INCA: Balancing Power and Ease-of-Use in Courseware Authoring Support for Engineering Faculty.

Eckehard Doerry, Karim Nassar
Dept. of Computer Science, College of Engineering
Northern Arizona University
Flagstaff, AZ 86011

Abstract

The increasing reliance on course websites, whether to support distance education efforts or simply to streamline conventional teaching, has placed an additional burden on engineering faculties as they struggle to make their course materials web-accessible. Because existing commercial courseware authoring packages are based on rigid, generic templates and clunky editing and updating interfaces, faculty are frequently left with the daunting alternative of creating course websites from scratch. The INtegrated Courseware Authoring (INCA) system described here explores the middle ground between free-form, from-scratch website authoring and monolithic commercial systems like WebCT, by strategically integrating a commercial HTML site editor into a comprehensive package of courseware authoring tools custom-designed to meet the specialized needs of engineering faculty. The resulting INCA system is flexible, extensible, and motivates and supports acquisition of web-authoring skills by users.

1.0 Introduction

Over the last decade, the web has grown increasingly important in the delivery and management of engineering courses as distance education has become a hot topic in higher education (Tiffin, 1995; Davies, 1998) and, even in conventional on-campus offerings, students have come to expect the conveniences of a course website. Creating a modern, aesthetic website can, however, be extremely arduous, requiring a solid background in HTML, cascading style sheets (CSS), and graphic design principles. Building such a website from scratch can take days, or even weeks for instructors with undeveloped web authoring skills. Further augmenting a course website with advanced server-side features (e.g., an assignment submission tool) additionally requires sophisticated programming skills and in-depth knowledge of web technology. In general, the substantially higher effort (Doube, 2000) associated with creating and offering on-line courses has deterred many faculty from even web-augmenting conventional courses, much less moving entirely to a distance-delivered format.

Although there has been much discussion of how to effectively organize or develop web-accessible course materials in general (Talbott, 2002; Prupis, 1998), and a number of on-line learning tools for supporting acquisition of specialized skills have been introduced (Emory, 2002; Barra, 2000), there has been little examination of the challenges faculty face in *creating* effective course materials and websites, and consideration of how best to support faculty in overcoming

these challenges.

At an institutional level, the most popular response to reducing the enormous effort associated with creation of course websites has been the adoption of a commercial courseware authoring package - a monolithic, centralized, server-based system that provides a generic course template that faculty "fill in" with course contents for various courses. Examples of such systems include WebCT (http://www.webct.com/) and Blackboard (http://www.blackboard.com/). Although these systems do initially simplify the process of creating course websites, creativity and efficacy of the resulting sites is limited by the one-size-fits-all philosophy of these systems. The rigid course website templates provided by these systems make it difficult or impossible to restructure the default site, add or leave away certain features, or add unusual resources not provided for in the generic template. This constraint is particularly onerous for engineering faculty who frequently want to include unique elements like on-line simulations, applets, video clips, animations, and other "non-standard" materials in their course websites. Finally, the user interfaces (i.e., the mechanism for all authoring and updating interactions) of monolithic systems are painfully limited by their browser-based access model: faculty are expected to edit web pages, create quizzes, and perform other site maintenance using HTML forms within a web browser, rather than using specialized, efficient website editing software that runs on their desktop PC and takes full advantage of the interface capabilities (e.g. highlighting, drag and drop, etc.) of their desktop machine.

	Use commercial authoring system	Build your own website
Advantages	Fast and easy • full, sophisticated website in a day. • Minimal technical skills required	Flexible • tailored to specific needs of each course • support for specialized course content areas and media • faculty has sense of direct control
Disadvantages	Rigid! • constrained to pre-defined site content. • Site organization/layout is fixed. • Generic template includes many inappropriate pages/facilities for given course, i.e., dead wood. Clunky Interaction • Inefficient browser-based interface. • Arduous to update/add site pages; difficult maintenance.	Large time/effort investment • time to learn web concepts • time to create full site infrastructure • no energy to left for innovation.
Outcomes	Frustrated Faculty • acceptable for novices only. • no support or motivation for evolving skills. Weak websites • generic websites; inability to stray from template • many features not used; either inappropriate or too clunky to be worthwhile. • out of date due to hassle of updating pages.	Frustrated Faculty • acceptable for web experts only • frustrated with details Weak websites • ultra-rudimentary course web pages • poor style and organization • sophisticated features out of reach

Table 1: Trade-offs between commercial and do-it-yourself website creation.

Faced with these realities, faculty are left with two choices: resign themselves to living with the frustrating restrictions of the monolithic commercial systems, or learn enough web-authoring basics to create their own site from scratch. Table 1 summarizes the trade-offs and outcomes of each avenue.

As indicated in Table 1, building a course website from scratch offers high potential for custom features and tailoring of the site to the needs to a specific course; this is particularly valuable given the unique challenges of presenting engineering curricula on the web. Unfortunately, this potential is challenging to realize due to the effort involved in simply creating even the basic site infrastructure. Use of a monolithic system like WebCT or Blackboard, on the other hand, streamlines the process of putting up a generic site, but limits innovation, expansion, or other deviations from the default site structure.

The comparison drawn in Table 1 yields two key insights:

1. Although the challenges presented by the two approaches are quite different, both approaches lead to similar outcomes, namely frustrated faculty and mediocre course websites.
2. The disadvantages of the build-it-yourself may be relatively easy for engineering faculty to overcome *with proper support*. Engineers are accustomed to dealing with technical concepts; what is daunting about designing a website is the steep learning curve that must be negotiated to create anything significant.

What engineers need is a courseware authoring system that defines a middle ground between rigid, fill-in-the-template systems and from-scratch website authoring.

The Integrated Courseware Authoring (INCA) system meets this need by providing a coherent set of authoring modules that support rapid-start course website authoring for web novices, while promoting growth of web-authoring skills and encouraging creative extensions through its open-ended site authoring model. Specific design goals of the INCA system include:

- ✓ Rapid Start. Be able to put up a fully-functional website *even faster* than using a commercial system like WebCT.
- ✓ Flexible, Empowering. Provide a default course website, complete with core pages, but allow users to extend and modify the default structure freely. The system should function as a sort of "training wheels" for web authoring: the default site should be fully transparent to users so that they can learn from the default code and build their web authoring skills as they modify and extend it to meet their specialized needs.
- ✓ Efficient. The system should rely on client-side authoring interfaces wherever possible to streamline authoring and updating. Existing off-the-shelf software packages should be integrated where possible to avoid re-inventing the wheel.
- ✓ Ease of Use. The system should be simple to learn and use. All learning should situated within the context of creating an actual course website, i.e., "learn on-demand".

The remaining sections of this paper discuss the design and development of the INCA system in detail. We begin with a brief overview of the user-centered design process used to tailor our design precisely to the needs of engineering faculty. Section 3 then provides an in-depth description of the overall system and its key components.

2.0 Designing for engineers: a user-centered approach

The INCA project was motivated by the observation that, despite the availability of a wide variety of web authoring tools, engineering faculty are almost universally reluctant to develop web courses. Rather than simply designing yet another software package, we began our effort with an intensive, six-month user study to identify core web authoring challenges and obstacles encountered by faculty, and to understand why current authoring technologies are unsatisfactory. Specifically, the goal of our analytic effort was to address three central issues:

1. Obstacles. What are the main challenges that engineering faculty perceive in the creation of course websites and content? What aspects of previous authoring experiences have been the most frustrating?
2. Adequacy of existing systems. Are current courseware authoring alternatives, in fact, fundamentally inadequate? Or is it merely a matter of better training users?
3. Specialized needs. What types of course content do engineering faculty want to make available on the web, i.e., what authoring functionalities must an effective courseware authoring system support?

We began our analysis with an exploratory survey of engineering faculty that had offered or participated in the preparation of a web-delivered course at some time in the last five years. The survey served to (a) establish that faculty were, in fact, dissatisfied with current courseware authoring support and (b) identified a group of interested stakeholders that we could draw on. Our subsequent requirements analysis process proceeded along two parallel tracks.

First, following the principles of User-Centered Design (Greenbaum, 1991), we established monthly design meetings attended by our stakeholder group, which focused on educating stakeholders with surveys and demonstrations of what was possible with current web technology. Individual faculty were also interviewed separately to gain deeper insights into the obstacles and challenges encountered in course website authoring. This process yielded concise insights as to the perceived shortcomings of existing courseware authoring support, and a prioritized list of desired functionalities an ideal courseware authoring system should support.

Our second analytic thrust approached the design issue from the opposite direction, by evaluating a wide variety existing authoring tools, including not only comprehensive courseware packages like Blackboard and WebCT, but also Flash editors, HTML editors, audio/video editing systems, and systems for streaming live lecture over the web. Each of these systems was installed, used to create demonstration sites, and ultimately rated on a variety of metrics ranging from feature-richness to usability. The resulting matrices provided a comprehensive overview of the relative merits of various software packages[1]. The demo sites produced by this process were also fed back into our stakeholder meetings to stimulate discussion.

The outcomes of our analysis and requirements acquisition process are summarized in the following points:

- Monolithic systems like WebCT and Blackboard do a poor job of supporting faculty; they are plagued by rigidity and inefficient interfaces, and provide weak support for authoring and posting certain media types (e.g. video clips) that engineering faculty routinely include

[1] A more complete discussion of this comparative analysis and the resulting matrices can be found on the INCA website (http://denali.cse.nau.edu/INCA/)

in websites.

- Although commercial products are useful for some aspects of content creation (e.g. HTML editing), faculty were frustrated by having to wade through comprehensive, general-purpose tutorials and manuals to extract and learn the simple skills needed to create basic web content elements.
- Prioritized features of a courseware authoring system include easy site creation, support for efficient HTML authoring/editing, easy site updates, course management features (grades, uploading deliverables, etc.), and support for video and Flash editing.
- Speed and efficiency were emphasized at all levels. Faculty are frustrated by steep learning curves, but also by clunky, slow interfaces and arduous data entry processes.
- Faculty are willing, able and, indeed, eager to learn new technical skills, provided they have focused tutorial support, and the learning curve is moderate. Importantly, faculty should not be *forced* to learn advanced skills; rather, the system should support this learning over time, with direct and immediate benefits to reward learning effort.

These observations, in aggregate, form the fundamental motivating vision driving the design of the INCA system: to create a course content authoring environment that validates and supports the technical aptitude of engineering faculty, meets specific engineering course website needs, provides a streamlined authoring and editing model, and is able to evolve to complement the growing web authoring skills of faculty.

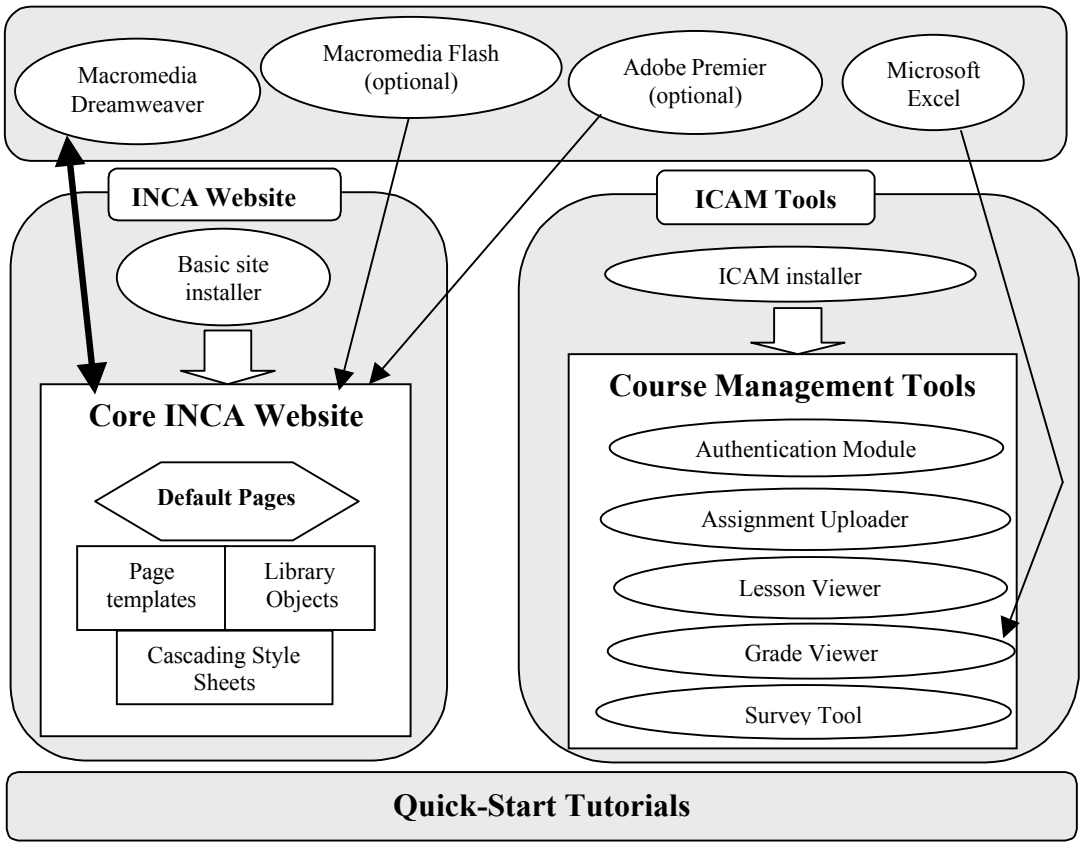

Figure 1: INCA system overview.

3.0 The INCA system

The overall design approach of the INCA system is to selectively integrate commercial authoring tools into a coherent courseware authoring framework that leverages the best features of the commercial products, while at the same time compensating for their shortcomings. Custom software components were then created to fill functional voids within the overall authoring environment. Figure 1 provides a graphical overview of the INCA system. As indicated in Figure 1, various commercial authoring products are used to efficiently create content for the INCA website, taking advantage of their efficient, specialized interfaces. This is particularly important for web page authoring and site organization (handled by Dreamweaver), a high-frequency activity which is poorly supported in commercial systems like WebCT. The following paragraphs describe the two central components of the INCA system – the Core Website Installer and the INCA Course Administration Module (ICAM) - in more detail.

3.1 Core INCA Website Installer

Our user analysis showed that many faculty use (or have attempted to use) HTML editors like Macromedia Dreamweaver, Adobe GoLive, or Microsoft Frontpage to maintain rudimentary websites. The central complaints with these systems were (a) the frustration and steep learning curve involved in sifting through the plethora of sophisticated features available in these professional grade packages to identify and learn the simple subset needed to get even started and (b) the enormous learning investment required to acquire the advanced web authoring skills it takes to create a truly sophisticated and aesthetic website.

The INCA approach is based on the observation that both complaints reflect *initial start-up costs*: if faculty were *provided* with a sophisticated core website, they could easily add content and *maintain* the site. Faculty could also, if they *choose*, extend or modify the core site as skills and the needs of their particular courses dictate.

Figure 2 shows an actual course website developed on the INCA core site framework. The core site framework we developed is simple and efficient, complete with "tabbed" topic areas (course info, assignments, help, etc.) and "dummy" content pages for default site components (e.g. syllabus, schedule, etc.). Creating an INCA website is efficient and easy, requiring only three simple steps:

1. Download and and unzip the site template package; this creates a complete Dreamweaver[2] website on the instructor's desktop machine. Launch Dreamweaver and define a new site based on the installed files.
2. Modify the dummy default documents (e.g. syllabus, assignments) to install actual content for the course. Additional pages (e.g. more assignments) can easily be created by using page templates[3], or by writing them from scratch. The site layout and organization can

[2] The decision to rely on Dreamweaver as an HTML editor and site manager was purely practical. We wanted to provide a maximally sophisticated core website, that was nonetheless easy for novices to edit and extend; Dreamweaver's templating and style management features allow us to do so.

[3] The advantage of using page templates is that they predefine the structure of the pages, as well as the layout, fonts, and other styles (i.e. a CSS) that apply to the pages. This makes it easy to extend the core site while maintaining the core look and feel of the site.

also be extended/modify as desired.

3. Upload the complete site to the university web server. This involves a one-time setup of remote server information in Dreamweaver; all subsequent uploads and updates can then be accomplished effortlessly at the push of a button.

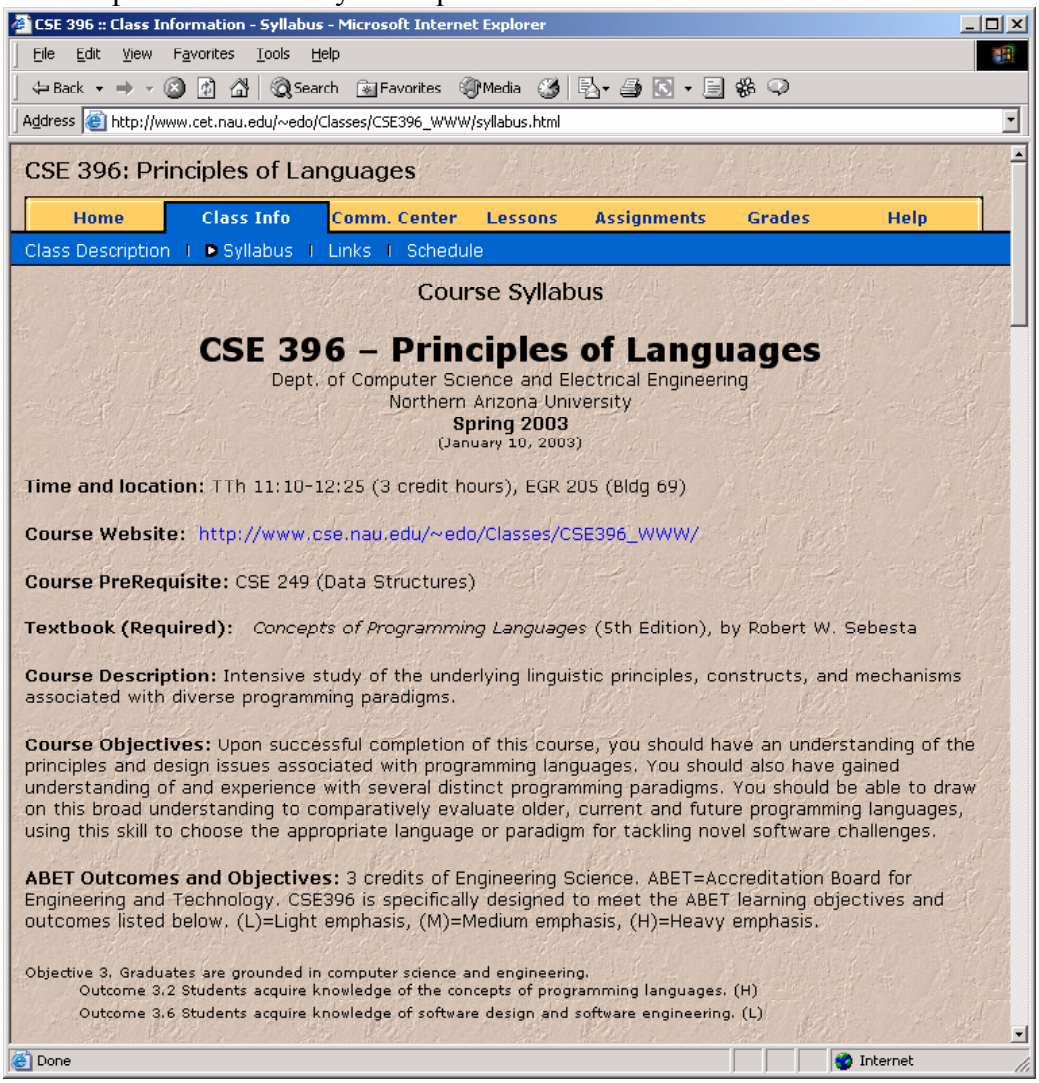

Figure 2: The syllabus page of an INCA website.

A complete website can be created and populated with content in about two hours by a faculty member with moderate experience in using commercial HTML editors. This streamlined approach satisfies our user's central desire to get a website up as fast as possible. The fact that the instructor has complete control over site structure and contents addresses the rigidity and efficiency complaints noted for existing commercial course authoring systems like WebCT. The basic default website can be extended as time, course needs, and growing web editing skills permit. Because the structure and page contents are completely flexible, the instructor may also create (using the other commercial authoring tools like Adobe Premier and Macromedia Flash) and install arbitrary content on the site; there are no limits to what is possible.

3.2 INCA Course Administration Module (ICAM)

The simple INCA website install process, combined with the use of a powerful production HTML editor (Dreamweaver), makes it fast and easy for instructors without extensive web programming skills to create and maintain an elegant and aesthetic course website. But this addresses only one portion of the website support needs expressed by faculty in our user studies. Although a basic course website provides a powerful way to *disseminate* course materials, it lacks support for *managing* everyday course-related activities, including gathering student feedback, collecting assignments, and distributing grades.

The goal of the INCA Course Administration Module (ICAM) is to reduce faculty workload and increase student satisfaction by automating the most common and time-consuming course management functions and making them web-accessible. Based on our initial user study, electronic assignment submission was given highest priority, followed by grade dissemination, and collection of feedback/voting from students; lower priority functions included on-line quizzes and facilities for distributed communication (chat, class newsgroup, etc.).

Although the underlying ICAM architecture is quite complex (see Section 3.2.5), a simple and elegant web interface insulates instructors from all system level considerations. Security is seamlessly managed by the system as well: all users are authenticated against the central database of student and faculty accounts, and course enrollments maintained by campus computing services. Only the official instructor may administer the ICAM components for a given course, and only students currently enrolled in a course may access its ICAM functions. Authentication also makes it possible to reliably identify accessing students to control viewing of grades, tag uploaded documents with the identity of the submitter.

An important feature of the ICAM module is that access to ICAM functionalities is completely unrelated to the core INCA website described earlier. Because all ICAM components are ultimately accessed via a WWW address (URL), they can easily be added to *any* course website, whether it was created using the INCA website template or not. Thus, instructors with mature web-programming skills and pre-existing websites can easily add ICAM course management components.

The following sections provide detailed descriptions of the four main ICAM components: the assignment submitter, the grade viewer, the survey tool, and the lesson viewer.

3.2.1 ICAM Assignment Submitter

In both conventional and distance education contexts, faculty have become increasingly interested in electronic submission of coursework, including programming assignments, written reports, CAD/CAM files, and other electronic media. The advantages, in principle, are attractive: 24-hour access, unambiguous submission timestamps, paper savings, and so on. Unfortunately, the logistic effort of sorting through email attachments, enforcing submission deadlines, and simply managing the sheer volume of submissions often makes it easier for faculty to continue relying on hardcopy submissions.

The ICAM assignment submission component allows an instructor to quickly create “assignment

uploaders" for each assignment. After naming a new uploader, it is configured to specify the due date, whether late assignments should be accepted and, if so, how they should be identified in the upload log (Figure 3). The assignment uploaders exist on the central webserver and are accessed via a URL provided to the faculty member upon creation of the uploader. The instructor simply pastes this URL into an appropriate link (e.g. "Click here to upload your answer to Homework 1") on the course website; students clicking on the link are authenticated, then taken to a page that allows them to upload their solution file. An upload log showing student name, files uploaded, and timestamps allows the professor to monitor submissions as they trickle in. To retrieve the uploads, the professor simply clicks a button and is presented with a zip archive containing all submissions.

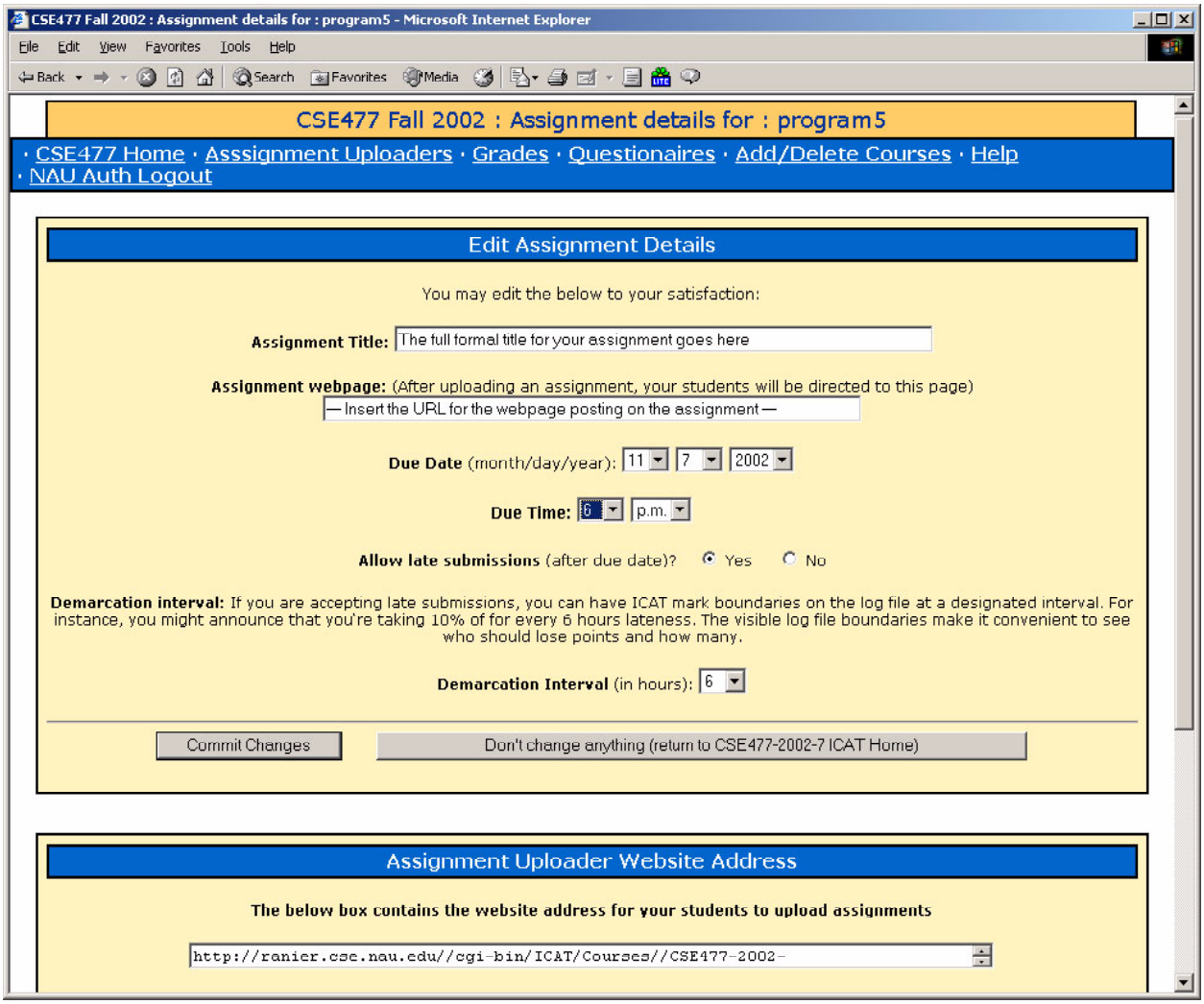

Figure 3: Creating a new ICAM assignment uploader. Note the URL assigned to the new uploader that students will use to access the uploader shown near the bottom.

3.2.2 ICAM Grade Viewer

Even in conventional teaching scenarios, keeping students apprised of their grades is irksome, usually involving constant inquiries during office hours. A secure mechanism for online grade distribution would allow students to check their grades anytime; both instructor and student could

benefit. Unfortunately, attempts to provide such a facility in typical commercial courseware systems (e.g. WebCT) require grades for each assignment to be individually entered for each student using a clunky web interface, an effort that completely outstrips any potential advantage.

Grade dissemination in ICAM is based on the observation that most faculty use some sort of spreadsheet software to maintain their course grades. Accordingly, we developed an efficient mechanism for importing spreadsheet files directly into ICAM. Instructors maintain their grades as usual, using their favorite spreadsheet software. At convenient intervals, they export the grade file in tab-separated format, and upload it to the course website via the ICAM system. The entire export-update process takes less than two minutes. Instructors are required to add a column containing each student's "user id" to their spreadsheets, which is used by ICAM to display (only) the appropriate row of scores to a given student. In addition, rows that should be shown to all students (e.g. labels, points possible, etc.) and columns that should remain hidden (e.g. private comments) can be easily marked in a point-and-click fashion.

3.2.3 ICAM Survey Tool

In conventional classroom contexts, faculty are able to access student feedback continually, ranging from the peripheral "hallway gossip" to an explicit show of hands. This dynamic is compromised in distance education scenarios. Although email surveys are possible, they are arduous to manage for large classes and do not provide for anonymous feedback.

The ICAM survey tool allows an instructor to quickly create a new "survey", consisting of a sequence of questions, and make it accessible to students via a link from the course website. After being authenticated, students may submit free-form answers to each question. Answers from individual students are uniquely identified (e.g. "respondent1", "respondent2") to allow tracking of an individual's responses across various questions. Although the instructor can view a list of respondents (to allow, for example, assigning participation credit) there is no way to correlate respondents with the submitted responses, thus guaranteeing anonymity.

3.2.4 ICAM Lesson Viewer

On-line course notes (or "lessons") are becoming increasingly popular as many faculty convert their course notes to electronic format.

Posting such notes to the web is simple in principle – one merely converts them to HTML format and uploads the resulting files to the course website. This approach, however, destroys the coherence of a lesson. What was once a related set of pages organized by topic and perused sequentially is now just a collection of files in a directory. Although one can create an "index page" that provides an outline of the lesson with hotlinks to the individual lesson pages, creating index pages is arduous, particularly if lessons are frequently updated or re-organized.

The ICAM lesson viewer solves this problem by dynamically traversing a folder hierarchy (i.e., a directory tree) of lesson files and *automatically* constructing a sophisticated table of contents for the lesson. Thus, the instructor merely organizes HTML lesson files in a folder hierarchy that reflects the proper topic-subtopic structure, names the various files and folders appropriately[4], and the lesson viewer does the rest.

[4] The folder and lesson file names provide the topics/subtopics that appear in the table of contents constructed by

When activated and passed the root folder of a lesson hierarchy, the lesson viewer opens displaying the first page of the first lesson, as well as an expandable/collapsible table of contents pane in the top left corner (Figure 4).

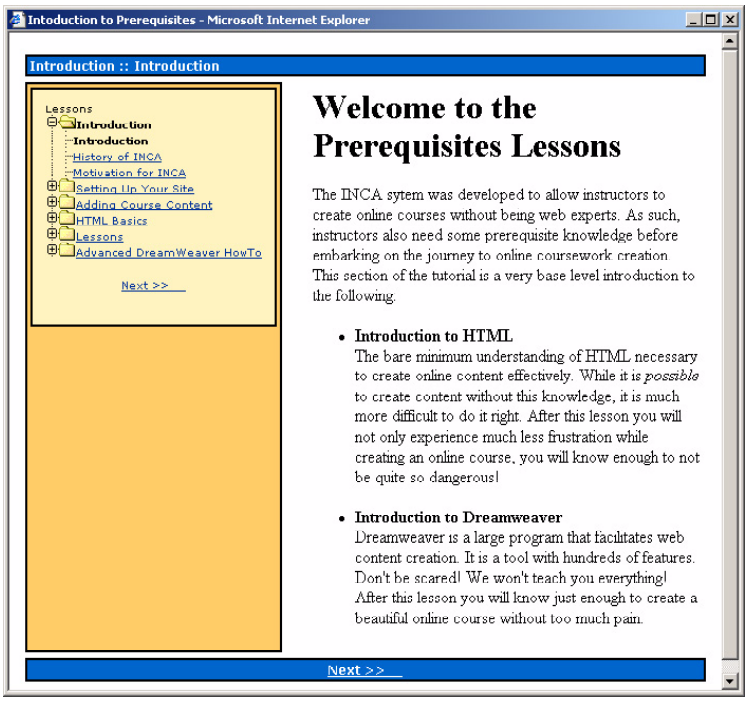

Figure 4: ICAM Lesson Viewer. The active table of contents in the top left corner presents clickable, expandable hierarchy of lesson pages.

In this way, the reader is continually oriented within the lesson, can preview upcoming topics, and can jump around within the lesson at will, by clicking on the heading shown in the table of contents. Importantly, there is no cost associated with providing and maintaining the table of content for the instructor. Lesson files can be edited, re-arranged, added or deleted at will; the table of contents created by the lesson viewer reflects these changes automatically.

3.2.5 ICAM architecture overview

The functional core of the ICAM module is a set of Perl scripts that dynamically generate the ICAM web interface pages (e.g. Figs. 3 & 4) based on a specific information for each course stored by the ICAM system. The scripts can be divided into two categories: *management scripts* are used by the instructor to create and configure ICAM elements; *access* scripts generate the web pages that students use to access ICAM information (e.g., to submit assignments, view grades, etc.). Security is enforced by combining the standard "htaccess" mechanism provided in the Apache webserver with a proprietary Perl module that accesses the central faculty, student and course enrollment databases maintained by campus IT services[5]. This makes it possible to selectively limit access to various ICAM resources to only students and instructors officially

lesson viewer.

[5] To install the ICAM module outside of the NAU environment would require providing a replacement for this custom Perl module that somehow supports the same functions, i.e., returns information on instructors, students and course enrollments.

associated with a given course, to distinguish between students and instructor, and to return the specific login ID of users. This, for example, ICAM can limit access to management scripts for a course to the instructor assigned to the course; both students and instructor may use the access scripts to view ICAM information.

As indicated in Figure 5, all courses ICAM components (uploaders, gradeviewers, surveys) associated with each course are accorded a separate directory on the ICAM server in which to store their data. Because security is maintained at a directory level (the htaccess model), the ICAM scripts must exist within the directory associated with each course. To simplify code updates, however, it is highly desirable to make all ICAM courses access the same centrally maintained set of PERL scripts. We solve this problem by maintaining a centralized set of scripts, which are accessed by "hard links" from within each course directory. The behavior of the scripts is customized for each course via a "course configuration file", which resides in the course directory and contains configuration information, file system paths and security data for the specific course.

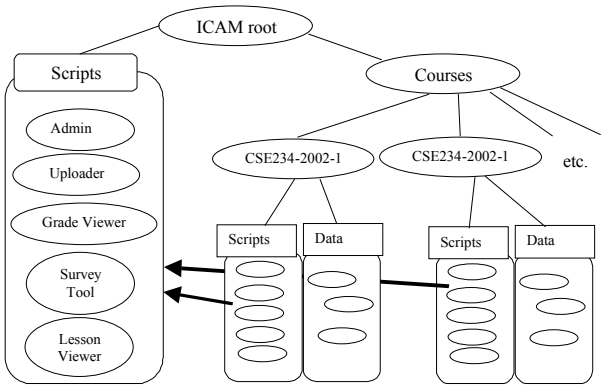

Figure 5: Overview of ICAM architecture. Each course is maintained on the ICAM server; instructor and student access is limited to web interfaces only, ensuring integrity and security.

The various data directories associated with each course serve as "databases" for the ICAM modules, storing information on each ICAM component installed for a course. In this way, a course can have, for instance, multiple assignment uploaders associated with it, each with its own data files. Each course is also assigned a unique identifier based on the semester and year of the course offering, making it possible to have multiple instances of a repetitively taught course.

Creating a new course is simple under the architecture outlined above: after querying the instructor for necessary course information, the ICAM course installer creates the appropriate course directories, creates the (ht)access control files, creates links to the core ICAM scripts in the new directory, and creates a course configuration file.

Although the ICAM system involves sophisticated server-side functionality, installation of the ICAM module on the central (Unix) web server is simple: the system administrator simply edits a configuration file and runs an installation script. Once installed, all faculty with user accounts on the central web server are able to access ICAM functionality. The only "specialized" aspect of the ICAM scripts is the custom authorization component noted earlier, that queries the centralized course and registration database maintained by campus IT services; no other non-standard PERL modules are used. Thus, the ICAM system is easily portable, requiring only minor modifications

to reference some locally-appropriate security management function (which could, at its most basic, simply look up login ID's in flat text file).

3.3 Quickstart Tutorials

Although the INCA system is designed for simplicity and ease-of-use at every level, it is impossible to insulate users from *all* of the technical skills required for web authoring. For example, INCA users must have some basic knowledge of HTML, and master certain basic features of Dreamweaver in order to successfully create and edit web pages. Similarly, the ICAM interface, despite all efforts to make it as simple as possible, may still require some explanation for first-time users. As discussed in Section 2, our initial user studies indicated that a basic level of technical sophistication and an ability to learn technical concepts is not unreasonable to expect from engineering faculty. Indeed, most faculty were eager to learn basic technical skills, *provided that focused, task-specific support was provided.*

The INCA system provides a range of "Quickstart tutorials" on topics including basic HTML authoring, creation of new site pages based on the core INCA templates, use of ICAM components, creating and importing video clips and other media, and many other topics related to key authoring tasks identified in our user study. Each of these tutorials is structured as an INCA lesson and made accessible via the ICAM lesson viewer. The tutorials purposefully maintain a tight focus, providing specific step-by-step instructions for accomplishing core website editing tasks, rather than sweeping discussions of a technology in general. In this way, learning effort is directly related to immediate concerns and outcomes; faculty have a sense of immediate progress towards getting up their basic course websites.

4.0 Conclusion

In every design context, there are multiple stakeholders, each with differing conceptions of what is most important in the designed artifact. Successful software design hinges on ensuring that the users that have to invest the most effort in using the software are also those that benefit most directly from doing so (Grudin, 1988). Our analysis of the web authoring experiences of engineering faculty suggests that monolithic courseware systems like WebCT fail to recognize that *faculty* who the primary stakeholders in a courseware authoring system. Our findings indicate that dissatisfaction stemming from the overall rigidity, inefficiency, and poor authoring interfaces of the monolithic systems typically provided by universities was the major reason why engineering faculty are reluctant to take on web courses. Engineering faculty require a new approach to courseware authoring, one that minimizes authoring effort, supports evolution of instructor's web authoring skills, and is tailored to the precise needs of engineering faculty.

Specifically, our analysis revealed that engineering faculty require more flexibility and higher efficiency than is possible in authoring packages like WebCT, and are quite able to cope with somewhat more technical responsibility in exchange.

The INCA system directly reflects these specialized needs by providing a comprehensive, powerful, and fully integrated courseware authoring package that allows faculty to:

- easily create sophisticated course websites in a matter of hours

- edit and maintain those sites, including easy integration and management of "non-standard" media, and free-form restructuring and extension of the core site to accommodate the unique needs of individual engineering courses.
- securely and efficiency incorporate sophisticated course management functions like electronic assignment submission, grade dissemination, and collection of student feedback.

From a more general perspective of design philosophy, the INCA system demonstrates the power of compromise, splitting the difference between monolithic, centralized systems and completely unsupported free-form web authoring. By exploring a middle ground between these two extremes, the INCA approach is able to maintain much of the flexibility and adaptability of free-form authoring, while at the same time requiring only minimal technical skills of the instructor. The result is a system geared towards engineering education, a context characterized by a user community with strong basic technical aptitude but widely varying web programming skills, and a unique set teaching approaches and materials that do not fit easily into the rigid website schemas provided by commercial systems.

After several semesters of beta-trials, the INCA system is now in full release within the engineering college at Northern Arizona University. We expect that the system will continue its evolution as faculty adopt the system and provide feedback on their experiences. Whatever changes this brings, we are confident that our middle-ground design philosophy, which empowers users rather than restricting them, and which explicitly focuses on minimizing authoring effort, represents the key to creating effective authoring support for engineering educators.

Bibliography

Barra, M., G. Cattaneo, U. Ferraro Petrillo, V. Garofalo, C. Rossi and V. Scarano (2000). Teach++: a cooperative distance learning and teaching environment. *Proceedings of the 2000 ACM symposium on Applied computing*. Como, Italy. ACM Press

Davies, G. (1998). Distance education and the Open University. ACM SIGCSE Bulletin. Volume 30 Issue 4.

Doube, W. (2000). Distance teaching workloads. *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*. Volume 32 Issue 1.

Emory, D. and R. Tamassia (2002). JERPA: a distance-learning environment for introductory Java programming courses. *Proceedings of the 33rd SIGCSE technical symposium on Computer science education,* Volume 34 Issue 1.

Grudin, J. (1988). Why CSCW applications fail: problems in the design and evaluation of organization of organizational interfaces. *Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, January 1988.

Kearsley, G. (1982) Authoring systems in computer based education. Communications of the ACM. Volume 25 Issue 7.

Kurtz, B., D. Parks and E. Nicholson (2002). Effective internet education: a progress report. *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*. Volume 34, Issue 1.

Greenbaum, J. and M. Kyng (1991). Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates. Hillsdale, New Jersey.

Prupis, S. (1998). Introduction to creating online courses. *Proceedings of the 26th annual ACM SIGUCCS conference on User services*, October 1998.

Tiffin, J. and Rajasingham, L. (1995). In search of the virtual class, Routledge Publishers, London.

Acknowledgements.

We would like to gratefully acknowledge the generous support of the NSF (Grant #EEC-9725036), which provided the funding for the initial phases of this work. The NAU College of Engineering and, in particular, the Dept. of Computer Science also provided critical funds to bring the project out of the laboratory and to full release. Bill Busby and Daniel Wallace contributed much work to the user studies and early implementation, respectively.