We accomplished this by using the PHP scripting language (Ratschiller 2000) in combination with the MySQL relational database software to create and manipulate a treelike data structure. We originally planned for a product containing approximately 2000 HTML pages contributed by a small number of authors. The authors created content using the Microsoft@ FrontPage HTML editor. Completed content was then delivered by authors to the project content manager as sections were completed by each author. Sections were integrated into a common structure outline by the content manager and then delivered to the infrastructure manager for inclusion into the on-line system. If content was to be revised or additional material was to be added to a section the same process had to take place. This made progress slow and tedious at best.

The initial release of the ESAFE was evaluated using an on-line survey. This survey asked participants to designate whether they were natural resource professionals and to provide general comments on the achievement of the goals set for the project. Results indicated that users were impressed with the project as a whole and that they would definitely like to see greater scope and more depth of content (Kennard et al. in press). Concerns expressed included missing content, availability of reference citations, how the quality of content was to be assured, and how content would be kept current. A direct result of this evaluation was the decision to expand the ESAFE by adding additional content and to fund the development of a second encyclopedia on "Southern Fire Science."

Plans for expanding this initial ESAFE system immediately brought about the realization that the existing PHP infrastructure was inadequate. Not only that, our entire content creation paradigm of a few

[1] Evaluation version can be found online at: *www.forestencyclopedia.net*

authors feeding content to a single content manager had to be revised. Numerous authors providing a continuous stream of content required a much more efficient content processing system than the one we originally envisioned. Quality control and assurance required scientific peer review of all content to provide the credibility that users were demanding. The organizational structure that gives meaning to the individual content HTML pages would have to be very dynamic and easy to update and maintain. In short, we needed a complete content management system (CMS) with capabilities well beyond the original PHP infrastructure design. After examining many competing approaches, we chose to use the Zope Web application framework (Spicklemire et al. 2001) to develop our second-generation CMS encyclopedia infrastructure.

## AN INTRODUCTION TO ZOPE

Zope is an open source software product that provides a framework to develop Web applications. In this context, a framework is a set of standards and tools for standardized development. A Web application is a computer program that users access with a Web browser over the Internet (Latteier 2001). In contrast to static information, Web applications provide a means for users to interact and utilize various tools built into the application. Common examples of Web applications include search engines such as Google™ or e-mail programs such as Hotmail™. Zope is actually several different things in one package. Zope includes an HTTP, FTP, and Web-DAV server, an object database, and a framework for creating applications.

Zope was created as a new way to structure CGI-based programming envisioned by Jim Fulton, the current chief technology officer of Zope Corporation. The Zope Corporation leads in setting the directional development of Zope, yet Zope has an extensive development community that extends far beyond that of the Zope Corporation (Schmitz 2001).

### Basics of Zope

Zope is primarily developed in the object-oriented programming language python (Learner 2002a). In fact, Zope stands for *Z Object Publishing Environment.* This title describes the core of Zope very accurately. Zope was founded with the awareness that the Web is essentially object-oriented. A URL to a Web resource is fundamentally just a path to an object in a containment hierarchy, and the HTTP protocol provides a way to send messages to that object and receive its response. Zope differs from the traditional Web systems in that content rendering is based on objects and their definitions as opposed to a linear interpretation of static files.

Applications are built with Zope by combining and connecting various objects that are created independently (Learner 2002b). Zope comes out of the box with a means to manage and create sites and objects directly through the Web. The Zope Management Interface, or ZMI, allows for development of sites by multiple authors by simply using a Web browser. By using the ZMI, users can create many types of objects such as folders, files, documents, Python scripts, and much more. Different object types are meant to provide their own distinct functionality. As an example, folders are used to collect and organize other objects, whereas Python scripts are used to provide the ability to program logic. This is a fundamental principle of Zope that allows for presentation, content, and logic of a site to be completely separated. Besides the ZMI, other methods to develop with Zope such as FTP and Web-DAV are supported.

#### *Acquisition*

A cornerstone for understanding the operation of Zope is the principle of acquisition. Acquisition is a type of object-oriented programming similar to that of “inheritance” in other languages, such as Java and C++.

The difference between the two can be simplified by considering inheritance as an "is-a" relationship, but acquisition as a "in-a" relationship. With acquisition, objects can acquire attributes from their containers, even if they don't have those attributes themselves. When you place objects inside of other objects you create much more than a Web site, you are creating an information structure. For example, if a folder has the ability to send e-mail, then any objects that are placed inside of that folder also can send e-mail.

Security

One of the more difficult parts of deploying a Web application is ensuring its security. It is important that users of the Web application can interact in ways that do not compromise the integrity of the application. This is achieved by restricting the areas in which a person can operate and also the functions that they can use inside of each area. Zope provides several tools for providing appropriate access to the server and the content that it manages. The first step in achieving this goal is having a means to distinguish one user from another. Users inside Zope are defined by a user object type that stores a user name and password for each user. User objects are stored inside of objects called user folders. Zope assumes that a connection to itself is anonymous unless the request provides authentication information that Zope can verify consisting of a user name and password. After authenticating a user, Zope regulates security and access to content on an object-by-object basis. This is achieved with the concept of permissions. Permissions govern whether a particular entity can take a specific action. When a user is authenticated inside of the Zope system, it checks to determine if the user object is permitted to perform the action that is requested. It does this by checking to see if the user has access to the appropriate permission that is protecting the action in question.

Hundreds of permissions are defined inside the default installation of Zope. In addition, permissions can be created or extended by adding onto or modifying the core Zope installation. To simplify the management of security inside of Zope, the concept of a "role" was introduced. Roles are simply non-exclusive groups of permissions. These roles are then associated with each user inside of a Zope application. Roles allow you to collect permissions that are common among a group of users inside of your application. Zope has four built-in roles by default: Anonymous, Manager, Owner, and Authenticated. Besides being able to assign roles to a user for the entire site, local roles allow for the assignment of roles to specific portions of a site. A user might be granted the authenticated role for the entire site, but only has the manager role for a portion of the site.

***Content Management Framework***

As Web sites have become mainstream methods for businesses and organizations to provide services and content, traditional means of Web site management have become a very expensive and brittle proposition. The software industry has developed a solution to handle this dilemma with a wide variety of products which are collectively known as content management systems (CMS). Content management systems typically provide a single solution for the creation, management, publishing, and presentation of content. Hundreds of software products are available that can handle anything from small enterprise level applications to massive international corporate needs. With price tags that usually start in the tens of thousands and can soar well into the millions, selecting a system that will meet your current needs while providing the ability for expansion is vital.

The Zope solution for content management is called the Content Management Framework (CMF). The CMF is a collection of Zope products that work together to provide authoring, publishing, and management of custom content types. Built on the basic Zope platform, the CMF extends the functionality of Zope by introducing some new features such as skins, workflows, and portal tools.

Zope provides a few default content types that include document, file, image, and folder. The true power of the CMF is harnessed by creating custom content types. Zope provides an easy to use product called a ZClass that allows developers to create custom products directly through the ZMI by building on the base products defined in the default installation.

Skins are sets of objects called Zope Page Templates that are used to define how the site functions. Zope Page Templates allow developers greater flexibility in separating an application's presentation layer from the business logic that drives it, thereby making it possible to easily update one without disrupting the other. Skins define the presentation and interaction for your CMF site. By creating different collections of skins, you can create entirely different presentations for your site. For example, you might want to have a set of skins that favours low-bandwidth users and one that provides a richer, high-bandwidth environment. Users could decide which set they wish to use when registering with the CMF. Skins also allow you to define common aspects of your site in only one place by using Template Attribute Language Expression Syntax (TALES), a tool implemented within Zope Page Templates.

Portal tools are objects that define how various aspects of the site behave. The membership tool, member data tool, and registration tool are used to control how users register with and interact with the site. The portal catalogue tool defines what objects are stored for searching and the data that is used to search for these objects. Finally, the portal types tool defines what types of content can be created inside of the site and what actions can be performed for each content type.

A workflow is an object that defines the rules for the content publishing process inside a CMF. Workflows govern' what steps must be taken and by whom so that content can be made available to the general public. A workflow is a state machine, in which a content type can only be in a single state at any point and time. Developers can define multiple workflows to use with different content types. This allows for increased restrictions on more sensitive content, while improving the deployment time for content that is less sensitive in nature.

## DEVELOPMENT OF THE ENCYCLOPEDIA USING ZOPE

### Why We Chose Zope

We were deliberate in selecting Zope as the new platform for the encyclopedia system. Our specific requirements for platform performance and abilities combined with limited resources focused our attention on Zope rather quickly. Other solutions were found to be too costly both financially and with respect to the training and support that would be required. After a period of extensive research on the features, limitations, and real-world applications of Zope, members of the development team attended a one-week training session conducted by the Zope Corporation, May 6–9, 2002, in Fredericksburg, Virginia. Already familiar with the basic concepts and skills associated with Zope, the training provided an opportunity to interact with the developers of Zope, which allowed us to address specific concerns and concepts about applying the platform to the encyclopedia system. After the training, we were convinced that Zope could remedy all of the issues that were identified by the evaluation version of the ESAFE.

Zope is an open-source product that allows users to read, redistribute, and modify the source code of software. Most open-source products are available at no cost. Nevertheless, the advantage of using an open-source solution includes much more than cost savings. Allowing the users of an application to contribute to the development extends the possibilities of collaboration and sharing of resources. Our development team was extremely familiar with open-source products having used Apache Web server running on Linux platforms for years, both of which are open source. The fact that Zope was recently converted to an open-source project was seen as beneficial. The ability to access source code would aid in our understanding of the underlying principles of the software and also provide the ability to tailor Zope to fit custom encyclopedia requirements. Our development team members are all experienced programmers. None of us had any previous experience with the Python programming language which presented a real concern. Investigation proved Python to be similar to other languages that we were currently using and that it had a relatively shallow learning curve.

One of our hopes was that by converting the encyclopedia to a content management system, we could extend the topic areas of encyclopedia sets to support the entire forest research community if it was so desired. To do this, a system that could expand as needed and handle very large request loads and data retrieval would be necessary. Zope provides several tools that allow you to assess its performance, including programmatic browsing, logging, and profiling (Bernstein 2002). Zope also provides both an accelerated HTTP cache manager and a RAM cache manager allowing you to save results that are computationally costly to generate and that change in their resulting output infrequently. Finally, the Zope Enterprise Object (ZEO) is a distributive processing system that allows multiple "clients" to a use the same object database. The ZEO is a very powerful tool that can scale to handle the most extreme traffic conditions.

### Encyclopedia Specifications

After the decision was made to develop a content management system with a Zope CMF that would address the issues previously mentioned, the development team agreed on a set of specifications which we wanted the system to implement. Authors would be able to create and edit multiple types of content inside of the system and have the ability to choose when to submit individual pages or complete sections for publishing. The system should handle the entire peer review and publishing cycle in an effective and comprehensible manner. Finally, the system should address any scenario that would jeopardize the integrity of published content.

### The Development Process

To convert the current encyclopedia to the new CMF version, several sequential steps were required. Having a body of data and several existing authors from the ESAFE in place was helpful for testing system design features as they were developed. The development sequence established for the new Zope systems was:

1. develop skins for general presentation and the navigation system,
2. develop custom products for content types,
3. create tools to ensure link and content integrity, and
4. develop workflows to associate with the content types.

A new domain name, *zope.forestencyclopedia.net*, was established for testing and commentary of the new system during its development. An on-line forum was established to facilitate communication about standards and specifications, development progress, and critique of the system among all involved in the project.

Skins

The USDA Forest Service released new templates to use for all Forest-Service-related Web sites during 2002. Fortunately, these templates were designed in such a way that they were easily adapted to work well with the features provided by Zope Page Templates. By utilizing the concept of acquisition, we are able to create entirely new encyclopedias without modifying any existing code by simply inserting new content at the top level of the content hierarchy. The Zope CMF comes with default skins. This simplifies developing new skins because we can use the old ones as a starting template. Altogether new skins were created for custom content types and to aid in certain actions and features unique to the encyclopedia system.

#### *Navigation*

The default skins were customized to provide a dynamic navigation system that would be available system-wide. The navigation structure is dynamic such that new options are created automatically as the structure of the encyclopedia system changes. Users can browse any part of the published encyclopedia

system. By first selecting an encyclopedia from the home page of the site or from the drop down menu found in the top right corner, a list of primary encyclopedia sections will appear below the title of the encyclopedia chosen. Secondary levels of an encyclopedia are displayed directly beneath the primary sections. Any pages that are located beneath the secondary order are displayed in a collapsing tree in the lower left portion of the users screen (see Figure 1).

In addition to reflecting the current structure of the system that allows both anonymous or authenticated users to browse through encyclopedias, for authenticated users an actions menu is displayed that reflects the user's role in relationship to the object that is currently selected or being viewed. If an authenticated user is logged into the system, available actions are displayed directly above the sub-secondary navigation tree. The actions menu is grouped into four possible categories depending on a user's role for the object that is currently selected. These categories are: object, workflow, folder, and global actions. The actions listed under the object category are directly associated with the selected object. Common examples of object actions include comment, view, edit, or metadata. Listings under the workflow category depend on the viewed item's current state in its associated workflow. Workflow actions may include submit, publish, retract, accept, assignment, deny, or many others. Folder actions work on subgroups of objects; that is, they display the objects that are contained by the currently selected item or assign local roles for a section. Finally, global actions provide actions that do not depend on any one object item or type. Undo, messaging, or site configuration are examples of global actions.

### *Custom Products*

Currently, the encyclopedia is composed of one primary content type called an encyclopedia page. Pages, in turn, can contain supporting content types such as images or citations. A default **CMF** provides several content types. We utilized the image type from the default **CMF.** All other content products such as citation or encyclopedia page were created from scratch by the development team.

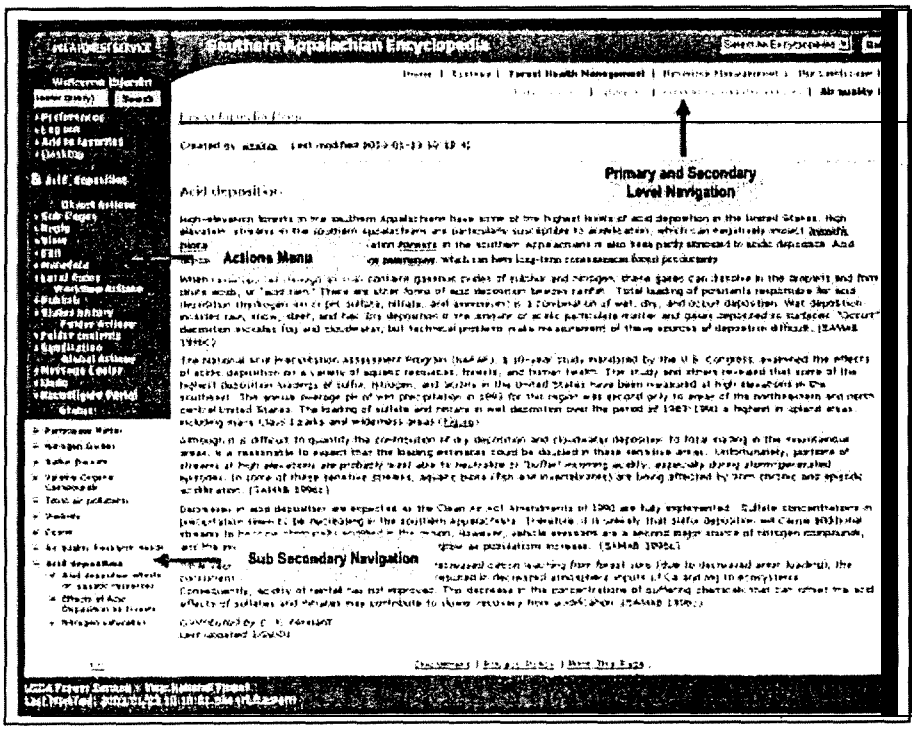


**FIGURE** 1 *Encyclopedia navigation system.*

### *Encyclopedia Page*

As the primary component of the encyclopedia system, pages are the most commonly created content. The encyclopedia page is both a document and folder in nature. A page has the attributes of a document in that it displays text, images, and other content using standard **HTML.** A page also has folder attributes in that it serves as a structuring object that can store other objects. This feature creates the structure and hierarchy for the entire encyclopedia system. Pages can store images, citations, or even other pages. The ability to have page objects contain other page objects is central to several aspects of the system (e.g., the navigation).

Page content is unique in how it is created and edited. By using the Document Object Model developed by Microsoft ® and JavaScript, a Web **HTML** editor was created. This "what you see is what you get," or **WYSIWYG,** editor allows authors to format text, such as setting headings or italicizing text, within a restricted set of **HTML** standards. Authors are not required to know **HTML** to add links to either other published encyclopedia pages or to outside Web sites inside of a page they are editing. The editor also allows authors to insert tables or graphic images into a page.

### *Citations*

One of the features of the encyclopedia system is that citations referenced inside of an encyclopedia page are hyperlinked to the full reference for each citation. To create a citation, authors first choose the type of media they are citing. Based on this decision, the author then inputs values for each variable required to correctly form the citation. The layout or style for each citation type is defined in the class for the citation product. By storing the variables, such as title, author, publisher, or year separately, we have the ability to change from one style guide to another by simply modifying the source code for the citation product.

Authors use a button on the **WYSIWYG** editor to insert citations into their work. Before authors create a new citation to insert, they are forced to search to ensure that the work they are citing has not already been created by another author. If a matching citation already exists, authors can simply reference that object. This reduces the amount of redundancy found inside of the system.

### *Link Management*

One of the strengths of the encyclopedia system is that an author can link to content that has already been contributed and published by other authors. However, this presents a problem when and if that content object is moved thereby breaking links in other objects that link to it. The link management system will ensure that links to all content, including citations, images, and other pages, will always stay current. In addition to assuring that links to internal content are never broken, the link management concept will allow us to implement tools in the future to analyze various aspects of the system such as perceived importance of a page based on how many other pages link to it.

The system revolves around the use of a relational database (MySQL) with one table. The table contains a unique **"ID,"** a "to," and a "from" column. When a user inserts a link to internal content into a page via the **WYSIWYG** editor an entry will be made into the table. The "from" entry will contain the path or location to the item being edited or containing the link. The "to" column will contain the path or location to the item that the link represents. For example, consider a page called *Foo* with links to another page called *Bar; Bar* in turn contains an image called *Baz.* Two entries would be made into the link management database as follows:

| **ID** | To | From |
|---|---|---|
| 1 | /path/to/Bar | /path/to/Foo |
| 2 | /path/to/Baz | /path/to/Bar |

The actual links that are created inside the page being edited refer to a special Python script, called "renderLink," which provides an interface to the database to find the correct content. The renderLink script will look up the "to" field for the given **ID** and use it to redirect to the corresponding content. This way when objects are moved we aren't forced to manipulate countless other objects.

Now let's consider the movement of objects that are referenced by other objects. What would happen if the page *Bar* were moved to another section with the path of "/path/to/some/new/Bar"? This will require the manipulation of the Cut, Copy, and Paste actions in addition to the final steps in the publishing process. These actions will be modified so that a query would be made to replace all instances of "/path/to/Bar," the original path, in both the "to" and the "from" column to the new path "/path/to/some/new/Bar." This would change the way the links would be rendered by using the renderLink script to reflect the new location of the object.

### *Workflows for Content and Peer Review*

Custom workflows allow for peer review of content before it is published and made publicly available. A workflow is associated with each content type that can be created inside of the encyclopedia. The most complex of these workflows is related to the encyclopedia page type. Authors submit pages which will in turn submit any supporting content for that page. Authors must designate their intended location for the page inside of the encyclopedia by noting a parent page when submitting it for publishing. This intended location inside of the encyclopedia is used to designate the appropriate users to review and publish the object. Each step of the workflow is performed by different users based on the roles that they have in the intended destination section. Users are notified by weekly e-mail and with messages left for them in their desktop area when content requires their attention in order to progress in the publishing process.

After the initial step is taken by the author to publish a page, the next step is for the page to be evaluated by the technical editor of its intended section. The technical editor can provide commentary and either reject or accept the authors submission. If, at any point, a page is rejected in-the workflow, it must travel through the entire process again. After a technical editor has accepted the page, the editor for the intended location or section editor, designates users to peer review the document and supporting content for its scientific merit, appropriate style, and form, and whether the intended location is correct. Reviewers are assigned based on their expertise relative to the content in question and also their current workload. After a critical number of reviewers accept the page, it progresses to the final step of editorial board review. The editorial board consists of those users who have an editorial role at the top level of the associated encyclopedia set. All members of the editorial board must accept the page for it to be published. Figure 2 represents a flow chart that illustrates the page workflow as described.

Authors can, at any time, check on the status of objects that are currently within the publishing process. Certain comments are restricted so that reviewers can maintain their anonymity.

## Editing Existing Content and Content Ancestries

Once a component is published, any user will be able to create a copy of the content in their desktop area to make additions or changes. It is important to prevent multiple authors from editing the same content at once, thereby creating problems with different versions. Once a user creates a copy of the page and supporting content for editing purposes, a lock is placed on the published version indicating that it is being edited. Several tools are being implemented to ensure that content is not locked for an extended period of time. Any revisions to content must go through the appropriate channels of peer review before replacing the original content in the encyclopedia structure.

By just replacing current content with edited versions, we lose the ability to document changes in science over a period of time. To preserve and represent this change in knowledge, a historical representation of such changes is needed. We have discussed multiple methods of implementing this feature including, doubly linked objects, a naming scheme, or object variables. The implemented solution was to use the workflow for editing a page and to create another product for the archive called "Page Archive." This product will be almost identical in aspect to the "Page" product. However, certain features will be different, mainly the roles. *Once* a page is archived, it is important that only a few people are able to edit it, if anyone at all. In addition, page archives are omitted from the portal catalogue to make sure they are not returned in search results.

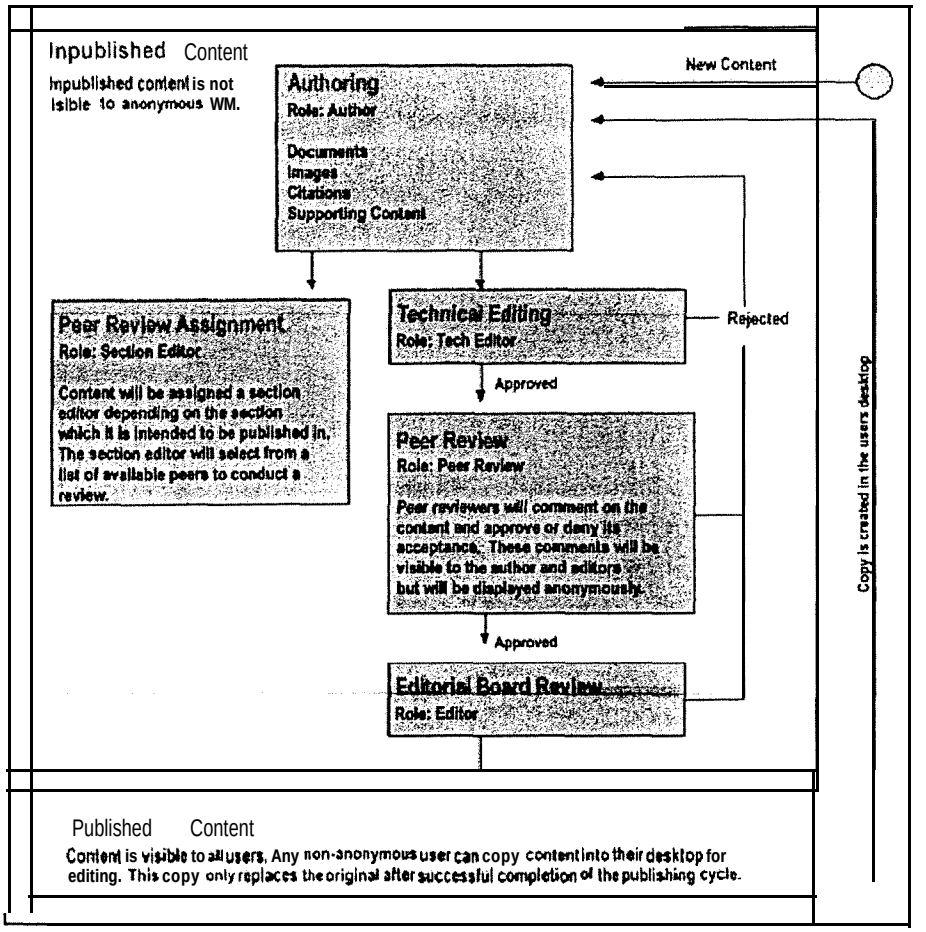


**FIGURE** *2 Encyclopedia page workflow for publishing and peer review.*

Once a page that is intended to replace an existing one completes the peer review cycle, the original page is converted into a "Page Archive" and is contained by the newly approved page. The approved page then takes the place of the original in the encyclopedia structure. When a user views any page that contains an archived version, they are notified by a hyperlink that allows them to view the entire lineage or set of archives for the current page and the date that the archive was created. Our hope is that by browsing this collection of archives, a user can get a sense of the direction of research and the patterns associated with changes in what is known about the topic in question.

## CONCLUSION

The Technology Encyclopedia Development Team of the Southern Regional Forestry Extension Office is currently on schedule to release a beta version of the new Zope CMF-based ESAFE during the spring of 2003. After conducting a series of tests and case studies using existing authors and content, the new Southern Fire Science encyclopedia will be developed exclusively with this new system.

Learning to develop with Zope has proven to be a challenging, yet rewarding, project. A lack of quality documentation regarding the CMF created considerable frustration at times. The questions that documentation did not readily answer were solved by referencing the source code for Zope. In the future, we hope to extend Zope itself and make our contributions available to the general public to allow other groups to build on our success.

## REFERENCES


Bernstein, M.R., S. Roberston, and Codeit Development Team. 2002. Zope Bible. Hungry Minds Inc., New York, N.Y.

Kennard, D.K., H.M. Rauscher, P.A. Fleebe, D. Schmoldt, J.B. Jordin, W.G. Hubbard, and W. Milnor. [2003]. A prototype of an online scientific knowledge management system. Forest Ecology and Management. In press.

Latteier, A. and M. Pelletier. 2001. The Zope Book. New Riders, Indianapolis, Ind.

Learner, R. 2002a. Introducing Zope. Linux Journal 94:20–23.

_____. 2002b. Zope products. Linux Journal 95:14–18.

Nodvin, S.C., J.S. Rigell, and S.M. Twigg. 1993. An indexed reference database of the Great Smoky Mountains, North Carolina and Tennessee. National Park Service, Southeast Region, Technical Report NPS/SERGRSM/NRTR-93/08. NPS-D-413.

Southern Appalachian Man and the Biosphere (SAMAB). 1996. The Southern Appalachian Assessment summary report. U.S. Department of Agriculture Forest Service, Southern Region, Atlanta, Ga. Reports 1-5.

Ratschiller, T. and T. Gerken. 2000. Web application development with PHP 4.0. New Riders, Indianapolis, Ind.

Schmitz, J. 2001. Statute of the EuroZope Association. URL: *www.eurozope.org/statutes_bilingual* [Accessed 29 January 2003]

Spicklemire, S., K. Friedly, J. Spicklemire, and K. Brand. 2001. Zope Web application development and content management. New Riders, Indianapolis, Ind.

Stickney, P.L., L.W. Swift Jr., and W. T. Swank. 1994. Annotated bibliography of publications on watershed management and ecological studies at Coweeta Hydrologic Laboratory, 1934-1994. U.S. Department of Agriculture Forest Service, Southeastern Forest Experiment Station, Asheville, N.C. GTR-SE-SO.


## AUTHOR


**** Correspondence to:*** John B. Jordin Jr., Information Technology Specialist, Southern Forestry Extension Office, University of Georgia Forestry 4-433, Athens, GA 30602

***E-mail:*** jbjordin@soforext.net

† Regional Extension Forester, Southern Forestry Extension, University of Georgia
‡ Scientist, U.S. Department of Agriculture Forest Service, Southern Research Station
§ Information Technology Specialist, Southern Forestry Extension, University of Georgia
‖ Scientist, U.S. Department of Agriculture Forest Service, Southern Research Station
# Information Technology Specialist, Southern Forestry Extension, University of Georgia