Roadmap for a Departmental Web Site

An academic department's Web site should provide extensive, dynamic information to serve faculty, staff, and students as part of an innovative Web presence

By **Guo-Qiang Zhang, Lee White, Christopher Hesse, Marc Buchner,** and **Mehran Mehregany**

Virtually every academic department in an institute of higher education requires Web presence as a critical component of its information technology strategy. The problem of how to leverage the World Wide Web and build effective and useful departmental Web sites seems to have long been solved. Yet browsing academic Web sites from around the world reveals a range from a mere collection of static HTML pages to more sophisticated enterprise-strength portals. If you pause to think about current practices and the issues an academic department faces, and then measure these against an "ideal" set of features a next-generation academic site should have, you will find an open-ended, complex problem rather than a "solved" one.

Here we provide a roadmap, based on our experiences, for developing a cost-effective and successful academic Web site. We believe that our experience and strategy can help an academic department build an innovative Web presence and provide critical IT support for its research and educational missions.

Challenges

The Web presence of an academic department presents a unique set of challenges. Chief among these are content management and budget constraints.

The content management issues facing an academic department—such as

the Electrical Engineering and Computer Science (EECS) department at Case Western Reserve University—are not trivial. The EECS department at Case has about 45 faculty, 15 staff, 200 graduate students, and 450 undergraduate students. A modern university department like ours provides a dynamic environment for students, faculty, and staff that should be reflected in—or better, facilitated by—its Web site. This translates to providing information that is both extensive in scope and dynamic in nature: seminars, courses, social activities, student and staff management, research publications, internal administrative documents, and even room-reservation and collaborative activities. All are examples of what a Web site can offer.

Consider, however, the traditional one-person "Webmaster" content-management paradigm. In the standard sense, a Webmaster is a person responsible for the design and maintenance of a Web site. As more information content shifts to the Web site, managing the site becomes increasingly intractable in this traditional centralized model. In this model, the Webmaster is the workflow bottleneck—the single point through which all updates are channeled. Not only is data entry tedious and time consuming, but the communication overhead is significant. Additional time and effort are needed to formulate and communicate requests in a precise yet efficient manner.

Budget constraints change the nature of the problem. Unlike a business or university's central administration, a typical department does not have sufficient resources to employ a team of IT professionals for its Web site development, let alone another team of full-time staff members for content creation and maintenance after deployment. This requires us to look for solutions other than the conventional, costly, business outsourcing model or ad hoc, HTML-Java–based, low-level approaches that might quickly bring a site up although making it difficult to maintain in the long run. Indeed, a key motivation for our Web development project arose from our earlier experience with the latter approach; our previous HTML-Java–based departmental site had been running for more than five years, but its content was difficult and labor-intensive to update. The site had deteriorated to the extent that it was largely ignored.

Building on our successful experience, the CSE is implementing a Web overhaul plan that reuses the EECS site by cloning it for the other seven departments within CSE

Meeting the Challenges

The EECS Web site (http://www.eecs.case.edu) was created two years ago as part of the Case School of Engineering's (CSE's) Web overhaul project. Recognizing that most academic Web sites are outmoded because of the centralized Webmaster paradigm, we looked for a cost-effective, long-term alternative. Our experience includes two key components to overcome these challenges: the application of content management systems[1] to ease the maintenance burden after deployment, and the identification of an open-source content-management development environment—Zope-Plone,[2] in our case—that provides both custom expandability and zero cost for the software.

Plone is a suite of powerful and flexible content management tools built on top of Zope, a Web applications development environment. Both Zope and Plone are implemented in the Python programming language, an interpreted, object-oriented language. The coupling of Zope and Plone provides a superior development environment for content management systems because of its open-source, object-oriented nature. It allows rapid software development through code reuse and extension of proven functional modules.

Content management systems offer a paradigm to streamline the content management process by

- separating form and functionality from content;
- providing workflow tools for maintenance and updates;
- associating to documents a set of built-in attributes differentiating among levels of content control, mode of publication, and versioning; and
- integrating seamlessly with database software.

Our content-management Web site, which has been running since fall 2003, successfully meets its design goals. Our site was also developed with reusability in mind. Building on our successful experience, the CSE is implementing a Web overhaul plan that reuses the EECS site by cloning it for the other seven departments within CSE. We also believe our approach is valuable and can be duplicated for academic departments at other institutions of higher education.

Requirements Specification

To understand the unique features a modern departmental Web site should have and why they are needed, it helps to summarize the shortcomings of existing sites and practices. One of the most common problems is a failure to keep information on a Web site up to date, caused in part by the traditional Webmaster-centric paradigm and in part by the lack of streamlined content responsibility assignment. Even with content responsibilities clearly delineated, the responsible party might not be facile with HTML syntax and source-level text editing, making the lack of a user-friendly content management interface yet another factor. Moreover, different implementation strategies and tools can have profound implications on how a Web site evolves in the long run, affecting aspects such as maintenance, usability, upgrades, scalability, and possible growth. The conventional notion of a Web site as a place to post news and information rather than as a gateway for information and interaction prevents the site from achieving its full potential.

We believe a modern departmental Web site should be an *information*

management hub fulfilling the management and developmental needs of the department in terms of research, education, and administration in a highly integrated way. This translates into the following set of desirable features:

- The Web site should be based on a content-management approach that achieves a separation of form and functionality from content.
- The content of the site should be partitioned into smaller fragments, with control delegated to responsible parties—the "generators" of the contents. These fragments are the basic units of information for assigning ownership and the building blocks of pages to be displayed on a user's screen. This eliminates the need for a Webmaster, replacing the bottleneck with a decentralized content-management paradigm.
- The owners of content "slices" should be provided with an interface seamlessly integrated into the Web site for information creation and update. The interface should be user friendly and require the user only to have baseline computer skills.
- A security infrastructure should control access so that assigned persons or groups have appropriate levels of access to specific areas of the site as specified by the design. A log-in process should be included to facilitate user authentication.
- The tension between form and functionality should be satisfactorily resolved as more weight shifts to the functionality and content control in the content-management system paradigm. Structured content entails more regularity and consistency in presentation, but it becomes harder to find the lowest common page-layout denominator to fit all the structured texts, graphics, and icons.
- The Web site should include a rich set of contents, features, and functionalities so that it becomes a place for "one-stop shopping" to the extent possible. At a minimum, the Web site should include faculty and staff profiles, seminar announcements, course offerings, the student information database, and other rudimentary information such as resources, newsletters, degree programs, and research areas.

The Web site should include a rich set of contents, features, and functionalities so that it becomes a place for "one-stop shopping" to the extent possible

Decentralization of content management (DCM) is the requirement from which several other requirements follow. The relatively autonomous governance of an academic department by its faculty and committees (such as the curriculum committee, recruitment committee, admissions committee, as well as student organizations) makes DCM a central pillar for departmental Web design and implementation. This contrasts with information management in a typical corporation, where the workflow aspect of content management is more important because management operates in a hierarchical rather than a decentralized manner.

Content Management Tools

Having committed to a DCM paradigm, the next step is to identify appropriate content-management tools for the Web development task. There are three primary approaches,[3] with the option of in-house or outsourcing development and customization for each category:

1. Creating a custom-built content-management application either from scratch or from some existing framework.
2. Purchasing a content-management application suite from a vendor and customizing it.
3. Using an open-source content-management development environment to build and customize the application.

Each category includes an array of choices as well. The unique situation of an academic department allowed us to quickly narrow our choice to the third option with in-house development rather than outsourcing. Among many possible open-source alternatives, we chose Plone as a clear winner because of its object-oriented approach.

Our choice had two important benefits:

- The customers are the developers. This dual representation helps reduce time and effort in design, specification, negotiation (or lack of it), testing, and modification.
- Adoption of the object-oriented Plone technology allows rapid development through code reuse and extension of proven functional modules.

Additional advantages of Plone include its acquisition mechanism, which allows objects placed inside other objects (such as folders) to reuse their parents' attributes, contents, and functionality. This mechanism makes it possible, for example, to have the layout of the whole Web site defined only once at the root folder, using a cascading

style sheet (CSS).[4] All the site content will be displayed according to this CSS definition by default unless a particular object overrides this property. Plone is also available in all major computer platforms: Linux, Sun Solaris, Mac OS X, and Windows.

Plone is not without drawbacks, however. Since it is written in Python, for example, site response can be slower than if it were written in another language; this is a common disadvantage of an interpreted language. Documentation has not kept up with software updates, sometimes forcing users to rely on message-board information to get hints on how to use new features. Professional-level application development in Plone requires programming experience and an understanding of the system, in part due to the rich set of features that come with it.

Based on our own experiences, the benefits of Plone's features far outweigh their shortcomings—which could well be dealt with in future versions. For example, we experienced faster responses after upgrading our site to Plone 2.0 in summer 2004.

Content-Framework Design

As mentioned, the DCM plan requires partitioning Web content into smaller slices so that individual control of the slices can be assigned to responsible parties. Each virtual slice represents an area to be formulated and implemented as a table or database, interconnected with other slices as needed. For example, the "Research Advisor" field in the graduate student database naturally cross-references to the faculty database; the "TA" and "Instructor" fields of the course database refer both to the graduate student database and the faculty and staff database. The records in each database can be further grouped according to specific values in some fields, with the finest partition of the database as individual records or entries with unique IDs.

Some of the content areas cannot be conceptualized as flat, relational databases; rather, they contain multilevel, "complex" data and are better understood from an object-oriented database viewpoint. For example, in the courses area, a yearly offering is a table; for each semester in an academic year, the offered courses are another table; for each course, there are further fields.

A working prototype can inspire users to generate in-depth and more valuable feedback on features and interface design for the next iteration

Currently implemented contents at the EECS site include the faculty and staff database, the seminar database, the graduate student database, and the courses database. Additional contents will be moved to the Web site as our effort continues. For example, an internal resource-reservation system (rooms, projectors) has been implemented and tested but not yet rolled out.

Site Implementation

The object-oriented paradigm coming with Plone and its underlying language, Python, naturally take us to the hybrid "rapid prototyping" and "iterative refinement" software development paradigm. This is in contrast with the traditional "waterfall" process where one phase has to be complete before the next phase begins. The hybrid approach suited our Web development effort particularly well because of the dual customer-developer representation in the same team. Moreover, a working prototype can inspire users to generate in-depth and more valuable feedback on features and interface design for the next iteration.

For both the Web server and the Web development effort we used Dictate, a Sun UltraSPARC IIe running Solaris 9 with a single 500 MHz processor, 1.5 gigabytes of RAM, and a 40 gigabyte hard disk drive. The Web site has been running since summer 2003, first in Plone 1.0 and then in Plone 2.0. The site has been actively used by all members and groups of the EECS department. We highlight some specific experiences with the Web site's deployment, maintenance, and reusability here.

Response Time

One issue with our initial deployment using Plone 1.0 was response time. Some of the most complex Web pages were noticeably slow to upload, sometimes taking up to six seconds. Our first approach to solving this problem was through the use of the caching mechanism in Plone 1.0 to retain information on frequently accessed Web pages. Although this improved the response time by a factor of two, we discovered that caching interfered with the normal log-in and log-out process, and we have since dropped this approach.

When a new Plone release became available in early 2004—one rewritten to address some of the response-time problems with Plone 1.0—we worked to redefine the Web site in Plone 2.0. One of the essential issues in Web design is with the skin of the design, which has to do with the "look and feel" of the displayed Web page. The use of Plone 2.0 substantially streamlined the skin of our layout, and we took this opportunity to incorporate the university template[5] in our upgrade to Plone 2.0. Although this was a major effort, the Web server response time showed a 40 percent improvement on the complex pages we tested, using the same software-hardware setup. In early 2005, we ported our Web site to an Apple Xserve server machine with 2 gigabytes of RAM, 80 gigabytes of hard drive, and dual 2 GHz G5 processors. The response issue has been completely resolved through this platform upgrade.

Seminar Scheduling

Of all the databases on our Web site, we have had the most experience with the seminar page. There have now been two faculty members in charge of departmental seminars, one per semester, and we have upgraded its implementation from ZCatalog, an earlier technology, to a more streamlined and customizable Archetype framework.

Testing

We learned a valuable—if painful—lesson about trusting data entered into the Web site, especially data related to access control: new data must always be checked and verified. As previously explained, our graduate student database allows faculty to view private fields such as GPA and qualifier standing pertaining to any graduate student, but not to modify these fields. To access this information, the faculty member must provide an ID and password, which will identify them as an EECS faculty member and give them this permission. Our faculty list had changed substantially, so we obtained a new list from the university registrar and entered it into the system. We were later embarrassed to discover that a misclassified graduate student was permitted to examine the records of other graduate students. The breach did not result from flaws in the security infrastructure but rather because of a mix-up in the list provided by the registrar, which we did not cross check. We now check and verify all changes to files of role groups.

Usability

The decentralized management of content has greatly alleviated centralized data-entry work. The menu-driven, Web-based data entry for individuals has worked well even for faculty, staff, and students lacking computer expertise. For certain content areas, users can choose from among several formats to capture their content: plain text, structured text, or HTML.

Reusability

We are currently cloning and customizing the EECS site to other departments in the engineering school. Following our roadmap, we expect content creation and organization to be the main task for the rest of the departments. We will save the other departments the developmental cost for the functionality framework, however, while at the same time disseminating the modern content management paradigm, which will simplify content management and organization and improve site quality in the long run.

The benefits of extending our Web development experience to other departments will become evident as their individual Web sites begin to serve their users successfully sooner and for less cost than we encountered

Conclusions

There is probably nothing in each component of our Web development that has not been discussed in the literature. We feel that our integrated solution includes the following benefits:

- Selection and packaging of the necessary components and their seamless integration so that the result exceeds the sum of the individual parts.
- Moving from research and discussion to developing a working Web site that makes a real-world impact. This entails a clear vision, careful design, good judgment in tool selection, and execution of the plan with disciplined software development management; for example, milestones and task decomposition.
- Custom-tailoring a content management system to an academic department setting and demonstrating its implementation roadmap, feasibility, and benefits.

We have much to say about reusability as well, but did not discuss the topic in depth here due to space limitations. We believe the benefits of extending our Web development experience to other departments will become evident as their individual Web sites begin to serve their users successfully sooner and for less cost than we encountered in developing our own site.

Acknowledgments

The Web development project was supported in part by Case Western Reserve University's Provost Opportunity Fund for Undergraduate Research. Our thanks go to James McGuffin-Cawley, Associate Dean of the Case School of Engineering, for such support, and to the past and present EECS Web-development student team members: Brian Beck, Steven Cerveny, Ian Charnas, Matthew David Crowley, Eric Friesen, Michael Lee, Michael Starke, and Rachel Ward. Our thanks also go to Jayne Hoon, then Marketing Director of the Case School of Engineering, for her continued interest, feedback, and support of this project. Many EECS faculty and staff members provided valuable feedback from their interactions with the Web site, including but not limited to Mike Branicky, Mark Doblekar, Steven Garverick, Simon Kuhn, Paul Schneider, Vincenzo Liberatore, Massood Tabib-Azar, and Tom Trelvik; our sincere thanks go to all of them.

Christopher Hesse acknowledges the support of a SOURCE (Support for Undergraduate Research and Creative Endeavors) grant, administered by Sheila Pedigo, Case's Director of Undergraduate Research, for this project.

Endnotes

1. G. Alonso et al., *Web Services, Concepts, Architectures, and Applications Series: Data-Centric Systems and Applications* (New York: Springer, 2004), <http://www.springeronline.com/sgw/cda/frontpage/ 0,11855,4-40109-22-2265535-0,00.html> (accessed April 18, 2005); and W. Powel and C. Gill, "Web Content Management Systems in Higher Education," *EDUCAUSE Quarterly*, Vol. 26 No. 2, 2003, pp. 43–50, <http://www.educause.edu/ir/library/pdf/eqm0325.pdf>.
2. See <http://plone.org> and <http://zope.org>.
3. Powel and Gill, op. cit.
4. E. Meyer, *Cascading Style Sheets: The Definitive Guide,* O'Reilly Media, 2000, <http://www.oreilly.com>.
5. See <http://www.cwru.edu/univrel/marcomm/creative> (accessed May 6, 2005).

Guo-Qiang Zhang (gq@case.edu) is Principal Investigator of the EECS Web Development Project, and an Associate Professor in the Department of Electrical Engineering and Computer Science at Case Western Reserve University in Cleveland, Ohio. Lee White is Professor of Computer Science, Christopher Hesse is an electrical engineering student, Marc Buchner is Associate Professor, and Mehran Mehregany is Goodrich Professor of Engineering Innovation and Chairman of the EECS Department at Case.