DOCUMENT RESUME

ED 429 544 IR 019 452

AUTHOR	Salustri, Filippo A.
TITLE	Web-Based Course Delivery and Administration Using Scheme.
PUB DATE	1997-11-00
NOTE	7p.; In: WebNet 97 World Conference of the WWW, Internet & Intranet Proceedings (2nd, Toronto, Canada, November 1-5, 1997); see IR 019 434.
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	Computer Assisted Design; Computer Mediated Communication; *Computer Software Development; *Computer System Design; *Computer Uses in Education; Cooperative Learning; *Courseware; Foreign Countries; Higher Education; *Programming; Programming Languages; World Wide Web
IDENTIFIERS	HTML; University of Windsor (Canada)

ABSTRACT

This paper discusses the use at the University of Windsor (Ontario) of a small World Wide Web-based tool for course delivery and administration called HAL (HTML-based Administrative Lackey), written in the Scheme programming language. This tool was developed by the author to provide Web-based services for a large first-year undergraduate course in computer-aided design. Various problems encountered in administering the course are discussed, and a solution is proposed that includes formation of student design groups, maintaining registration information, facilitating communications, disseminating information, and assigning and recording grades. The design and implementation of HAL is then briefly described, and examples are given. Based on the success of HAL, it is concluded that--although larger and more complex course delivery systems are suitable in some circumstances--there are situations in which simpler and smaller systems are better suited. Three figures present plain scheme code for an HTML (HyperText Markup Language) unordered list, an HTML unordered list using the Scheme CGI (Common Gateway Interface) library, and construction of a whole HTML page using the Scheme CGI library. (Author/AEF)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

Web-based Course Delivery and Administration using Scheme

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Filippo A. Salustri
Department of Industrial and Manufacturing Systems Engineering
The University of Windsor
Windsor, Canada
e-mail: salustri@uwindsor.ca

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
G.H. Marks
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ED 429 544

Abstract: This paper discusses the use of a small Web-based tool for course delivery and administration called HAL (HTML-based Administrative Lackey), written in the Scheme programming language. Various problems encountered in administering the course are discussed and proposed solutions are presented. The design and implementation of HAL is then briefly described, with examples. Based on the apparent success of HAL, it appears that though larger and more complex course delivery systems are suitable in some circumstances, there are situations in which simpler and smaller systems are better suited.

Introduction

Administering and delivering course material to large groups of students can be very difficult. This is especially true in courses which significant technical content or requiring laboratory or other group efforts by the students. Typical problems include: organizing student groups, scheduling tutorials or laboratories taking into account possibilities of course conflicts, availability of teaching assistants and other resources, updating and disseminating information to students without tremendous waste of paper, controlling student evaluation, managing and tabulating student marks/grades, and ensuring that students who may not know one another can communicate with each other remotely (over a computer network, for example).

Clearly, there is a role that the World Wide Web can play to resolve these issues effectively. This paper discusses a tool being developed by the author to provide Web-based services for a large first-year undergraduate course in computer-aided design (CAD).

The Problem

The CAD course is a required course for all first-year undergraduate engineering students at the University of Windsor (currently, enrollment in the course is over 200). The goal of the course is to give the students some exposure to group-based engineering design techniques, especially in terms of concepts of *concurrent engineering* and *total design*. The following problems with administering the course and delivering material have been identified:

1. Students are required to work in small groups to actually design a product (this year, the product is a movable garden hose carrier). It is important to establish groups in some manner that hopefully maximizes the educational experience. A simple random selection process was not considered sufficient. Furthermore, since most first-year students are registered in the "General Engineering" program, it was impossible to use the chosen fields (mechanical engineering, electrical engineering, etc.) as a basis for group formation.
2. The University Registrar's Office is unable to provide an electronic list of student enrollment that is current. Students are allowed to drop a course - even a "compulsory" one such as CAD - up to two months after the start of the term. As enrollment changes, groups may need to be rearranged. Group management becomes a semester-long task.

IR019452

BEST COPY AVAILABLE

3. Students must be able to communicate with each other effectively using electronic mail. Every undergraduate student has a computer account for the duration of his/her program. Group-based mailing lists were an obvious mechanism to expedite communications, but the administrative load of expecting the University to create and administer such lists on a semester-by-semester basis was unacceptable.
4. Disseminating information to such a large group of students is a difficult task. Photocopying alone can significantly drain already thinly stretched teaching resources, and can involve a tremendous waste of paper. Getting information to students in a *timely* manner also becomes problematic when using paper.
5. Six teaching assistants are assigned to the CAD course, one for each laboratory/tutorial session. Tutors are charged with supervising their students, and marking progress reports for individual students and groups. The marks must be tabulated and stored consistently to simplify the assignment of final grades at the end of the semester. No standardized way of doing this currently exists in the department.

Formulating a Solution

Obviously, given the nature of the course, there is great potential for *Web-based* course delivery and administration in this situation. Two existing systems, [WebCT] and [Virtual U], were investigated. Both systems are quite large and provide many facilities besides those required for the CAD course. The administrative overhead associated with acquiring, installing, and maintaining the software, as well as designing the courseware itself, is considerably more than is reasonable at this time in the author's faculty.

The author also has an interest in developing Web-based administrative tools for educational and administrative environments. Most of the author's current research involves the Scheme programming language; it was considered desirable to be able to integrate the delivery system for the CAD course with the author's existing work.

In light of these issues, it was decided that a smaller, simpler solution should be tried. The system to be developed would be designed to meet the above-mentioned requirements, and possible to have the flexibility to be extended in the future, should there be sufficient interest. The proposed solution includes the following components.

Formation of student design groups. The use of so-called "personality tests" has found some popularity in industrial settings. There is also research to suggest that student design groups set up to have diversity of personalities have performed better than groups established by other means [Wilde 93]. A small personality test was available to the author; the test is a component of the delivery software. All registered students are required to take the test; the results are presented to them in a form they can understand. The results, along with course schedule information, are also used to assign students to groups. A simple algorithm was devised that automated the assignment process. A student would simply take the personality test, and as a result be assigned to a particular design group and laboratory/tutorial session. As enrollment changes, individuals can be automatically re-assigned to other groups as required.

Maintaining registration information. Since information from the University Registrar's Office was useless, the delivery system would have to allow students to "register" for the course. This allows enrollment information to be gathered quickly, and in a form immediately useful by the system.

Facilitating communications. The registration component of the delivery system requires students to supply the login of their University computer account, which is identical to their local e-mail address. This information, combined with the design groups database, is used to provide students with the means to e-mail messages to each other by name rather than login (many students do not know each other's login names), as well as sending messages to every member of their group without necessarily knowing the others' e-mail addresses. This is intended to allow groups to communicate outside of the assigned tutorial sessions.

Disseminating information. The system can alert students, by group or individually, that some information on the Web pages for the course has been updated. This allows the dissemination of information more efficiently, without tremendous waste of paper, time, etc. and with a higher degree of assurance that each student will actually be aware that updates have occurred. All course information except for lecture material is kept in Web pages that each student can access via the delivery system.

Assigning and recording grades. Since every student registers for the course via the delivery system itself, there is ample information to develop a unified internal database able to maintain grades assigned by the teaching assistants. Teaching assistants are recognized by the system, and are allowed to use it to enter grades. The system automatically tabulates semester and final grades based on this input.

Implementing the Solution

The course delivery system is implemented as a single CGI program able to return whatever HTML pages and forms are required. The program is able to develop pages dynamically, in response to user input, and to carry out other computations, such as assigning students to groups, and maintaining various databases. The Scheme programming language was used to implement the CGI.

The Scheme Programming Language

Scheme [IEEE 90] is a formalized dialect of Lisp. The particular implementation used, SCM by Aubrey Jaffer [Jaffer], supports the base language as well as POSIX-compliant extensions for the manipulation of files, and has a large library of portable packages. SCM programs can also be called as batch files, which allows it to be used for CGIs.

Scheme was used because it is a very simple yet powerful language, it has a very small "footprint" (significantly smaller than Perl), and is very efficient. It is also the language used by the author for a number of other research efforts.

The Scheme CGI

The first step in implementing the system was to develop a library of Scheme functions to facilitate creating HTML pages. The ultimate result was more than just able to decode CGI query strings, etc. The Scheme CGI in fact implements functions that have direct equivalents in HTML. While this might seem to simply duplicate what HTML already provides, the approach allows a far higher degree of integration of HTML and Scheme. A single consistent syntax allows programmers to develop Scheme programs that can also create and query arbitrary HTML pages, without concern for the syntax of HTML itself.

For example, without the Scheme CGI library, one might define the following function to display on the standard output a list of test string arguments as a compact unordered list in HTML, one string per HTML list item.

```
(display "<UL COMPACT>") (newline)
(for-each (lambda (string)
            (display "<LI>")
            (display string)
            (display "</LI>")
            (newline))
          a-list-of-strings)
(display "</UL>") (newline)
```

Figure 1: Plain Scheme code for an HTML unordered list.

The Scheme CGI library, however, implements various functions that allow such lists to be nested, or contain other HTML constructs, which the above function does not do. For example:

```
(unordered-list
  (!compact)
  (item "This is the" (bold "first") "string.")
  (item (unordered-list
          (item "The first sub-item.")
          (item "The" (italics "second") "sub-item.")))
  (item "The third string."))
```

Figure 2: An HTML unordered list using the Scheme CGI library.

The CGI query string is decoded as a list of name/value pairs in a Scheme variable called `*qargv*`, and values can be searched for by name with the function `getarg`. All the environment variables passed to a CGI are available as well (e.g. the function `(server-software)` returns the value of the `SERVER_SOFTWARE` environment variable. All the structures in HTML 3.2 are represented by Scheme functions. Attribute flags, like "`!compact`" above, are by convention named with an initial exclamation mark; attributes that take such values, such as "`:name`" start with a colon and take a single argument that can be a text string or another Scheme CGI library item.

Constructing an HTML page is quite simple. The following example shows how the initial login page is constructed for the CAD course system.

```
(html (head (title "85-131: Computer-Aided Design"))
      (body (:color/bg "black")
            (:color/text "white")
            (heading 1 (hal-logo) "Access Verification Page")
            (paragraph "Hi.  I'm HAL, the HMTL-based Administrative
                        Lackey, for this course."
                        :breakline
                        "To continue, you need to enter your"
                        (italics " login") " and your"
                        (italics " password.")
                        "When you're done, hit the"
                        (bold " submit") " button.")
            (form (:action "/cgi-bin/fil/85-131")
                  (:method "get")
                  (input "hidden" (:name "action") (:value the-action))
                  :breakline
                  (bold "Enter Login here:")
                  (input "text" (:name "login") (:size 10))
                  :breakline
                  (bold "Enter Password here:")
                  (input "password" (:name "pw") (:size 10))
                  :breakline
                  (input "submit"))))
```

Figure 3: Construction of a whole HTML page using the Scheme CGI library.

This will return a structure containing all the HTML fragments needed to product a Web page. The function "`/show-page`" is used to actually output this structure on the standard output.

The complete library is quite small, less than 50KB of commented Scheme code, and runs at speeds at least comparable to equivalent Perl code.

BEST COPY AVAILABLE

The HTML-based Administrative Lackey

The HTML-based administrative lackey (or "HAL") is a Scheme program that uses the Scheme CGI to centralize all the services needed for a given course, in one, integrated unit. The program simply returns an HTML page corresponding to the requested query. In the progress of creating the pages, the program may load various other files or perform other calculations.

The first time a user accesses HAL, by selecting the appropriate link on the web page of the CAD course, the user is asked to log in. Only registered students (and teaching assistants and instructors) are allowed passed this point. HAL searches for a student file by the login/password combination. The student file contains information such as the student's name, personality type, design group number and a list of the student's marks. If a student who is not registered attempts to log in, the student is presented with a registration page asking for the student's full name. This page also allows the student to take the personality test. Once this information is provided and the test is taken, HAL assigns the student to a design group, updates all relevant databases, and presents the student with the "main" page. This page gives the student the options of viewing various other pages relevant to the course, advices the student regarding recent updates and announcements, and provides the means for the student to send messages to the other members of his/her group, a teaching assistant, or the instructor. The student may also view his/her record, including grades, in a read-only format.

Finally, HAL can identify when it is running as a CGI and when it has been invoked from a simple command-line. In the latter case, the program will run only for the administrator and will allow various specialized tasks, such as identifying teaching assistants and instructors, initializing databases, etc.

Future Directions

Clearly, HAL is not intended to be as all-encompassing as are some of the other Web-based course delivery systems mentioned earlier. However, it is ample for the purposes of the CAD course taught by the author. If the HAL facility meets with success, the author will bring it to the attention of other instructors, in the hope that it will help them as well.

In this eventuality, it will be necessary to create a *meta*-HAL facility to assist other instructors in developing other HAL programs for their own courses. It is also possible that other administrative programs, similar to HAL, may be developed to assist in the daily administration of the author's department.

Conclusions

This paper has discussed the problems of administering a large class, and how a simple, small, and efficient language like Scheme can be used to quickly construct a system for aiding in the delivery of material and course administration. It appears that the HAL system will significantly improve the efficiency by which administrative details are dealt with, freeing the instructor and teaching assistants to focus their attention on helping the students learn. The larger systems that are available may be well suited in some circumstances, but clearly there are other situations for which a simpler, smaller system is advised.

References

[WebCT] World-Wide Web Course Tool, http://homebrew.cs.ubc.ca/webct/.

[Virtual U] The Virtual University, http://www.emediadesign.com/sfu/courses.html.

[Wilde 93] D. J. Wilde. (1993). Mathematical resolution of MBTI creativity data into personality type components, *ASME Design Theory and Methodology Conference*, T. K. Hight, ed. 37-43.

[IEEE 90] IEEE. (1990). *IEEE Standard for the Scheme programming language*, IEEE Std 1178-1990.

[Jaffer] Aubrey Jaffer's Meta Home Page, http://www-swiss.ai.mit.edu/~jaffer/index.html.

ERIC

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

REPRODUCTION BASIS

☒ This document is covered by a signed "Reproduction Release (Blanket) form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐ This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").