Web-Based Courseware Application Usability

Thomas Convery
Dept. of Electrical Engineering
and Computer Science
Vanderbilt University
Nashville, TN 37235
t.convery@vanderbilt.edu

Brandon Nuttall
Vanderbilt University
VU Station B, Box 354035
Nashville, TN 37235
b.nuttall@vanderbilt.edu

Bobby Bodenheimer
Dept. of Electrical Engineering
and Computer Science
Vanderbilt University
Nashville, TN 37235
bobbyb@vuse.vanderbilt.edu

ABSTRACT

Courseware packages are software applications that facilitate distribution of information from professors to students, as well as communication among students in a class. Most commercial off- the-shelf courseware solutions are web-based and do not require client-side installation of additional software packages. While the web-based interfaces provide nearly ubiquitous access, they have disadvantages. First, web browsers were never intended to be application platforms. Second, a web-based interface requires that all data processing short of the final rendering of the information be completed on the web server that supports the application. In the case of a courseware application, the limitations of the web- based interface usually require that the information accessible through the interface be divided into unnatural partitions, making the application difficult to use to its fullest benefit. This paper describes a new courseware interface without the problems of the web-based paradigm. Our solution makes use of the XML User Interface Language (XUL), which allows us to harness the power of the client machine for tasks such as filtering and sorting of displayed data, allowing a radical reorganization of the presentation of data in the courseware application. We describe our courseware solution, Theo, and present results from a comparative user study.

Keywords

Courseware, XUL, User Interface Design

1. INTRODUCTION

As high schools and colleges have invested money into their information infrastructure, the software that students, educators, and administrators use to interact with their technology has evolved. One of the focuses for software development in the schools has been toward an online classroom interface, which we refer to as courseware. For the purposes of this paper, courseware is defined as software that:

- Students use to monitor classroom assignments, retrieve classroom materials, and communicate with one another and their educators, and
- Educators use to create and manage classes, post classroom materials, and communicate with students.

Prometheus [1], originally developed at George Washington University, is an HTML-based courseware system authored in Macromedia ColdFusion, an application for serving dynamic HTML applications. It is described by its owners, Blackboard Inc., as an "easy-to-use, flexible, [and] scalable. . ." system. Casual surveys of students and educators who use Prometheus, however, have suggested that there is much room for improvement concerning Prometheus' usability, which prompted our investigation into alternate interface designs.

Based on heuristic evaluations and feedback from users at Vanderbilt University, we identified the main problems with Prometheus from a user's perspective as:

1. Ambiguous partitioning of content.
2. Latency and interface responsiveness.
3. No enforcement of content partitioning.

This paper presents a prototype for an alternate courseware interface that addresses these problems, as well as the results of usability testing on the prototype interface.

2. BACKGROUND

Prometheus is certainly not the only courseware solution on the market; other solutions include Blackboard Learning System and WebCT (owned by WebCT, Inc.). For a comparative survey of several, see [2]. Most of these products share what we see as faults with Prometheus, namely that they are HTML-based and have mostly the same content partitioning scheme as Prometheus does. This study does not attempt to address issues with fat clients such as maintenance or compatibility. Nor does it address thin clients such as those using Java applets, e.g., [4]. Rather it focuses on content partitioning, and the limitations that a typical web server places on such partitioning.

Previous papers have addressed some of the problems we intend to solve with our alternate courseware interface: Nguyen et al. [5] acknowledged that some problems with courseware can be traced to problems with the internet, but then dismissed those issues as largely technical in nature. Calivi and De Bra [6] propose a solution to the partitioning problem, but as we will see the client's computer power can be leveraged to a far greater effect.

There has also been considerable research done on classroom software using the ubiquitous computing paradigm [7]. Our focus is more narrow, and is on one specific method of web-based classroom management. Alternative and often more ambitious designs for integrated learning environments have been proposed, e.g., the

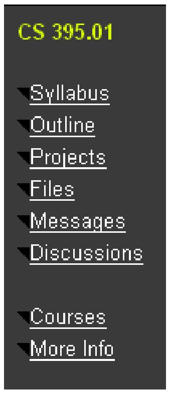


Figure 2: An example of how Prometheus partitions by type.

CoWeb project of Guzdial [3]. Such environments often take an unstructured approach to content partitioning.

3. COURSEWARE DESIGN

To demonstrate and test an improved courseware interface, we developed Theo, a XUL-based [8] interface that displays the same information as is available in Prometheus, albeit greatly rearranged. XUL is the XML User Interface Language, an offshoot of the Mozilla web browser project. XUL was developed as a solution to the problem of maintaining different front-ends for the various platforms on which Mozilla runs. Rather than maintain a separate front-end for each platform, Mozilla interface designers specify a single set of XUL interface descriptions for the application. Underlying platform-specific code is responsible for rendering the interface.

A benefit of Mozilla's XUL-based interface is that the Mozilla (and by extension, Netscape 6.x and 7.x) browsers can load a XUL file from any location, e.g., an HTTP uniform resource identifier (URI), and display the interface in the area usually reserved for a "web page." The XUL back end integrated into the browser will ensure that the resulting interface is identical regardless of the platform on which the browser is running. Because of the development of XUL, Mozilla provides an ideal cross-platform environment for developing a user interface that is universally accessible.

The Prometheus interface typically partitions information by class, shown in Figure 1, or by type, e.g., "assignments," "discussions," shown in Figure 2. In contrast, the Theo interface allows the user to view as much or as little information as he desires. Each individually posted piece of information appears in Theo as a node in the "contents tree." Each node can be flagged with one of four types: assignment, class meeting, discussion, or file. The node type allows the user to show or hide information based on its type; multiple types of nodes can be shown at the same time. These types were chosen to most closely mirror the divisions of information in Prometheus; the addition of additional types of information is trivial.

Additionally, each node, regardless of type, can have child nodes of the same or different types. A key problem observed in Prometheus is the partitioning of information that naturally belongs together. For example, an assignment is in a different partition from discussions. If a user wishes to find discussions related to a posted assignment, he must leave the assignments partition to access the discussions partition, where he must manually search for discussions related to the assignment in question. One would likely want to have discussions about a particular assignment easily accessible from the page that describes the assignment. Theo allows "discussion" nodes to be children of an "assignment" parent, making discussions about an assignment easily to locate, shown in Figure 3.

The back end for the Theo application is a set of Resource Description Framework (RDF) [9] files, each of which describes the content available for each class. XUL has built-in methods for parsing RDF files and displaying them as part of an interface, e.g., as an expandable tree.

The Theo interface is a three-pane interface, reminiscent of many popular e-mail readers. The left-hand pane allows the user to select or deselect what categories of information he wishes to see or hide, based on the information's category and class. Below the selection areas, buttons allow the user to call up static information about the classes in which he is enrolled (namely, a syllabus for the class and a list of classmates and their e-mail addresses). The right-hand pane is divided into two sub-panes. The upper pane shows the "contents" available to the user, based on the current filtering preferences selected in the left-hand pane. The bottom pane is a miniature web browser that shows the content the user has called up, either by selecting a node from the contents pane, or by selecting one of the buttons in the left pane. In both cases, the node selection or the button press causes the application to load a URI into the browser pane. The URI is stored in the RDF database that forms the back end to the application.

As mentioned, the user interface of Theo is described by XUL, which is tied to the Mozilla render core. However, the internals of Theo, particularly the part describing the data sources used by the courseware interface, are described in RDF, a W3C standard. Thus Theo can be ported to other platforms by re-implementing only the interface.

4. EXPERIMENTAL DESIGN

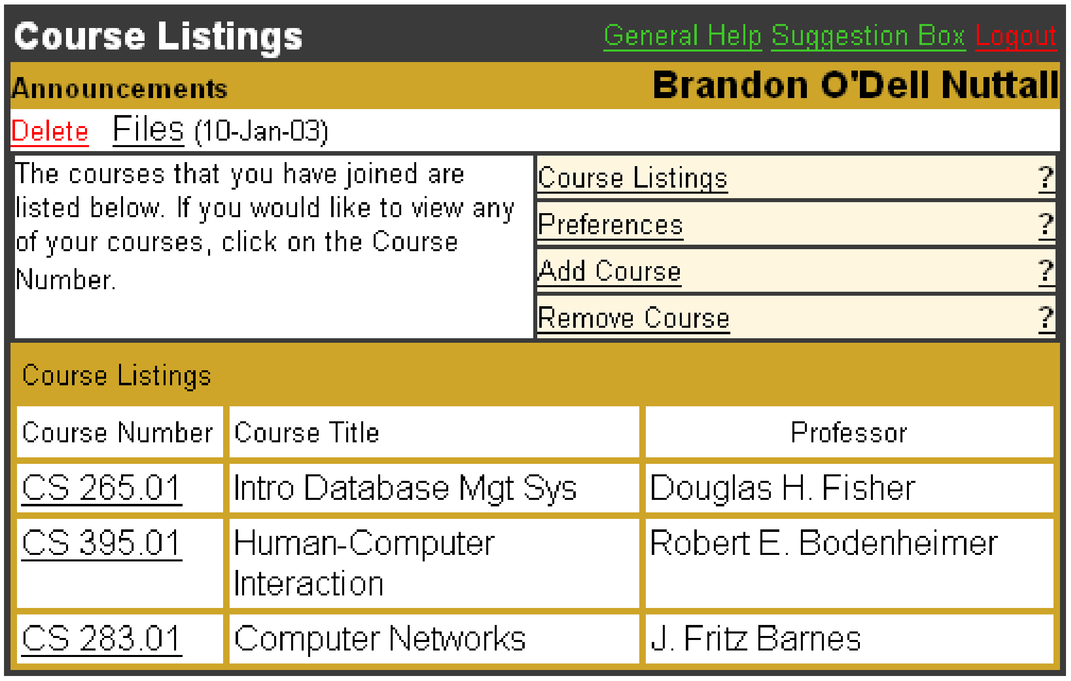
To test the utility of the Theo interface, we conducted a study on twelve undergraduate and graduate students of the Vanderbilt University School of Engineering. All subjects had at least some prior experience with Prometheus; however, our focal hypothesis was that Theo would outperform Prometheus. Therefore, any bias that prior use of Prometheus would bring would work against our hypothesis, not for it. The information available in Prometheus was duplicated into Theo for the purpose of the study. Each subject was given a short tutorial on Theo prior to evaluating the interface. Each subject was asked to complete three timed tasks on both Prometheus and Theo. The three tasks performed by the subjects were:

1. From among all of your classes, find the next three assignments due after September 10, 2002.
2. Consult the syllabus for CS 265 to find the ISBN for the textbook.
3. Assume that your name is John Smith. See if someone has posted a reply to your question about Exam 2 in CS 265.

One-half of the subjects completed the tasks first on Prometheus, then on Theo; the other half used Theo first, then Prometheus. Following the timed tests, the subjects were asked to complete a survey consisting of short-answer questions and five-point Likert scales to gauge their response to the new interface.

5. RESULTS

This section presents the results of the user study by task, and summarizes the Likert scale responses.



Course Listings

General Help Suggestion Box Logout

Announcements **Brandon O'Dell Nuttall**

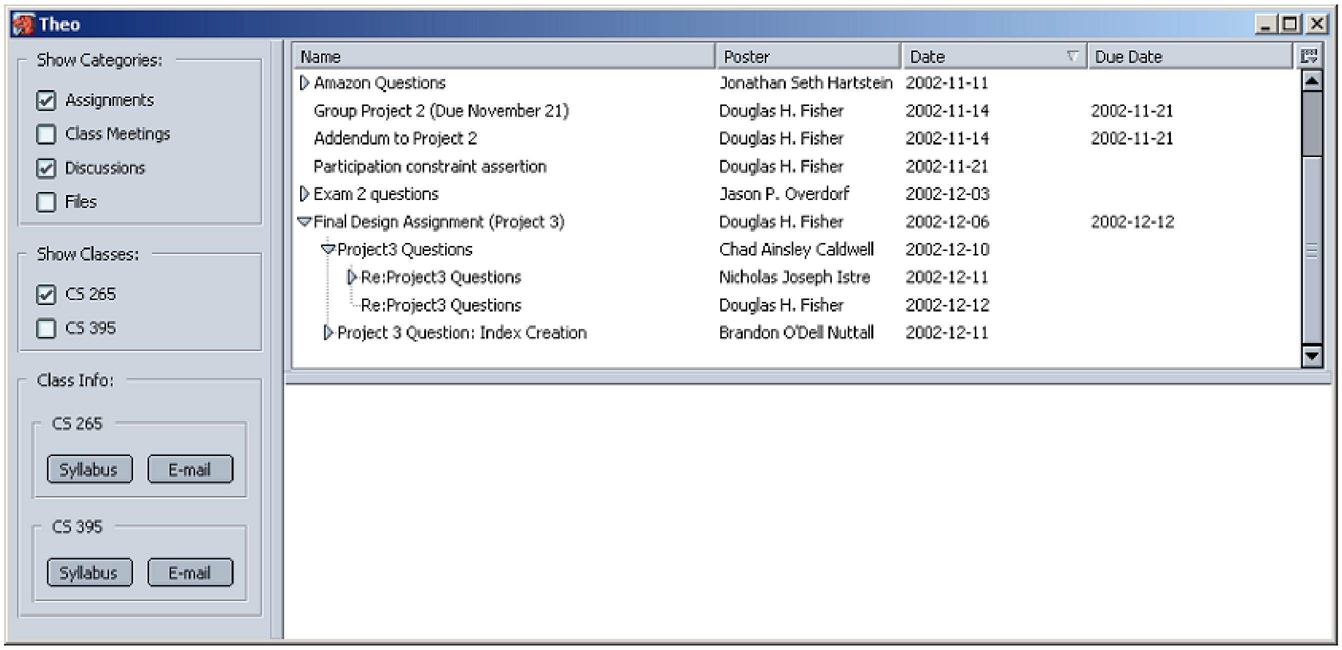
Delete Files (10-Jan-03)

The courses that you have joined are listed below. If you would like to view any of your courses, click on the Course Number.

Course Listings

Course Number	Course Title	Professor
CS 265.01	Intro Database Mgt Sys	Douglas H. Fisher
CS 395.01	Human-Computer Interaction	Robert E. Bodenheimer
CS 283.01	Computer Networks	J. Fritz Barnes

Figure 1: An example of how Prometheus partitions by class.



Show Categories: Assignments ✓, Class Meetings, Discussions ✓, Files

Show Classes: CS 265 ✓, CS 395

Name	Poster	Date	Due Date
Amazon Questions	Jonathan Seth Hartstein	2002-11-11	
Group Project 2 (Due November 21)	Douglas H. Fisher	2002-11-14	2002-11-21
Addendum to Project 2	Douglas H. Fisher	2002-11-14	2002-11-21
Participation constraint assertion	Douglas H. Fisher	2002-11-21	
Exam 2 questions	Jason P. Overdorf	2002-12-03	
Final Design Assignment (Project 3)	Douglas H. Fisher	2002-12-06	2002-12-12
Project3 Questions	Chad Ainsley Caldwell	2002-12-10	
Re:Project3 Questions	Nicholas Joseph Istre	2002-12-11	
Re:Project3 Questions	Douglas H. Fisher	2002-12-12	
Project 3 Question: Index Creation	Brandon O'Dell Nuttall	2002-12-11	

Figure 3: An example of the Theo interface, showing discussions about a particular assignment.

5.1 Task One

Users were asked to retrieve information from Theo and Prometheus as if they were constructing a to-do list of their assignments. The time it took the subjects to retrieve the data was recorded. Theo was significantly faster than Prometheus regardless of the presentation of the interfaces (F=48.7, p= $1 \cdot 10^{-5}$, MS= $6.7 \cdot 10^{4}$). Interaction effects were not significant. The mean Prometheus time was 126 seconds; the mean Theo time was 20.33 seconds.

5.2 Task Two

Users were asked to consult a static source of data (a syllabus) to retrieve a particular piece of data. The time it took the subjects to retrieve the data was recorded. There was no significant difference between Theo and Prometheus. The mean Prometheus time was 17.18 seconds; the mean Theo time was 19 seconds.

5.3 Task Three

Users were asked to consult a discussion to see if a response had been made to a particular posting. The time it took the subjects to retrieve the data was recorded. Theo was significantly faster than Prometheus regardless of the presentation of the interfaces (F=9.01, p=0.01, MS=837.8). Interaction effects were not significant. The mean Prometheus time was 26.82 seconds; the mean Theo time was 15.17 seconds.

5.4 Likert Scale Responses

To gauge the subjects' attitudes toward Prometheus and Theo, we administered a short survey after the tasks were completed. We used a five-point Likert scale, with a one corresponding to "strongly disagree," a two corresponding to "somewhat disagree," a three corresponding to "indifferent," a four corresponding to "somewhat agree," and a five corresponding to "strongly agree."

5.4.1 Ease of Use

Users were asked to rate how strongly they agreed with the statements "Theo is easy to use" and "Prometheus is easy to use." Theo was rated as significantly easier to use than Prometheus (F=26, p= $4 \cdot 10^{-5}$, MS=15.8). The mean Prometheus rating was "indifferent;" the mean Theo rating was "strongly agree."

5.4.2 Ease of Information Retrieval

Users were asked to rate how strongly they agreed with the statements "Theo makes it easy to retrieve the information I need" and "Prometheus makes it easy to retrieve the information I need." Theo was rated as significantly easier to use specifically when retrieving information than Prometheus (F=13.2, p=0.0015, MS=7.59). The mean Prometheus rating was "indifferent;" the mean Theo rating was "somewhat agree."

6. DISCUSSION AND FUTURE WORK

Users had a strong preference for Theo over Prometheus, and performed significantly better on two of the three tasks of our experiment, even considering that all of them had prior experience with the Prometheus system. These tasks were chosen based on the authors' experience that they are commonly performed by users. *A priori*, we expected that task one would favor Theo, task two would favor Prometheus, and task three would favor neither. In particular for task two, when entering the course partition in Prometheus, the Syllabus partition is displayed by default. Therefore only one click is required to retrieve syllabus information in Prometheus, versus two clicks for Theo. Task three requires two mouse clicks and page scanning for both Theo and Prometheus. Both interfaces present disadvantages for the task: in Theo, to see discussions for a particular class, users must use another mouse click on the filter pane; the formatting of discussion threads in Prometheus requires scrolling if more than four or five discussions are present on a 1024x768 display.

Task one, the creation of a to-do list, is something users should do frequently. Prometheus' performance is explained by its poor partitioning scheme and lax enforcement of content partitioning. In this task, we observed that the subjects looked in one of four different places for assignment data: the syllabus view, the outline view, the assignment view, and the file view. One subject even remarked verbally that unless he knew the professor, he couldn't find where the assignments should be. These observations suggest that Prometheus performance on this task would improve if the data were familiar to the user, and that educators are highly inconsistent about how they enter data into Prometheus. It is interesting to note that the Assignments content partition is not a top-level partition in Prometheus, but instead is accessed by clicking on an unemphasized HTML link within the Outline partition.

Task two, the retrieval of a static piece of information from the syllabus, was a task Prometheus should have done well in. By default on Vanderbilt's system, the syllabus is the data shown in the content table when a user first enters a course partition; also, users could take advantage of the web browser's built-in Find command to accelerate their search for the textbook's ISBN. However, only one user was observed using the Find shortcut to search for the requested data; most of the users scanned the data manually.

Task three, the consultation of course discussions, was designed so that both Theo and Prometheus should have performed equally; however, they did not. Approximately one-half of the subjects clicked on the discussion their theoretical user posted to see if there was a reply rather than simply looking at the "Replies" value, which was available from the main Discussions partition (though without any particular emphasis). This behavior makes Prometheus look disproportionately bad, although we don't know if the difference would be statistically significant.

Possible sources of bias in this study are due to the fact that Theo was only implemented locally. Theo could be at an advantage because its data is locally cached in the sample RDF files, and Prometheus could be at a disadvantage because it was being run over an 802.11b wireless link. Neither of issues are that significant: an optimal implementation of Theo would cache data locally and save that data between sessions, only contacting the server to update the local content headers and download new content on demand, whereas with Prometheus no local copies of content can be stored without manually saving HTML pages and files to disk. Also, while the 802.11b link may have imposed some additional latency, simple latency concerns cannot explain the time differences in task three, the task where both Prometheus and Theo were expected to perform equally.

Our study showed that the courseware environment is ripe for development of interfaces that improve upon what already exists, and that we made a step in this direction with the development of a XUL-based interface for a courseware system. In a broader sense, we have scratched the surface of a much larger trend in computer applications: the web-based interface. As discussed earlier, HTML, regardless of what extensions are employed, is not an ideal platform for the creation of a user interface. While using an interface presented in HTML is attractive from a cross-platform and universal-access point of view, it is not acceptable from the perspective of ease-of-use or user satisfaction. XUL is an ideal medium for deploying a cross-platform (and possibly universally accessible) interface.

In our development of Theo, we focused primarily on the creation of the user interface. Only a small portion of what can be called a "back end" was developed to allow users to test the interface. While this back end was designed with the possibility of extending it into a real, multi-user back end, the back end used in our user study made use of static data hosted on the local machine. Several mechanisms would need to be developed before this system could be used as an actual courseware application. The major remaining work would involve developing an "instructor" interface, developing a system for storing the files related to a class (e.g., lecture slides), and developing a system that dynamically generates the XUL for a given user. The last piece is necessary so that each user sees only his own classes listed in the left-hand pane of the interface.

One complaint that we heard from many of our test subjects is that educators use Prometheus inconsistently. Some post everything under a "files" section, while others may post everything under "discussions," or seem to randomly pick different categories in which to post items. In the case of Theo, the labels (and the accompanying ability to filter data by their labels) would be useless if the labels are incorrect. For example, if an "assignment" is posted, but it is flagged as a "file," any benefit gained from filtering is lost because a user must forego filtering and manually search through everything to find the desired information. This behavior presents a dilemma for the interface designer. It would likely be possible to restrict the user such that correct use of labels is nearly guaranteed; however, doing so would likely alienate advanced users who might wish to use the system beyond the initial developers' imagined scope. The most beneficial future work for this field would be to determine how best to ensure "proper" use of an application, without placing undue restrictions on a "power user" who would be annoyed or completely alienated by an application that prevents him from doing as he wishes.

7. ACKNOWLEDGMENTS

The authors would like to thank Jeanne Milostan for helpful suggestions on this project and the anonymous reviewers for their comments.

8. REFERENCES

[1] Blackboard, inc. `http://www.blackboard.com`.

[2] `http://www.edutools.info/course/compare/all.jsp`.

[3] Guzdial, M. Use of Collaborative Multimedia in Computer Science Classes. *The 6th Annual Conference on Innovation and Technology in Computer Science Education*. ISSN 0097-8418. June 2001, pp. 17–20. `http://coweb.cc.gatech.edu`.

[4] Fung, I. P. A Hybrid Approach to Represent and Deliver Curriculum Contents. *Proceedings of International Workshop on Advanced Learning Technologies (IWALT2000)*. ISBN 0-7695-0653-4, pp. 209—212.

[5] Nguyen, T., J. Newmarch, and J. Bird. Hypermedia in Educationn. `http://www.vacets.org/vtic97/tlnguyen.htm`.

[6] Calivi, L., and P. De Bra. Improving the Usability of Hypertext Courseware through Adaptive Linking. `http://wwwis.win.tue.nl/˜debra/flex97/`.

[7] Abowd, G. D. Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment. *IBM Systems Journal*, 38, 4, pp. 508-530, 1999.

[8] XUL 1.0. `http://www.mozilla.org/projects/xul/`

[9] Resource Description Framework (RDF) / W3C Semantic Web Activity. `http://www.w3c.org/RDF`.