

BUILDING A DATABASE AND SEARCH ENGINE FOR REUSE OF COURSE MATERIALS

Edward F. Gehringer¹ and Tony M. Louca²

Abstract $\frac{3}{4}$ We have developed software for creating databases of course materials on the World-Wide Web. The goal is to allow instructors at different institutions to share materials and develop them jointly. Our first two databases, in computer architecture and object technology, comprise thousands of problems and lectures downloaded (with permission) from course Websites around the world. The database is searchable by classification or fulltext string. To populate the databases with up-to-date material, we started by building a list of course Websites. Using several sources, we came up with a list of 73 sites in computer architecture and 40 sites in object technology. However, only a minority of authors gave us permission to load their material. To provide access to a larger amount of material, we are extending our search capability to include material on the Web as well as in the database. The search engine that we have used finds course Websites by searching a filtered set of educational domains for sites containing keywords characteristic of course material in the target discipline. Finally, we report on our experience of building software using a combination of funded research assistants, students working on senior-design projects, and independent-study students..

INTRODUCTION

With the advent of the World-Wide Web in the early '90s, instructors began to place course material on line. In 1995, academic attendees from the International Symposium on Computer Architecture indicated great interest in developing a Website of reusable course materials. By 1997, approximately half of the object-technology (OT) instructors attending a workshop organized by the first author had developed course Websites. Contributions were sought, and approximately 500 problems were obtained from nine different contributors. The database went online in 1998. In the beginning, questions were inserted by cutting and pasting them into a browser interface to the database. To automate this time-consuming process, a set of Perl scripts was developed. The scripts would take a URL pointing to the course Website and a regular expression. Based on the regular expression, the script would download all the homeworks (or other course material) it found and try to separate them into individual problems. Then, these problems would be inserted in the database. These scripts, however, often

required user intervention and did not keep track of where the material had been loaded.

WEBASSIGN

The database schema and search functionality of our database, on the other hand, were robust from the outset. Material is stored [1] in WebAssign [2], a Web-based multimedia exam and homework-grading system developed in the Physics department at NCSU. This gives our course database the search functionality and Web accessibility of the physics database. Although they share software with WebAssign, our course databases are totally separate from the physics database.

Teaming up with an existing on-line testing system freed us from the need to do database programming, and thus permitted us to bring up a small system with only a few thousand dollars of internal funding. Eventually, it will allow problems in the database to be used for quizzes administered over the Web and graded automatically.

As our database progresses, however, the limitations of WebAssign are becoming more apparent. First, information can only be entered into WebAssign through a limited number of fields defined by the database schema. These encode much of the information we need to store (e.g., author, degree of difficulty, related textbook), but do so in their own specific way. We would prefer to encode this information as IMS metadata [3], which is expected to achieve widespread usage. But IMS metadata is based on the XML standard, which, unfortunately, is not supported by WebAssign. Second, WebAssign does not offer printing and download functions, which would be useful for instructors trying to reuse the material. Therefore, we have reimplemented the "front end" of WebAssign to meet our needs and implement some of our users' suggestions. We will continue to store questions in WebAssign format, however, so that the ones that have an objective answer can later be used in machine-graded assignments.

Compared to a database, a specialized search engine has both advantages and disadvantages. Combining a search engine with a database seems to be the best approach, as detailed below in the section entitled "Search Engines."

¹ Edward F. Gehringer, Department of Electrical & Computer Engineering, Dept. of Computer Science, North Carolina State University, Raleigh, NC 27695-7911, efg@ncsu.edu

² Tony M. Louca, NVIDIA Corporation, 3535 Monroe Street, Santa Clara, CA 95051, tmlouca@eos.ncsu.edu (formerly Dept. Electrical & Computer Engineering, NCSU, Raleigh, NC).

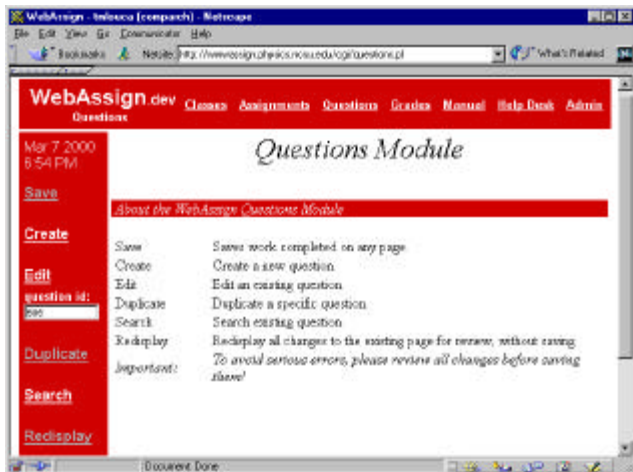


Figure 1: Question module

A SHORT TOUR OF THE DATABASE

A user logging into the database is taken directly to the WebAssign questions module (Figure 1). Clicking on “Search” on the left-hand menu brings up a form that gives several search options. A search may be performed by keyword, fulltext, limited to a certain author’s contributions, or to questions submitted by authors using a certain textbook (Figure 2). Then a set of search results is brought up (Figure 3), listing the first line of each question, and giving the user the option to view the text of an item (Figure 4), edit it (this option is only available to the question’s author), or duplicate it, perhaps in preparation for creating a different version of the question. The revised question may then be saved back to the database. The user may even insert questions directly into the database via the “Create Question” screen (Figure 5); however, only one or two authors have ever used this mode; in almost all cases, the questions in the database have been loaded over the Web. A more detailed tour of the database is presented in [4].

HOW TO FIND CONTRIBUTORS

While our current repertoire of hundreds of questions is useful to many instructors (we currently have about 70 users; results of a survey of them are reported in [5]), coverage of some topics is very thin, and material specifically geared to any but the most popular textbooks is hard to come by. Clearly, we need to find more instructors willing to contribute their online material under a royalty-free agreement. We have used the following approaches to contact instructors:

- We sent e-mail to people with academic affiliations on conference attendance lists.
- We used Web-based listings of faculty in particular disciplines. For the computer architecture course database, we contacted people listed on the WWW Computer Architecture Website [6]. For the object technology database, we used Steve Beaty’s list [7] of computer-

science course Websites. We also used the list of course Websites that the primary author generated for last year’s OOPSLA Educators’ Symposium [8].

- We used attendee lists from several specialized education workshops.
- We have done presentations and demos at conferences like the American Society for Engineering Education 1999 annual conference, SIGCSE 2000, Frontiers in Education 2000, and the 2000 OOPSLA Educators’ Symposium.
- We have used Web search engines to find course Websites on related material and then contact the instructors.

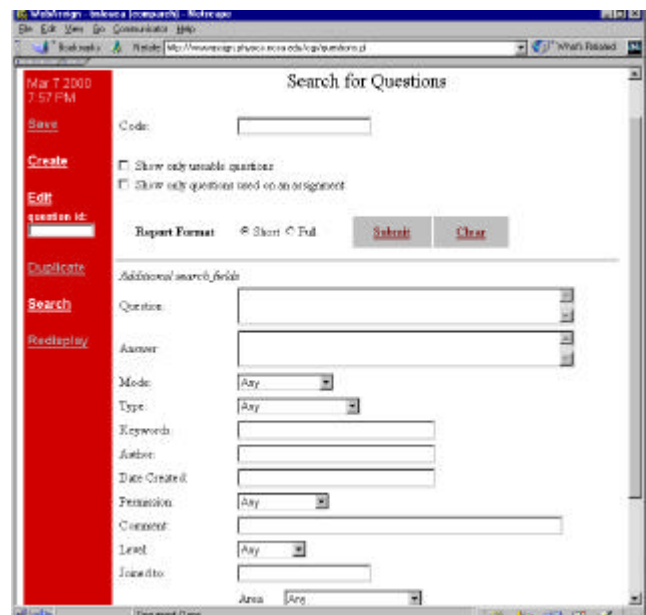


Figure 2: Search screen

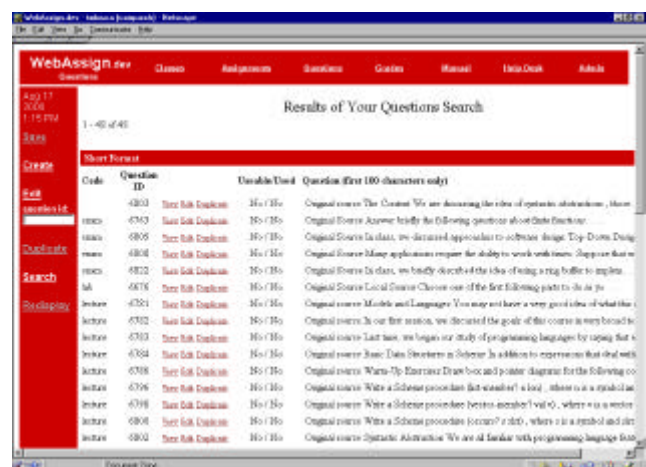


Figure 3: Search results

In addition, existing users of the database referred many users to us. Dave Patterson of UC-Berkeley deserves special mention for referring potential users.

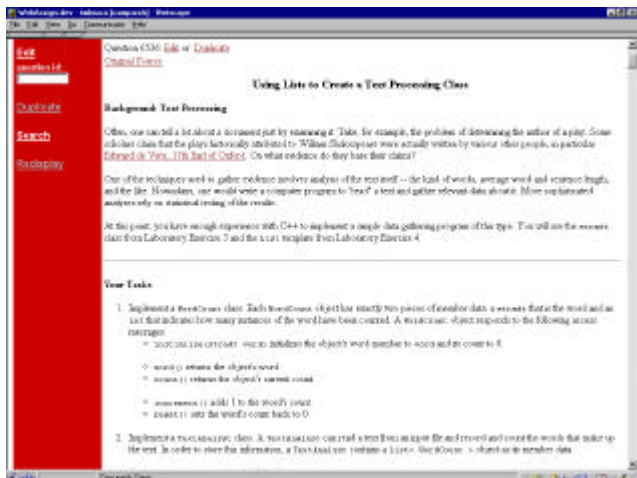


Figure 4: Search result (full)

HOW WE SOUGHT PERMISSION

We used e-mail to ask instructors for permission to download their course material. One month after of sending the original message, we sent a reminder for people who had not responded.

For computer architecture, we sent requests to 73 instructors. Twenty-nine agreed to give us permission, while 7 declined.

For object technology, we sent requests to 40 instructors. Sixteen agreed to give us permission, while 5 declined. The remainder did not reply.

The reasons instructors gave for turning down our request were about equally divided between two factors.

- **Copyright concerns.** Some instructors wanted to use their material in books they planned to write, and were afraid that making it available in advance might compromise their ability to do this. Other instructors had taken material from existing texts, and were concerned that they would have to seek permission from the copyright holders.
- **Diffidence.** Several instructors felt that their material was not polished enough to be used in the database, either because they were teaching a course for the first time, or because they had not been able to devote adequate attention to it. This is a common concern expressed by instructors regarding their course material [9]. Unfortunately, acting on that concern tends to compound the problem, since if material is not made available, others will not be able to help improve it.

We addressed the copyright issue by posting the following disclaimer on the login page of the database.

“The authors of the material in this database grant authorized users of this database permission to reuse it for educational purposes in their own courses. Any further republication of the material requires the express consent of each author whose intellectual property is being reused.”

The author of the material is also required to certify that the work (s)he is submitting is original. This is an advantage of using a database instead of a specialized search engine, which would have no easy way to keep track of copyright.

The diffidence problem has been attacked by combining the database with a search engine, as described below.

Another problem we faced while downloading course questions is the painful task of associating solutions with homeworks. Not all instructors publish their solutions and if they do they may not put them on the Web (another disadvantage of relying on the search-engine approach). Right now, for example, out of 883 question in computer architecture, 590 have solutions.



Figure 5: Create question screen

HOW INFORMATION IS LOADED

To replace the Perl scripts used early in the project, we have developed a Java application to automate the process of downloading from the Web. The application takes a wildcard URL and a set of editing instructions. It will search the URLs for all problems, labs, lectures, etc. that match the editing expression and download them to the local machine as separate files. It can fetch solutions to problems from separate URLs (since instructors often produce separate question and answer handouts). The application will then divide the handouts into separate files for each question, and upload them to the WebAssign database. The application will process information in either HTML or ASCII formats. When other formats are submitted (MS Word, PDF, etc.) a converter is used to translate the file to HTML before processing.

The application also provides management functions, such as maintaining contact and portfolio information for each instructor, checking course Websites for new material, and keeping local copies of each downloaded homework or lecture. Each problem in the database contains both a link to the “original source”—the Web page from which it was

taken, and a link to the local copy. The “original source” link is used to view the problem in the context of the assignment from which it was taken. This is useful in case a common set of assumptions apply to all problems in a problem set, or in case one problem is to redo a previous problem with a different set of conditions. The “local copy” link serves as a backup for the original source in case the original source is moved or removed after the end of the term when the course is taught.

The “original source” link also provides a way to access the most recent version of a problem. For example, if an instructor assigns homework and then discovers an error, (s)he may update the handout on the Web. The “original source” link will access the current version. Also, the application periodically goes back to each URL to check for new versions of problems and replaces the old version of these problems in the database. Both of these mechanisms help keep the database up to date.

Future development of the application will address some of its limitations. Currently, material in PDF, Postscript, or other formats cannot be processed automatically. We need to find converters to translate from these formats to HTML. Second, many homeworks have embedded images which are most probably located on the course website. The links break when these images move. To solve this, we must download these images to a local file space, building an internal link, and inserting it in the homework in place of the original image link. Finally, we are working on an interface to an editor so that the user can view each item as it is entered into the database and manually edit it, in case the application has made a mistake in separating one problem from another.

SEARCH ENGINES

Our experience in seeking permission to download material to the database makes it clear that we will never succeed in loading all of the relevant material. Therefore, it seems appropriate to combine the database with a search engine. Users of the database do not, of course, have an automatic right to reuse and adapt material that is not in the database; however, they are able to ask the copyright holders for permission individually. Although we have no experience yet, we expect that an author might well grant a single instructor permission to reuse material, even if it is not deemed ready for wider dissemination.

The search engine and database each have advantages and disadvantages.

- A search engine is easy to program to find a large amount of material, while it is time consuming to build a database.
- A search engine is capable of providing access to material only as long as it stays on the Web (which is frequently only until the end of the current academic term). A database retains material permanently.

- A search engine is harder to use than a database. It will inevitably return many pages that are not reusable course material.
- If material is updated, e.g., to remove bugs, a search engine will always find the most up-to-date material. A database will find it only if the database has been updated since the material changed.

The approach we have used to building a search engine is as follows:

- Program an existing search engine to find course Websites by selecting keywords from course syllabi in a particular area, and having the engine return links to pages containing those links. NorthernLight has been chosen, because it can be targeted to search only academic sites in a number of countries.
- Search those course Websites by using terms that are typical of homework problems, lab exercises, etc.
- On each search, give the author the option of searching only the database, or the database and search-engine results.

We expect that authors who fear copyright violations would be more likely to give permission to a *single* user to use a *single* item than to grant blanket permission to all users of the database. Eventually this technology could provide a fast way to bring up a database of materials in a new academic field: The search engine bootstraps a new database full of links; this is advertised to instructors who find it useful and contribute their own materials.

DEVELOPMENT AND STATUS OF THE PROJECT

It is always difficult to get adequate help with a software project. The average grant in this field will support only a single graduate student. Our approach is to seek other student help, for course or project credit, and use the supported student primarily as a project manager. Various aspects of the project have been the subject of three ECE senior design projects. The students are responsible to ECE’s Troxler Design Center for project agreements, milestones, and final reports. Technical supervision is provided by the PI and research assistant. In addition, the project has been helped by more than a dozen students working on independent-study projects for graduate credit. This is good for the student, who gets experience working on a real research-and-development project, and it is good for the project, which benefits from additional manpower. However, a major concern is continuity, since there is an almost complete turnover of personnel each semester. We have begun using pair programming [10] to alleviate this problem: two students work together on all aspects of their code; then when one leaves, the other can carry on. Fortunately, we have had several students who desire to begin a project in one semester and finish it in the next. We have them register for the latter semester, and pair them with someone who is working on a project for only a single semester.

At the time of writing, the computer architecture database contains approximately 1000 entries for computer architecture material. The object technology database, which we started to develop this year, has 286 entries, most of them homework and programming assignments with 19 lectures.

FUTURE WORK

Another approach to expanding the supply of problems is to get permission from textbook publishers to load their questions into our database. This is an avenue that has been pursued with great success by the WebAssign project with regard to physics textbooks. We have secured permission to include problems from two computer-architecture textbooks. One of the most promising approaches to increasing the accessibility of materials is to link our database up with a search engine. This is a three-step process.

Finally, we are investigating ways to maintain reuse counts of items in the database. We plan to collect information and keep track of when each item is reused. A high reuse count indicates that other instructors find quite useful. Moreover, a field can be provided for reuse reports, where authors who reuse a problem can report on their experiences. When fully developed, reuse counts and reviews can be cited by authors as evidence of excellence in teaching—opening the door to the same kind of qualitative and quantitative productivity measures that are now used to assess research contributions.

ACKNOWLEDGMENTS

This work has been supported by the Course, Curriculum and Laboratory Improvement program of the National Science Foundation under grant DUE 99-50318. The authors would like to acknowledge the help of more than two dozen NCSU students, including Hassan Shehab, Randall Noel, and Suping Li, who were supported by the

grant, Luis Noguera-Gonzalez, who implemented XML support in the database, and Joshua Jensen, Greg Pearman, and Brian Nobles, who programmed the search engine.

REFERENCES

- [1] Edward F. Gehringer, Ana E. Goulart, Xiaokang Sang, and Chenhao Geng, "Computer Architecture Course Database: Implementation and Status Report," Workshop on Computer Architecture Education, held in conjunction with the 25th *International Symposium on Computer Architecture in Barcelona*, Spain, June 1998 (<http://www4.ncsu.edu/~efg/isca-caew98.ps>).
- [2] Aaron P. Titus, Larry W. Martin, and Robert J. Beichner, "Web-based testing in physics education: methods and opportunities," *Computers in Physics* 12:2, March/April 1998, pp. 117–123.
- [3] The IMS meta-data project <http://www.imsproject.org/metadata/>
- [4] Edward F. Gehringer & Tony M. Louca, "Using the Computer Architecture Course Database," Proc. Workshop on Computer Architecture Education, Vancouver, June 10, 2000, pp. 85-89, also in September 2000 IEEE Technical Committee on Computer Architecture Newsletter
- [5] Edward F. Gehringer & Tony M. Louca, "A Web-Based Computer Architecture Database: Improving the Viability," Proc. Frontiers in Education 2000, Kansas City, Oct. 18-21, 2000
- [6] The WWW computer architecture maintained by Milo Martin <http://www.cs.wisc.edu/arch/www/>
- [7] Beaty, Steve, "Resources for university teaching," <http://lamar.colostate.edu/~beaty/>
- [8] Edward F. Gehringer, "A tour of course Websites," OOPSLA '99: Object-Oriented Programming Languages, Systems, and Applications (Educators' Symposium), Denver, CO, Nov. 1-5, 1999
- [9] Lillian Cassel, SIGCSE award luncheon address, 32nd SIGCSE Technical Conference on Computer Science Education, February 24, 2001, <http://lcassel.csc.villanova.edu/sigcse.ppt>.
- [10] Laurie A. Williams and Robert R. Kessler, "All I really need to know about pair programming I learned in kindergarten," *Communications of the ACM* 43:5 (May 2000), pp. 108–114.